亚马逊云科技

用户指南

Amazon A EC2 uto Scaling

Amazon A EC2 uto Scaling: 用户指南

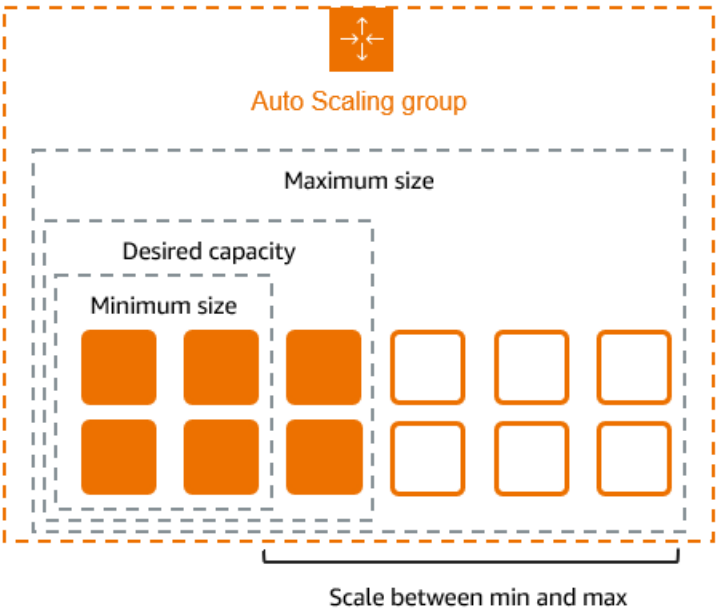
Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 中国的 Amazon Web Services 服务入门 (PDF)。

Table of Contents

什么是 Amazon A EC2 uto Scaling？

Amazon A EC2 uto Scaling 可帮助您确保有正确数量的亚马逊 EC2 实例可用来处理应用程序的负载。您可以创建 EC2 实例集合，称为 Auto Scaling 组。您可以指定每个 Auto Scaling 组中的最小实例数，Amazon A EC2 uto Scaling 可确保您的组永远不会低于此大小。您可以指定每个 Auto Scaling 组中的最大实例数，Amazon A EC2 uto Scaling 可确保您的组永远不会超过此大小。如果您在创建组时或之后的任何时候指定所需的容量，Amazon A EC2 uto Scaling 可确保您的组拥有这么多的实例。如果您指定扩展策略，那么 Amazon A EC2 uto Scaling 可以在应用程序需求增加或减少时启动或终止实例。

例如，以下自动扩缩组的最小大小为四个实例，所需的容量为六个实例，最大大小为十二个实例。您制定的扩展策略是按照您指定的条件，在最大最小实例数范围内调整实例的数量。



Amazon A EC2 uto Scaling 的特点

使用 Amazon A EC2 uto Scaling，您的 EC2 实例会被组织为 Auto Scaling 组，因此可以将其视为逻辑单元进行扩展和管理。Auto Scaling 组使用启动模板（或启动配置）作为其 EC2实例的配置模板。

以下是 Amazon A EC2 uto Scaling 的主要功能：

监控正在运行的实例的运行状况

Amazon A EC2 uto Scaling 会使用运行状况检查自动监控您的实例的运行 EC2 状况和可用性，并替换已终止或受损的实例以保持所需的容量。

自定义运行状况检查

除了内置的运行状况检查外，您还可以定义特定于您应用程序的自定义运行状况检查，以验证其是否按预期响应。如果某个实例未通过自定义运行状况检查，则会自动替换该实例以保持所需的容量。

跨可用区平衡容量

您可以为 Auto Scaling 组指定多个可用区，随着组的扩展，Amazon A EC2 uto Scaling 会在可用区域之间均衡您的实例。这可以保护您的应用程序免受单一位置故障的影响，从而提供高可用性和故障恢复能力。

多种实例类型和购买选项

在单个自动扩缩组内，您可以启动多种实例类型和购买选项（竞价型实例和按需型实例），从而能够通过竞价型实例使用量优化成本。您还可以将预留实例和节省计划折扣与组中的按需型实例结合使用，从而享受这些折扣。

自动替换竞价型实例

如果您的组包含竞价型实例，则在您的竞价型实例中断时，Amazon A EC2 uto Scaling 可以自动请求替换竞价型容量。通过容量再平衡，Amazon A EC2 uto Scaling 还可以监控并主动替换中断风险较高的竞价型实例。

负载均衡

您可以使用 Elastic Load Balancing 负载均衡和运行状况检查来确保将应用程序流量均匀分配给运行状况良好的实例。无论何时启动或终止实例，Amazon A EC2 uto Scaling 都会自动向负载均衡器注册和注销这些实例。

可扩展性

Amazon A EC2 uto Scaling 还为您提供了多种扩展 Auto Scaling 群组的方法。使用自动扩缩可以增加容量以处理峰值负载，并在需求较低时移除容量，从而保持应用程序可用性并降低成本。您也可以根据需要手动调整自动扩缩组的大小。

实例刷新

实例刷新功能提供了一种机制，可在您更新 AMI 或启动模板时以滚动方式更新实例。您还可以使用分阶段方法（称为金丝雀部署）在一小部分实例上测试新 AMI 或启动模板，然后再将其推广到整个组。

生命周期钩子

生命周期挂钩对于定义在新实例启动时或实例终止之前调用的自定义操作非常有用。此功能对于构建事件驱动架构特别有用，但它也可以帮助您在实例的整个生命周期中对其进行管理。

支持有状态的工作负载

生命周期挂钩还提供了一种在关闭时保持状态的机制。为确保有状态应用程序的连续性，您还可以使用横向缩减保护或自定义终止策略来防止具有长时间运行进程的实例提前终止。

有关 Amazon A EC2 uto Scaling 优势的更多信息，请参阅应用程序架构的 Auto Scaling 优势。

亚马逊 A EC2 uto Scaling 的定价

Amazon A EC2 uto Scaling 不收取任何额外费用，因此您可以轻松试用，看看它如何使您的 Amazon 架构受益。您只需为使用的 Amazon 资源（例如，EC2 实例、EBS 卷和 CloudWatch 警报）付费。

开始使用

要开始使用，请学完创建您的第一个自动扩缩组教程，以创建一个自动扩缩组并了解它在该组中的实例终止时如何进行响应。

使用 Auto Scaling 组

您可以通过下面的任何一种方式来创建、访问和管理 Auto Scaling 组：

- Amazon Web Services Management Console – 提供了可用来访问 Auto Scaling 组的 Web 界面。如果您已经注册了 Amazon Web Services 账户，则可以通过登录来访问您的 Auto Scaling 群组 Amazon Web Services Management Console，使用导航栏上的搜索框搜索 Auto Scaling 群组，然后选择 Aut o Scaling 群组。
- Amazon Command Line Interface (Amazon CLI) — 为大量用户提供命令，并在 Windows Amazon Web Services 服务、macOS 和 Linux 上受支持。要开始使用，请参阅 准备使用 Amazon CLI。有关更多信息，请参阅 Amazon CLI 命令参考中的弹性伸缩。
- Amazon Tools for Windows PowerShell— 为那些在 PowerShell 环境中编写脚本的用户提供一系列 Amazon 产品的命令。要开始使用，请参阅 Amazon Tools for Windows PowerShell 用户指南。有关更多信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考。
- Amazon SDKs— 提供特定于语言的 API 操作并处理许多连接细节，例如计算签名、处理请求重试和处理错误。有关更多信息，请参阅 Amazon SDKs。
- 查询 API – 提供了您使用 HTTPS 请求调用的低级别 API 操作。使用查询 API 是访问 Amazon Web Services 服务的最直接方式。但它需要您的应用程序处理低级别的详细信息，例如生成哈希值以签署请求以及处理错误。有关更多信息，请参阅 Amazon A EC2 uto Scaling API 参考。

- Amazon CloudFormation— 支持使用 CloudFormation 模板创建 Auto Scaling 组。有关更多信息，请参阅 使用 Amazon CloudFormation创建 Auto Scaling 组。

要以编程方式连接到 Amazon Web Services 服务，请使用终端节点。有关调用 Amazon A EC2 uto Scaling 的 ，以及中国亚马逊网络服务入门 Amazon 。

应用程序架构的 Auto Scaling 优势

将 Amazon A EC2 uto Scaling 添加到您的应用程序架构中是最大限度地发挥 Amazon 云优势的一种方式。当您使用 Amazon A EC2 uto Scaling 时，您的应用程序将获得以下好处：

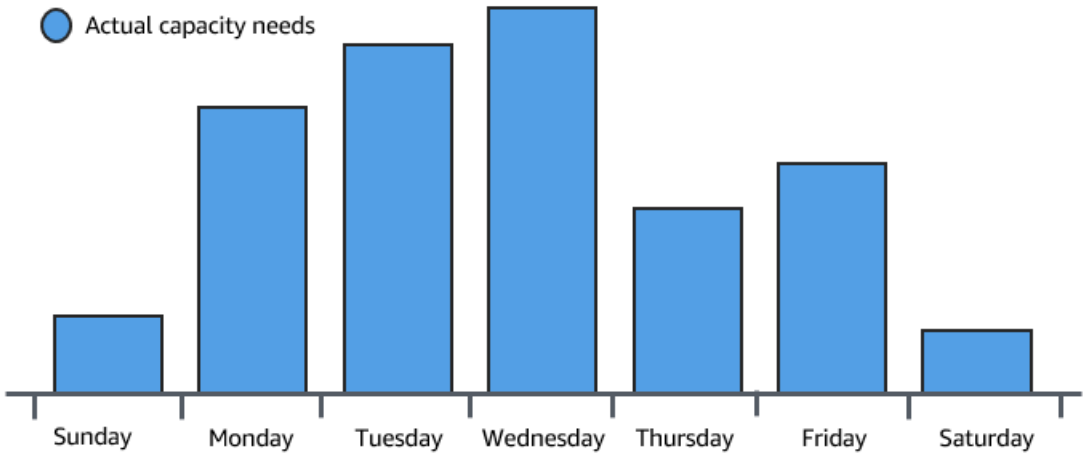
- 提高容错能力。Amazon A EC2 uto Scaling 可以检测实例何时运行状况不佳，将其终止，然后启动实例来替换它。您也可以将 Amazon A EC2 uto Scaling 配置为使用多个可用区。如果一个可用区不可用，Amazon A EC2 uto Scaling 可以在另一个可用区中启动实例以进行补偿。
- 提高可用性。Amazon A EC2 uto Scaling 有助于确保您的应用程序始终具有适当的容量来处理当前的流量需求。
- 加强成本管理。Amazon A EC2 uto Scaling 可以根据需要动态增加和减少容量。由于您需要为所使用的 EC2 实例付费，因此您可以通过在需要时启动实例并在不需要时将其终止来节省资金。

内容

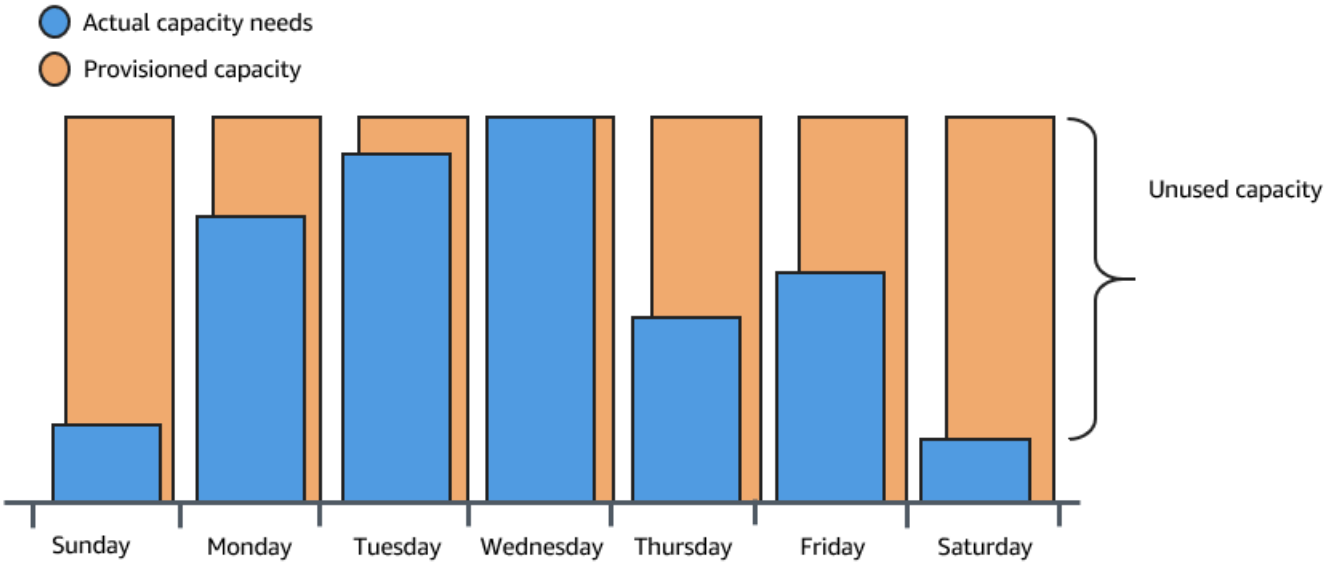
示例：覆盖可变需求

要演示 Amazon A EC2 uto Scaling 的一些好处，可以考虑在上面运行一个基本的 Web 应用程序 Amazon。此应用程序允许员工搜索可用于开会的会议室。每周开始和结束时段，此应用程序的使用率最低。每周中期，有更多的员工安排会议，因此对此应用程序的需求会显著提高。

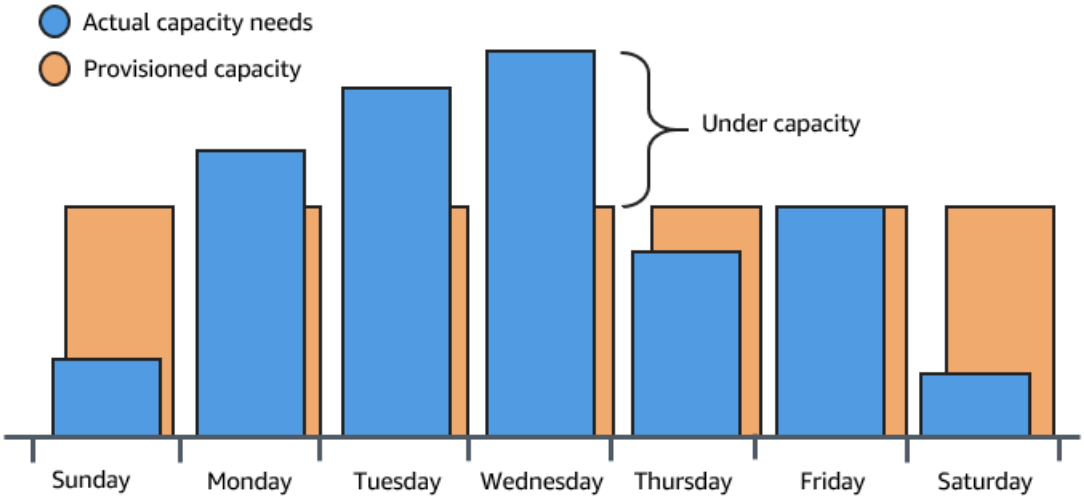
下图显示此应用程序的容量在一周中的使用情况。



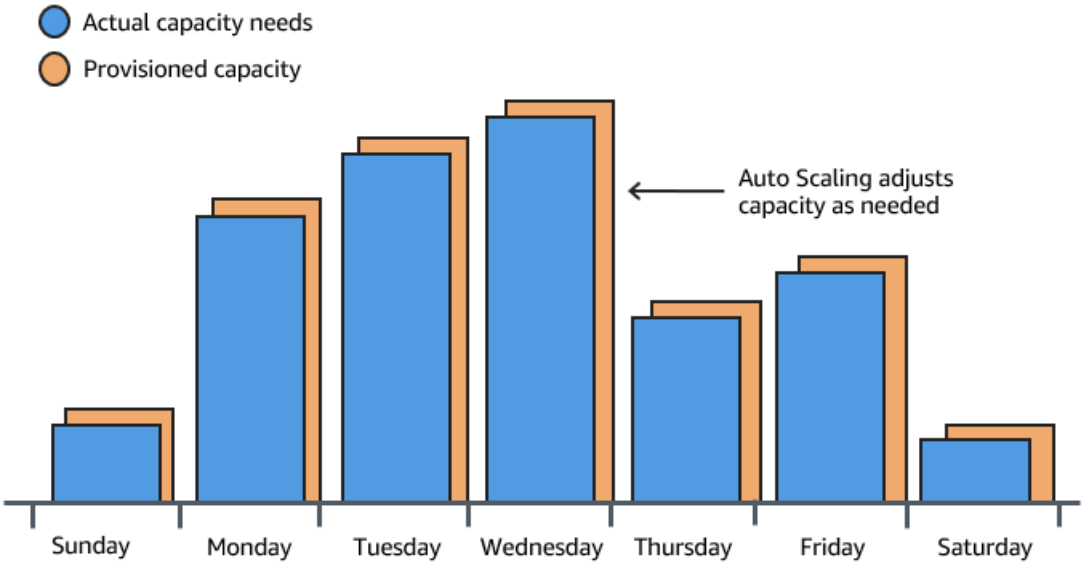
按照传统做法，可通过两种方式为这些容量变化做好规划。第一种选择是添加足够多的服务器，以便应用程序始终具有足够的容量来满足需求。但是，这种做法的缺点是应用程序在某些天并不需要这么多容量。额外容量闲置不用，并且实际上提高了使应用程序保持运行的成本。



第二种选择是采用处理应用程序平均需求所需的容量。这种做法成本更低，因为不用购买仅仅偶尔使用的设备。然而，这样做的风险是：当对应用程序的需求超过其容量时，可能造成糟糕的客户体验。



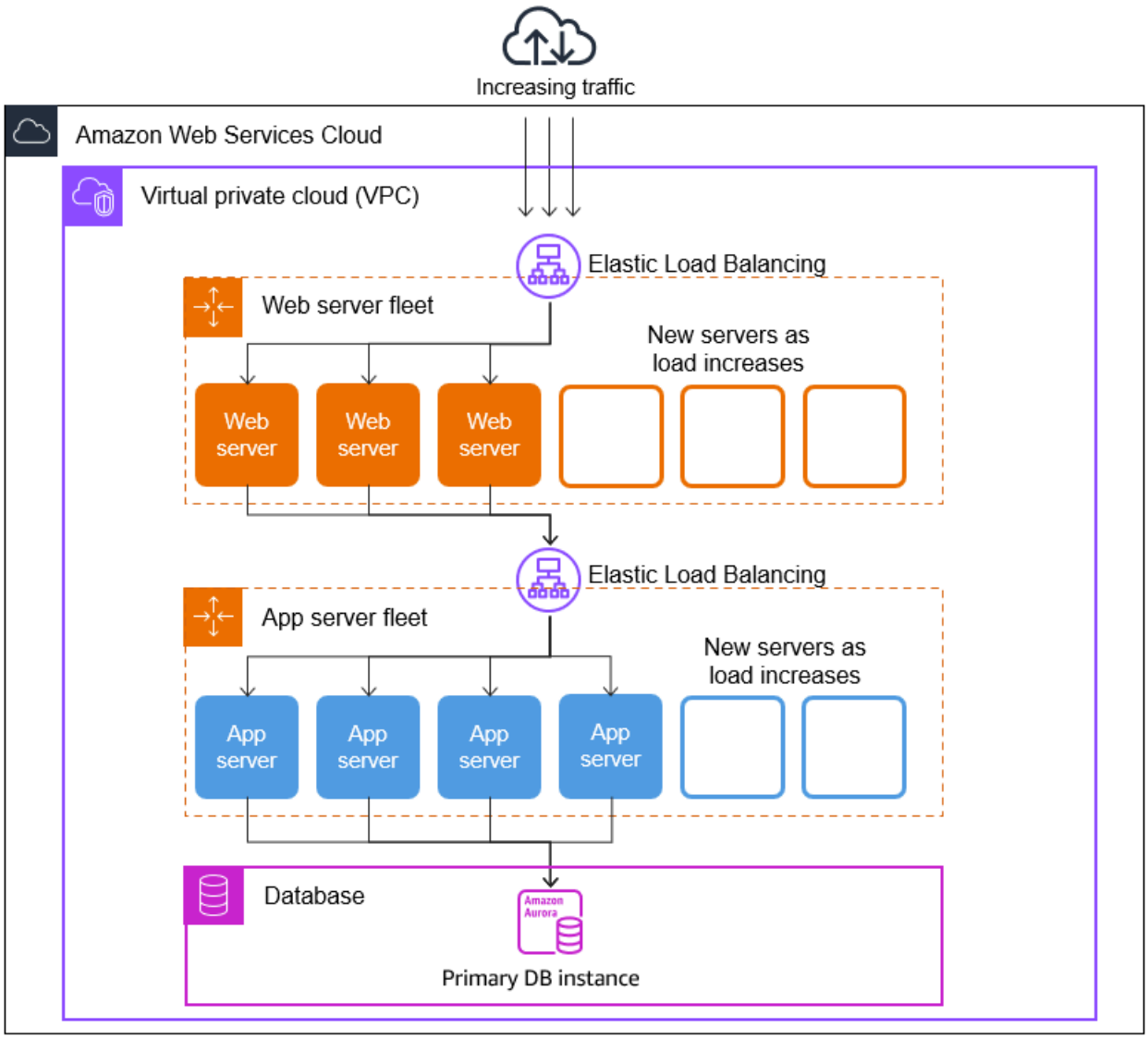
通过将 Amazon A EC2 uto Scaling 添加到此应用程序，您就有了第三种选择。您可以仅在需要时才向应用程序添加新实例，并在不再需要这些实例时终止它们。由于 Amazon A EC2 uto Scaling 使用 EC2 实例，因此您只需在使用实例时为所使用的实例付费。您现在有了一个具有成本效益的架构，可在尽量减少支出的同时提供最佳客户体验。



示例：Web 应用程序架构

在常见的 Web 应用程序场景中，您同时运行应用程序的多个副本来满足客户流量。您的应用程序的这些多个副本托管在相同的 EC2 实例（云服务器）上，每个副本都处理客户请求。

Amazon A EC2 uto Scaling 代表您管理这些 EC2 实例的启动和终止。您可以定义一组标准（例如 Amazon CloudWatch 警报），用于确定 Auto Scaling 组何时启动或终止 EC2 实例。将 Auto Scaling 组添加到网络架构有助于提高应用程序的可用性和容错能力。



您可以根据需要创建任意数量的 Auto Scaling 组。例如，您可以为每个层创建一个 Auto Scaling 组。

要在您的 Auto Scaling 组的各实例之间分配流量，可在您的架构中引入一个负载均衡器。有关更多信息，请参阅 Elastic Load Balancing。

示例：在可用区之间分配实例

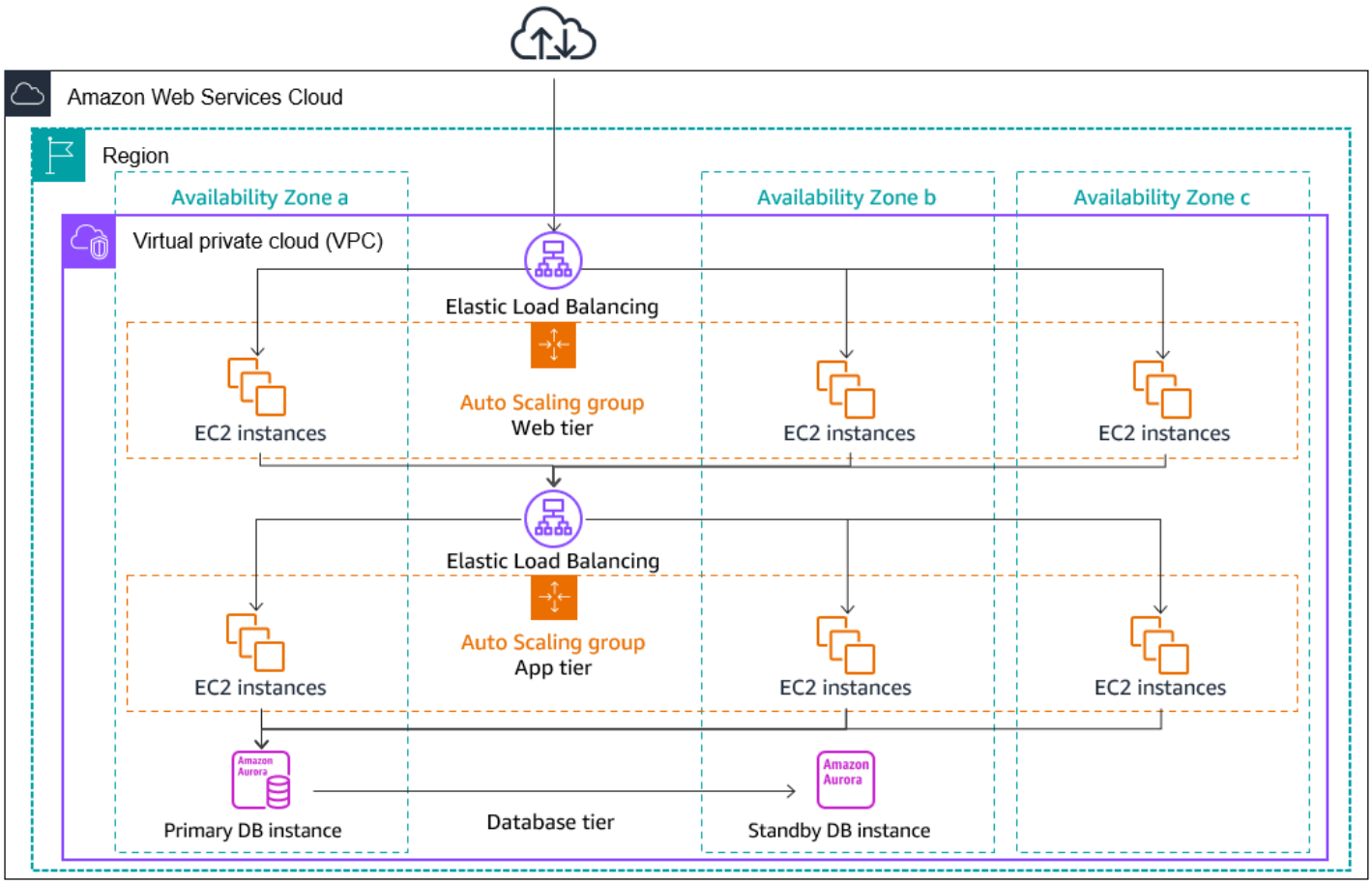
可用区是给定 Amazon Web Services 区域中相互隔离的站点。每个区域都有多个可用区，旨在为该区域提供高可用性。可用区相互独立，因此，设计为使用多个可用区的应用程序可以提高应用程序的可用性。有关更多信息，请参阅 Amazon A EC2 uto Scaling 中的弹性。

可用区由 Amazon Web Services 区域 代码和字母标识符进行标识（例如，`us-east-1a`）。如果您创建了自己的 VPC 和子网而不是使用默认 VPC，则可以在每个可用区中定义一个或多个子网。每个子网

都必须完全位于一个可用区之内，不能跨越多个可用区。有关更多信息，请参阅《Amazon VPC 用户指南》中的 Amazon VPC 的工作原理。

在创建自动扩缩组时，必须选择要在其中部署该自动扩缩组的 VPC 和子网。Amazon A EC2 uto Scaling 在您选择的子网中创建您的实例。因此，每个实例都与 Amazon A EC2 uto Scaling 选择的特定可用区相关联。实例启动时，Amazon A EC2 uto Scaling 会尝试在各个区域之间均匀分配实例，以实现高可用性和可靠性。

下图概括演示了跨三个可用区部署的多层级架构。



实例分配

Amazon A EC2 uto Scaling 会自动尝试在每个已启用的可用区域中保持相同数量的实例。Amazon A EC2 uto Scaling 通过尝试在可用区中启动实例最少的新实例来实现这一点。如果为可用区选择了多个子网，Amazon A EC2 uto Scaling 会尝试在可用区域中可用 IP 地址数量最多的子网中启动实例。但是，如果尝试失败，Amazon A EC2 uto Scaling 会尝试在另一个可用区启动实例，直到成功为止。

如果某个可用区运行状况不正常或不可用，实例在可用区之间的分布可能不再均匀。可用区恢复后，Amazon A EC2 uto Scaling 会自动重新平衡 Auto Scaling 组。为此，系统将会在具有最少实例的已启用可用区中启动实例，并在其他可用区中终止实例。

再平衡活动

再平衡活动分为两类：可用区再平衡和容量再平衡。

可用区再平衡

在某些操作发生后，Auto Scaling 组可能会在不同可用区之间变得不平衡。Amazon A EC2 uto Scaling 通过重新平衡可用区域来进行补偿。以下操作可能导致重新平衡活动：

- 您更改了与您的自动扩缩组关联的可用区。
- 您显式终止或分离了实例，或将实例设为待机状态，这时该组将会失衡。
- 之前没有足够容量的某个可用区已经恢复，现在具有额外的容量。
- 之前 Spot 价格超出您最高价的可用区现在的 Spot 价格低于您的最高价。

重新平衡时，Amazon A EC2 uto Scaling 会在终止之前的实例之前启动新实例。这样可确保重新平衡不会影响应用程序的性能或可用性。

由于 Amazon A EC2 uto Scaling 会尝试在终止之前启动新实例，因此达到或接近指定的最大容量可能会阻碍或完全停止再平衡活动。

为避免此问题，在再平衡活动期间，系统可以暂时超出组的指定最大容量。预设情况下，系统可以执行 10% 或一个实例的裕度，以两者中最大的为准。仅在该组达到或接近最大容量并且需要重新平衡时，才会提供边际。该超出状态仅持续重新平衡该组所需的时间（通常为几分钟）。

或者，您可以使用实例维护策略为自动扩缩组设定阈值，以便该组只能在该阈值范围内增加或减少容量。这样，您就可以控制该组自我重新平衡的速度。有关更多信息，请参阅 实例维护策略。

容量再平衡

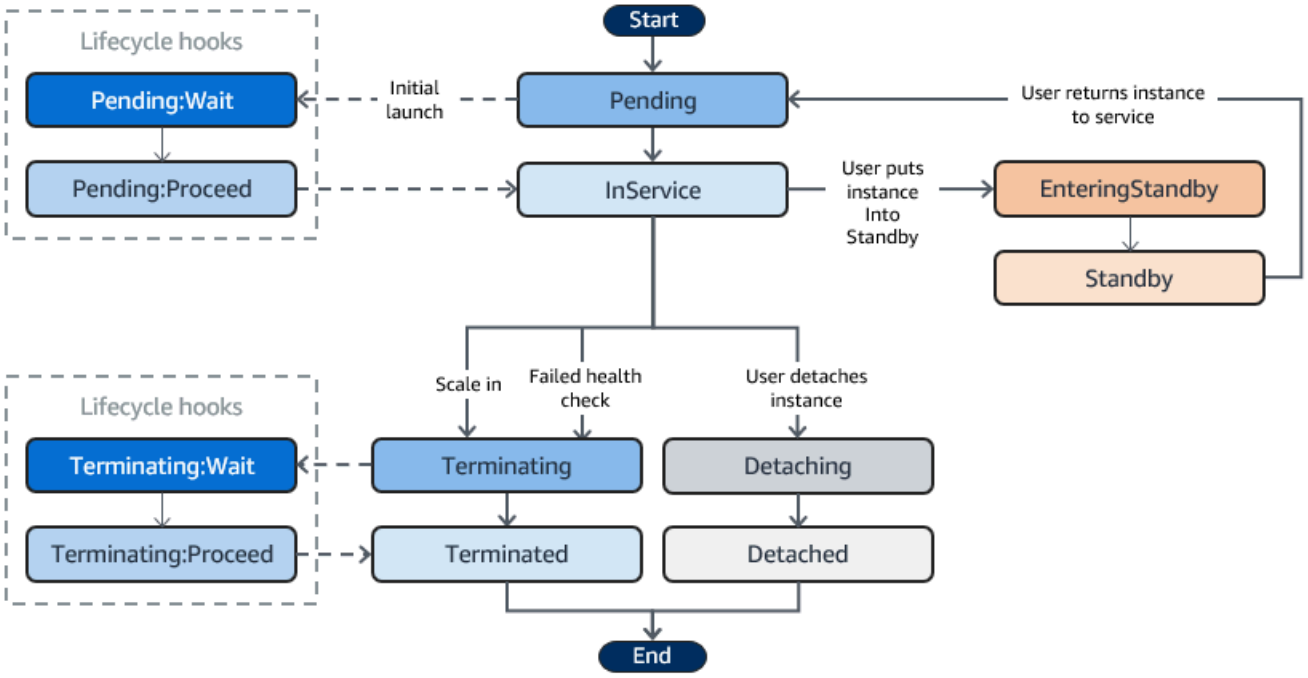
使用竞价型实例时，您可以为您的自动扩缩组开启容量再平衡。这样，当亚马逊 EC2 报告竞价型实例中断风险较高时，Amazon A EC2 uto Scaling 就可以尝试启动竞价型实例。启动新实例后，它会终止旧实例。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

Amazon A EC2 uto Scaling 实例生命周期

Auto Scaling 组中的 EC2 实例的路径或生命周期与其他 EC2 实例的路径或生命周期不同。生命周期从 Auto Scaling 组启动实例并将其投入使用时开始。生命周期在您终止实例或 Auto Scaling 组禁用实例并将其终止时结束。

> **Note**
>
> 一旦启动实例，您就需要为实例付费，包括尚未将实例投入使用的时间。

下图显示了 Amazon A EC2 uto Scaling 生命周期中实例状态之间的转换。



扩展

以下横向扩展事件指示 Auto Scaling 组启动 EC2 实例并将其附加到该组：

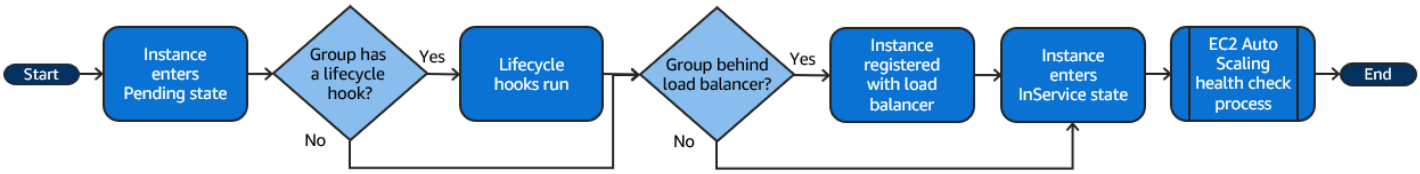
- 手动增大组的大小。有关更多信息，请参阅 更改现有自动扩缩组的所需容量。
- 您创建一个扩展策略来自动根据指定的所需增量来增大组的大小。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。
- 您可以通过安排在某个特定时间增大组的大小来设置扩展。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。

发生扩展事件时，Auto Scaling 组会使用其分配的启动模板启动所需数量的 EC2实例。这些实例最初处于 Pending 状态。如果您向 Auto Scaling 组添加生命周期挂钩，则可在此处执行自定义操作。有关更多信息，请参阅 生命周期钩子。

当每个实例都完成配置并通过 Amazon 运行 EC2 状况检查后，它就会附加到 Auto Scaling 组并进入InService状态。针对 Auto Scaling 组的所需容量对实例进行计数。

如果您的 Auto Scaling 组配置为接收来自 Elastic Load Balancing 负载均衡器的流量，则 Amazon A EC2 uto Scaling 会在将您的实例标记为之前自动向该负载均衡器注册该实例InService。

以下总结了为横向扩展事件向负载均衡器注册实例的步骤。



已投入使用的实例

实例将保持 InService 状态，直至出现下列情况之一：

- 发生了缩减事件，Amazon A EC2 uto Scaling 选择终止此实例以缩小 Auto Scaling 组的大小。有关更多信息，请参阅 控制在横向缩减过程中要终止的 Auto Scaling 实例。
- 将实例置于 Standby 状态。有关更多信息，请参阅 进入和退出备用状态。
- 您从 Auto Scaling 组分离实例。有关更多信息，请参阅 从自动扩缩组中分离或附加实例。
- 实例未通过所需数目的运行状况检查，因此将从 Auto Scaling 组中删除实例、终止实例和替换实例。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

缩小

以下缩减事件指示 Auto Scaling 组从该组中分离 EC2 实例并将其终止：

- 手动减小组的大小。有关更多信息，请参阅 更改现有自动扩缩组的所需容量。
- 您创建一个扩展策略，自动根据指定的所需减少量来减小组的大小。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。
- 您可以通过安排在某个特定时间减小组的大小来设置扩展。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。

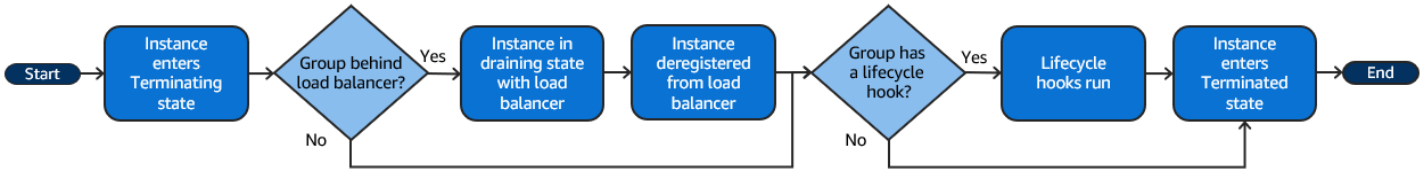
您必须为所创建的每个横向扩展事件创建一个相应的缩减事件。这有助于确保分配给您的应用程序的资源与对这些资源的需求尽可能相符。

发生缩减事件时，Auto Scaling 组会终止一个或多个实例。Auto Scaling 组使用其终止策略来确定要终止的实例。正在从自动扩缩组中终止的实例将进入 `Terminating` 状态，且无法重新将其投入使用。

如果您的 Auto Scaling 组配置为接收来自弹性负载平衡负载均衡器的流量，则 Amazon A EC2 uto Scaling 会自动从负载均衡器注销正在终止的实例。取消注册实例可确保将所有新请求重定向到负载均衡器目标组中的其他实例，同时允许与该实例的现有连接继续，直到取消注册延迟到期。

如果您向自动扩缩组添加生命周期挂钩，则可在终止中的实例上执行自定义操作。有关更多信息，请参阅 生命周期钩子。最后，实例将完全终止并进入 `Terminated` 状态。

以下总结了为横向缩减事件向负载均衡器取消注册实例的步骤。



分离实例

您可以从 Auto Scaling 组中分离实例。分离实例后，您可以独立于 Auto Scaling 组管理实例或者将实例附加到其他 Auto Scaling 组。

有关更多信息，请参阅 从自动扩缩组中分离或附加实例。

附加实例

您可以将符合特定条件的正在运行的 EC2 实例附加到您的 Auto Scaling 组。在附加实例后，将该实例作为 Auto Scaling 组的一部分进行管理。

有关更多信息，请参阅 从自动扩缩组中分离或附加实例。

生命周期钩子

您可以将生命周期挂钩添加到 Auto Scaling 组，以便在实例启动或终止时执行自定义操作。

当 Amazon A EC2 uto Scaling 响应扩展事件时，它会启动一个或多个实例。这些实例最初处于 `Pending` 状态。如果您已将一个 `autoscaling:EC2_INSTANCE_LAUNCHING` 生命周期挂钩添加到您的 Auto Scaling 组，则实例将从 `Pending` 状态转换为 `Pending:Wait` 状态。完成生命周期操作

后，实例将进入 Pending:Proceed 状态。在完全配置实例后，实例将附加到 Auto Scaling 组并进入 InService 状态。

当 Amazon A EC2 uto Scaling 响应缩减事件时，它会终止一个或多个实例。这些实例将从 Auto Scaling 组中分离并进入 Terminating 状态。如果您已将一个 autoscaling:EC2_INSTANCE_TERMINATING 生命周期挂钩添加到您的 Auto Scaling 组，则实例将从 Terminating 状态转换为 Terminating:Wait 状态。完成生命周期操作后，实例将进入 Terminating:Proceed 状态。在完全终止实例后，实例将进入 Terminated 状态。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

进入和退出备用状态

可以将任何处于 InService 状态的实例置于 Standby 状态。这使您能够终止对实例的使用，排查实例的问题或对实例进行更改，然后重新将实例投入使用。

处于 Standby 状态的实例继续由 Auto Scaling 组管理。但是，在将这些实例重新投入使用前，它们不是您的应用程序的有效部分。

有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

自动扩缩资源和组的配额

您的每项 Amazon 服务 Amazon Web Services 账户 都有默认配额，以前称为限制。除非另有说明，否则，每个限额是区域特定的。您可以请求增加某些配额，但其他一些配额无法增加。

要查看 Amazon A EC2 uto Scaling 的配额，请打开服务配额控制台。在导航窗格中，选择Amazon 服务，然后选择 Amazon A EC2 uto Scaling。

要请求提高配额，请参阅《Service Quotas 用户指南》中的请求提高配额。如果 Service Quotas 中尚未显示配额，请使用 Auto Scaling 限制表。配额的提高会与请求所针对的区域关联。

Amazon A EC2 uto Scaling 资源

您的配额 Amazon Web Services 账户 与您可以创建的 Auto Scaling 组数量和启动配置相关。

资源	默认配额
每个区域的 Auto Scaling 组	500
每个区域的启动配置	200

Auto Scaling 组配置

您 Amazon Web Services 账户 有以下与 Auto Scaling 组配置相关的配额。无法对其进行更改。

资源	限额
每个 Auto Scaling 组的扩展策略	50
每个 Auto Scaling 组的计划操作	125
每个步进扩展策略的步进调整	20
Auto Scaling 组的生命周期挂钩	50
每个 Auto Scaling 组的 SNS 主题	10
每个 Auto Scaling 组的 Classic 负载均衡器数	50
每个自动扩缩组的 Elastic Load Balancing 目标组	50
每个自动扩缩组的 VPC Lattice 目标组	5

Auto Scaling 组 API 操作

Amazon A EC2 uto Scaling 提供了 API 操作，可以批量更改您的 Auto Scaling 群组。有关单个操作中允许的最大项目数（最大数组成员数）的 API 限制如下。无法对其进行更改。

操作	最大数组成员数
AttachInstances	20 个实例 IDs
AttachLoadBalancers	10 个负载均衡器
AttachLoadBalancerTargetGroups	10 个目标组
BatchDeleteScheduledAction	50 个计划操作
BatchPutScheduledUpdateGroupAction	50 个计划操作
DetachInstances	20 个实例 IDs

操作	最大数组成员数
DetachLoadBalancers	10 个负载均衡器
DetachLoadBalancerTargetGroups	10 个目标组
EnterStandby	20 个实例 IDs
ExitStandby	20 个实例 IDs
SetInstanceProtection	50 个实例 IDs

Amazon Aut EC2 o Scaling API 的请求限制

Amazon A EC2 uto Scaling API 请求使用令牌存储桶方案进行限制，以维护服务带宽。有关更多信息，请参阅《Amazon A EC2 uto Scaling API 参考》中的 API 请求速率。

EC2 终止率

Amazon A EC2 uto Scaling 会动态确定在您的 Auto Scaling 组缩减规模时它可以执行的 EC2 实例终止操作的数量。这意味着，在自动扩缩组中，一次终止的实例数量可能会有所不同。这些差异是由外部考虑因素造成的，例如 Amazon A EC2 uto Scaling 是否必须向负载均衡器注销实例。

其他服务

其他服务（例如亚马逊 EC2 和亚马逊 VPC）的配额可能会影响您的 Auto Scaling 群组。您可以使用更新 Service Quotas 中 EC2 实例和其他资源的配额 Amazon Web Services 账户。在 Service Quotas 控制台中，您可以查看所有可用的服务配额并请求增加配额。有关更多信息，请参阅《Service Quotas 用户指南》中的申请增加限额。

有关特定于启动模板的配额，请参阅 Amazon EC2 用户指南中的启动模板限制。

设置为使用 Amazon A EC2 uto Scaling

在开始使用 Amazon A EC2 uto Scaling 之前，请完成以下任务。

任务

- 准备使用 Amazon CLI

准备使用 Amazon CLI

您可以使用 Amazon 命令行工具在系统的命令行中发出命令以执行 Amazon A EC2 uto Scaling 和其他 Amazon 任务。

要使用 Amazon Command Line Interface (Amazon CLI)，请下载、安装和配置版本 1 或 2 的 Amazon CLI。版本 1 和版本 2 中也提供了相同的 Amazon A EC2 uto Scaling 功能。要安装 Amazon CLI 版本 1，请参阅《Amazon CLI 用户指南》中的安装、更新和卸载 Amazon CLI。要安装 Amazon CLI 版本 2，请参阅版本 2 用户指南 Amazon CLI中的安装或更新最新Amazon CLI版本的。

Amazon CloudShell 允许您跳过 Amazon CLI 在开发环境中安装的步骤， Amazon Web Services Management Console 而是在中使用它。除无需安装外，您还无需配置凭证，也不需要指定区域。您的 Amazon Web Services Management Console 会话为提供了此上下文 Amazon CLI。你可以在支持 Amazon CloudShell 中使用 Amazon Web Services 区域。有关更多信息，请参阅 使用命令行创建 Auto Scaling 组 Amazon CloudShell。

有关更多信息，请参阅 Amazon CLI 命令参考中的弹性伸缩。

开始使用 Amazon A EC2 uto Scaling

要开始使用 Amazon A EC2 uto Scaling，您可以按照向您介绍该服务的教程进行操作。

主题

- 教程：创建您的第一个自动扩缩组
- 教程：设置具有扩展和负载均衡功能的应用程序

有关重点介绍管理自动扩缩组中实例生命周期的特定工具的其他教程，请参阅以下主题：

- 教程：配置调用 Lambda 函数的生命周期钩子。 本教程向您展示如何使用 Amazon EventBridge 创建规则，这些规则可根据发生在您的 Auto Scaling 组中的实例的事件来调用 Lambda 函数。
- 教程：使用数据脚本和实例元数据检索生命周期状态. 本教程演示如何使用实例元数据服务 (IMDS) 从实例本身内部调用操作。

在创建用于应用程序的自动扩缩组之前，请全面检查应用程序在 Amazon Web Services 云中运行时的情况。请考虑以下事项：

- Auto Scaling 组应跨多少个可用区。
- 可以使用哪些现有资源，例如安全组或 Amazon 系统映像 (AMIs)。
- 无论您是希望进行扩展以增加或减少容量，还是只希望确保始终运行特定数量的服务器，请记住，Amazon A EC2 uto Scaling 可以同时执行这两项操作。
- 哪些指标与应用程序的性能关系最密切。
- 启动和预置服务器需要多长时间。

您越了解您的应用程序，Auto Scaling 架构的效率就越高。

教程：创建您的第一个自动扩缩组

本教程通过以下方式亲身体验了 Amazon A EC2 uto Scaling Amazon Web Services Management Console。您将创建一个用于定义您的 EC2 实例的启动模板和一个包含单个实例的 Auto Scaling 组。启动自动扩缩组后，您将终止该实例并验证实例是否已从服务中删除并被替换。为了保持实例数量的恒定，Amazon A EC2 uto Scaling 会自动检测并响应亚马逊运行 EC2 状况和可访问性检查。

注册后 Amazon，您可以使用免费套餐Amazon 免费开始使用 Amazon A EC2 uto Scaling。您可以使用免费套餐在 12 个月内免费启动和使用 `t2.micro` 实例（在 `t2.micro` 不可用的区域，您可以使用免费套餐下的 `t3.micro` 实例）。如果您启动的实例不在免费套餐范围内，则需要为该实例支付标准的 Amazon EC2 使用费。有关更多信息，请参阅 Amazon EC2 定价。

任务

准备演练

本演练假设您熟悉启动 EC2 实例，并且已经创建了密钥对（key pair）和安全组。

要开始使用 Amazon A EC2 uto Scaling，您可以为自己使用默认 VPC Amazon Web Services 账户。默认 VPC 在各个可用区中包含一个默认公有子网，以及连接到您 VPC 的 Internet 网关。您可以在亚马逊 Virtual Priv ate Cloud（亚马逊 VPC）控制台的 “我的 VPCs页面” VPCs 上查看。

步骤 1：创建启动模板

在此步骤中，您将创建一个启动模板，该模板指定 Amazon A EC2 uto Scaling 为您创建的 EC2 实例的类型。包含一些信息，例如将使用的 Amazon Machine Image (AMI) 的 ID、实例类型、密钥对和安全组。

创建启动模板

1. 打开 Amazon EC2 控制台，进入启动模板页面。
2. 在顶部导航栏上，选择一个 Amazon Web Services 区域。您创建的启动模板和自动扩缩组将会与您所指定的区域绑定。
3. 选择Create launch template（创建启动模板）。
4. 对于 Launch template name（启动模板名称），输入 **`my-template-for-auto-scaling`**。

5. 在 Auto Scaling guidance（Auto Scaling 指导）下，选中复选框。
6. 对于 Application and OS Images (Amazon Machine Image) [应用程序和操作系统镜像（Amazon Machine Image）]，请从 Quick Start（快速启动）列表中选择一个 Amazon Linux 2（HVM）版本。AMI 用作实例的基本配置模板。
7. 对于 Instance type（实例类型），选择与您指定的 AMI 兼容的硬件配置。
8. （可选）对于 Key pair (login) [密钥对（登录）]，选择一个现有的密钥对。您可以使用密钥对通过 SSH 连接到 Amazon EC2 实例。有关连接到实例的内容不包括在本教程中。因此，除非您计划使用 SSH 连接到实例，否则不需要指定密钥对。
9. 对于 Network settings（网络设置），展开 Advanced network configuration（高级网络配置），然后执行以下操作：

   a. 选择 Add network interface（添加网络接口）以配置主网络接口。
   b. 对于自动分配公有 IP，请指定您的实例是否接收公有 IPv4 地址。默认情况下，如果实例在默认子网中启动，或者 EC2 实例启动到配置为自动分配公有 IPv4 地址的子网，Amazon EC2 会分配一个公有 IPv4 地址。如果您不需要连接到实例，则请选择禁用。
   c. 对于安全组 ID，在计划用作自动扩缩组 VPC 的同一 VPC 中选择安全组。如果您没有指定安全组，实例会自动与 VPC 的默认安全组关联。
   d. 对于终止时删除：选择是以在删除实例时删除网络接口。

10. 选择Create launch template（创建启动模板）。
11. 在确认页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

步骤 2：创建单个实例 Auto Scaling 组

创建启动模板后，使用以下步骤从中断的地方继续操作。

创建 自动扩缩组

1. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，输入 **`my-first-asg`**。
2. 选择 Next（下一步）。

   此时将显示选择实例启动选项页面，以便选择您希望自动扩缩组使用的 VPC 网络设置，并提供启动按需型实例和竞价型实例的选项。
3. 在网络部分，将 VPC 设置为您选择的默认 VPC Amazon Web Services 区域，或者选择您自己的 VPC。默认 VPC 会自动配置为向您的实例提供 Internet 连接。此 VPC 在区域的每个可用区中均包含一个公有子网。

4. 对于 Availability Zones and subnets（可用区和子网），请从您希望包含的每个可用区中选择一个子网。可以在多个可用区中使用子网以提供高可用性。有关更多信息，请参阅 选择 VPC 子网时的注意事项。
5. 在 Instance type requirements（实例类型要求）部分中，使用默认设置简化此步骤。（请勿覆盖启动模板。）在本教程中，您将仅使用启动模板中指定的实例类型启动一个按需型实例。
6. 保留本教程的其余默认值，然后选择 Skip to review（跳到审核）。

> **Note**
>
> 组的初始规模由其所需容量决定。默认值为 1 实例。

7. 在 Review（审核）页面上，查看组的信息，然后选择 Create Auto Scaling group（创建 Auto Scaling 组）。

步骤 3：验证您的 Auto Scaling 组

既然您已经创建了 Auto Scaling 组，就可以验证该组是否已启动 EC2 实例了。

> **Tip**
>
> 在以下过程中，您需要查看 Auto Scaling 组的 Activity history（活动历史记录）和 Instances（实例）部分。这两个部分都会已经显示已命名的列。要显示隐藏的列或更改显示的行数，请选择每个部分右上角的齿轮图标以打开首选项模式，根据需要更新设置，然后选择 Confirm（确认）。

验证您的 Auto Scaling 组是否已启动 EC2 实例

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中刚创建的 Auto Scaling 组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。可用的第一个选项卡是 Details（详细信息）选项卡，显示有关 Auto Scaling 组的信息。
3. 选择第二个选项卡，即 Activity（活动）。在 Activity history（活动历史记录）下，您可以查看与 Auto Scaling 组关联的活动的进度。Status（状态）列显示您实例的当前状态。当您的实例启动时，状态列将显示 `Not yet in service`。该实例启动后，状态会变为 `Successful`。您还可以使用刷新按钮来查看您的实例的当前状态。

4. 在 Instance management（实例管理）选项卡上的 Instances（实例）下，您可以查看实例的状态。
5. 验证您的实例已成功启动。启动实例只需很短的时间。

   - Lifecycle（生命周期）列显示您的实例的状态。最初，您的实例处于 Pending 状态。在实例准备好接收流量时，其状态为 InService。
   - 运行状况列显示对您的实例进行 Amazon A EC2 uto Scaling 运行状况检查的结果。

步骤 4：终止 Auto Scaling 组中的实例

使用这些步骤详细了解 Amazon A EC2 uto Scaling 的工作原理，特别是它如何在必要时启动新实例。您在本教程中创建的 Auto Scaling 组的最小大小为一个实例。因此，如果您终止了正在运行的实例，Amazon A EC2 uto Scaling 必须启动一个新实例来替换它。

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。
3. 在 Instance management（实例管理）选项卡上的 Instances（实例）下，选择实例的 ID。

   这会将您带到 Amazon EC2 控制台的 “实例” 页面，您可以在其中终止实例。
4. 依次选择 Actions（操作）、Instance State（实例状态）和 Terminate（终止）。当系统提示您确认时，选择 Yes, Terminate（是，终止）。
5. 在导航窗格的 Auto Scaling 下，选择 自动扩缩组。选择您的 Auto Scaling 组，然后选择 Activity（活动）选项卡。

   当您从实例页面终止实例时，在终止该实例后需要一两分钟的时间才能启动新实例。在活动历史记录中，当扩展活动启动时，将会看到有关终止第一个实例的条目以及有关启动新实例的条目。使用刷新按钮直到看到新条目。
6. 在 Instance management（实例管理）选项卡上，Instances（实例）部分仅显示新实例。
7. 在导航窗格上的 Instances（实例）下，选择 Instances（实例）。此页面同时显示已终止的实例和新的正在运行的实例。

步骤 5：后续步骤

如果要删除您刚刚创建的基本基础设施，请转到下一步。否则，您可以将该基础设施作为基础，然后尝试以下一个或多个操作：

- 使用会话管理器或 SSH 连接到 Linux 实例。有关更多信息，请参阅 Amazon EC2 用户指南中的使用会话管理器连接到您的实例和使用 SSH 连接您的 Linux 实例。 EC2
- 配置 Amazon SNS 通知，以便在您的自动扩缩组启动或终止实例时收到通知。有关更多信息，请参阅 Amazon SNS 通知选项。
- 手动扩展您的自动扩缩组以测试 SNS 通知。有关更多信息，请参阅 更改您自动扩缩组的所需容量。

您也可以通过阅读关于 目标跟踪扩展策略 的内容开始熟悉自动扩缩概念。如果应用程序的负载发生变化，自动扩缩组可以通过在最小和最大容量限制之间调整组的所需容量来自动横向扩展（添加实例）和横向缩减（运行更少的实例）。有关这些限制的更多信息，请参阅 为自动扩缩组设置扩缩限制。

步骤 6：清除

您可以删除扩缩基础设施，或者仅删除自动扩缩组而保留启动模板以供将来使用。

如果您启动的实例不在 Amazon 免费套餐范围内，则应终止实例以避免产生额外费用。当您终止实例时，与其关联的数据也将被删除。

删除 Auto Scaling 组

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组 (`my-first-asg`) 旁边的复选框。
3. 选择删除。
4. 当系统提示进行确认时，键入 **`delete`** 以确认删除指定自动扩缩组，然后选择 Delete（删除）。

   名称列中的加载图标指示 Auto Scaling 组正在被删除。发生删除时，Desired（所需）、Min（最小）和 Max（最大）列显示 Auto Scaling 组具有 `0` 个实例。终止实例并删除组需要几分钟时间。刷新列表以查看当前状态。

如果要保留启动模板，请跳过以下过程。

删除启动模板

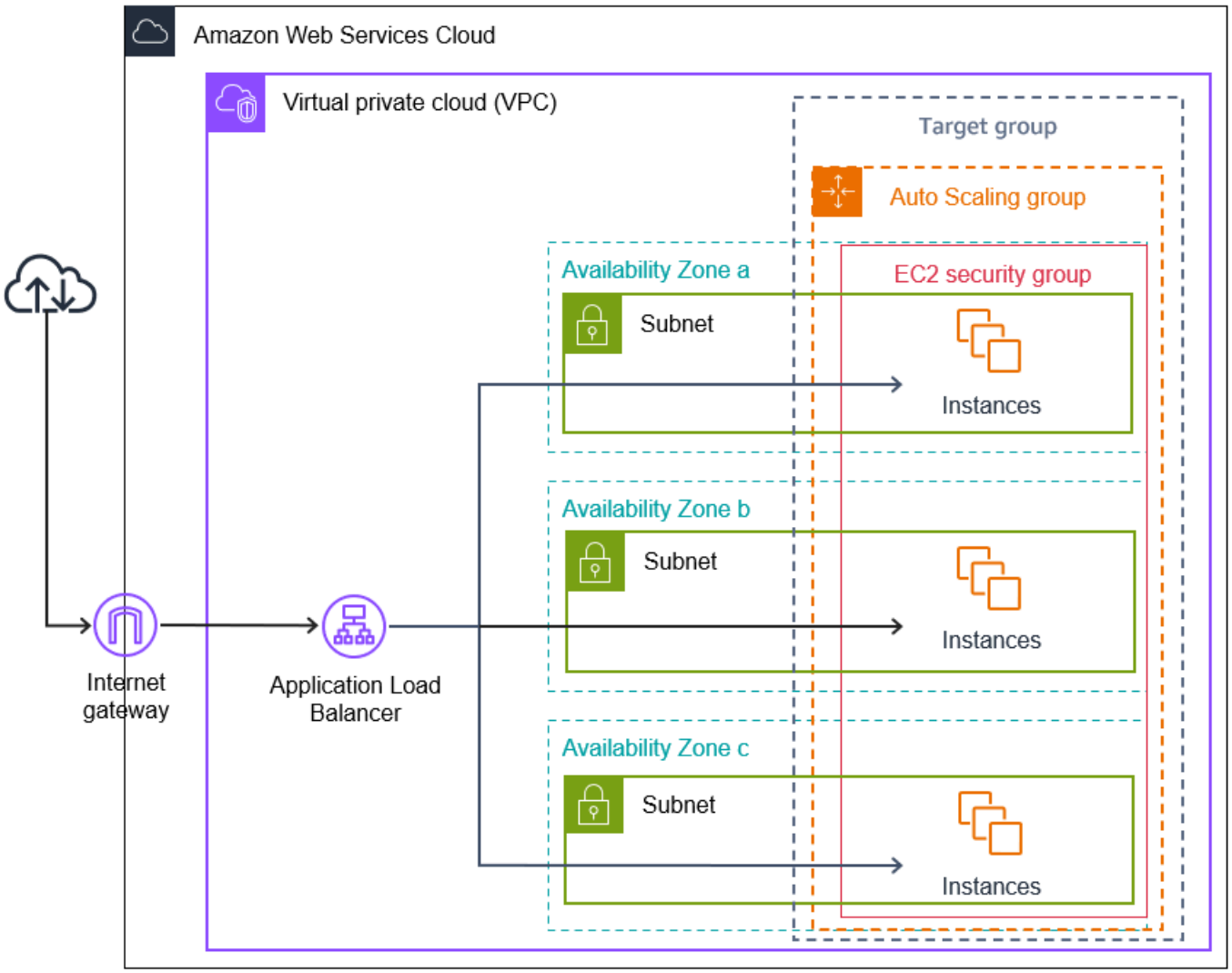
1. 打开 Amazon EC2 控制台的启动模板页面。
2. 选择您的启动模板 (`my-template-for-auto-scaling`)。
3. 选择 Actions（操作），然后选择 Delete template（删除模板）。
4. 当系统提示进行确认时，键入 **`Delete`** 以确认删除指定启动模板，然后选择 Delete（删除）。

教程：设置具有扩展和负载均衡功能的应用程序

> **Important**
>
> 在开始学习本教程之前，建议您首先阅读以下介绍性教程：创建您的第一个自动扩缩组。

使用 Elastic Load Balancing 负载均衡器注册 Auto Scaling 组可帮助您设置具有负载均衡功能的应用程序。Elastic Load Balancing 与 Amazon A EC2 uto Scaling 配合使用，在健康的亚马逊 EC2实例中分配传入流量。这将提高应用程序的可扩展性和可用性。您可以在多个可用区内启用 Elastic Load Balancing 来提高应用程序的容错能力。

在本教程中，我们介绍了在创建 Auto Scaling 组时设置负载均衡的应用程序的基本步骤。完成后，您的架构看起来应当如下图所示：



Elastic Load Balancing 支持三类负载均衡器。我们建议您在本教程中使用 Application Load Balancer。

有关在架构中引入负载均衡器的详细信息，请参阅 使用 Elastic Load Balancing 在 Auto Scaling 组中分配传入的应用程序流量 。

任务

先决条件

- 负载均衡器和目标组。确保针对计划为 Auto Scaling 组使用的负载均衡器选择了相同的可用区。有关更多信息，请参阅 Elastic Load Balancing 用户指南中的 Elastic Load Balancing 入门。
- 启动模板或启动配置的安全组。安全组必须允许从侦听器端口（通常为 HTTP 流量的端口 80）和您希望 Elastic Load Balancing 用于运行状况检查的端口上的负载均衡器进行访问。有关更多信息，请参阅相应文档：
  - Application Load Balancer 用户指南中的目标安全组
  - 《Network Load Balancer 用户指南》中的目标安全组

  如果您的实例将拥有公有 IP 地址，您可以选择允许 SSH 流量连接到这些实例。
- （可选）一个 IAM 角色，用于向您的应用程序授予访问权限 Amazon。
- （可选）定义为亚马逊 EC2实例源模板的亚马逊系统映像 (AMI)。要立即创建一个上述项，请启动一个实例。将 IAM 角色（如果已创建）和所需的任何配置脚本指定为用户数据。连接到实例并对其进行自定义。例如，您可以安装软件和应用程序、复制数据和连接更多的 EBS 卷。测试您的实例上的应用程序以确保实例配置正确。将此更新的配置另存为自定义 AMI。如果您以后不需要该实例，您可以终止它。从该新自定义 AMI 启动的实例包括您在创建 AMI 时设置的自定义项。
- 虚拟私有云 (VPC)。本教程引用默认 VPC，但您可以使用自己的 VPC。如果使用您自己的 VPC，请确保它拥有映射到您工作时所在区域的每个可用区的子网。您至少必须具有两个公有子网，且这些子

网可用于创建负载均衡器。您还必须具有两个私有子网或两个公有子网，以创建 Auto Scaling 组并使用负载均衡器注册。

步骤 1：设置启动模板或启动配置

对本教程使用启动模板或启动配置。

主题

选择或创建启动模板

如果您已拥有要使用的启动模板，请使用以下过程选择该启动模板。

选择现有启动模板

1. 打开 Amazon EC2 控制台的启动模板页面。
2. 在屏幕顶部的导航栏上，选择在其中创建了负载均衡器的区域。
3. 选择启动模板。
4. 选择 Actions (操作)、Create Auto Scaling group (创建 Auto Scaling 组)。

或者，使用下列过程创建新的启动模板。

创建启动模板

1. 打开 Amazon EC2 控制台的启动模板页面。
2. 在屏幕顶部的导航栏上，选择在其中创建了负载均衡器的区域。
3. 选择 Create launch template (创建启动模板)。
4. 为启动模板的初始版本输入名称并提供描述。
5. 对于 Application and OS Images (Amazon Machine Image) [应用程序和操作系统镜像（Amazon Machine Image）]，选择实例 AMI 的 ID。您可以搜索所有可用的 AMI AMIs，也可以从 “最近” 或 “快速入门” 列表中选择 AMI。如果您没有看到所需的 AMI，请选择 “浏览更多” AMIs 以浏览完整的 AMI 目录。
6. 对于 Instance type (实例类型)，选择与您指定的 AMI 兼容的实例硬件配置。

7. （可选）对于 Key pair (login) (密钥对（登录）)，请输入在连接到您的实例时使用的密钥对。
8. 对于 Network settings（网络设置），展开 Advanced network configuration（高级网络配置），然后执行以下操作：
   a. 选择 Add network interface（添加网络接口）以配置主网络接口。
   b. 对于自动分配公有 IP，请指定您的实例是否接收公有 IPv4 地址。默认情况下，如果实例在默认子网中启动，或者 EC2 实例启动到配置为自动分配公有 IPv4 地址的子网，Amazon EC2 会分配一个公有 IPv4 地址。如果不需要连接到您的实例，可以选择禁用以防止组中的实例直接从互联网接收流量。在这种情况下，它们将仅从负载均衡器接收流量。
   c. 对于 Security group ID (安全组 ID)，从与负载均衡器相同的 VPC 中为您的实例指定一个安全组。
   d. 对于 Delete on termination (终止时删除)，请选择 Yes (是)。这将在 Auto Scaling 组缩减并终止网络接口附加到的实例时删除网络接口。
9. （可选）要将凭证安全地分配到实例，对于 Advanced details (高级详细信息) 和 IAM instance profile (IAM 实例配置文件)，输入 IAM 角色的 Amazon Resource Name (ARN)。
10. (可选) 要为实例指定用户数据或配置脚本，请将其粘贴到高级详细信息和用户数据。
11. 选择Create launch template（创建启动模板）。
12. 在确认页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

创建或选择启动配置

> **Note**
>
> 强烈建议不要在新应用程序中使用启动配置，因为这是一项没有计划投资的旧版功能。此外，2023 年 6 月 1 日及之后创建的新账户都无法选择通过控制台创建新的启动配置。有关更多信息，请参阅 自动扩缩启动配置。

选择现有的启动配置

1. 打开 Amazon EC2 控制台的启动配置页面。
2. 在顶部的导航栏上，选择在其中创建了负载均衡器的区域。
3. 选择启动配置。
4. 选择操作、创建 Auto Scaling 组。

或者，要创建新的启动配置，请使用以下过程：

创建启动配置

1. 打开 Amazon EC2 控制台的启动配置页面。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
2. 在顶部的导航栏上，选择在其中创建了负载均衡器的区域。
3. 选择创建启动配置，然后为您的启动配置输入名称。
4. 对于 Amazon Machine Image (AMI)，请输入您的实例的 AMI 的 ID 作为搜索条件。
5. 对于实例类型，为您的实例选择硬件配置。
6. 在其他配置下，请注意以下字段：

   a. （可选）要安全地向您的 EC2 实例分发证书，对于 IAM 实例配置文件，请选择您的 IAM 角色。有关更多信息，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。

   b. （可选）要为实例指定用户数据或配置脚本，请将其粘贴到高级详细信息和用户数据。

   c. （可选）对于高级详细信息、IP 地址类型，保留默认值。创建 Auto Scaling 组时，您可以使用已启用公有 IP 寻址属性的子网（如默认 VPC 中的默认子网）为 Auto Scaling 组中的实例分配公有 IP 地址。或者，如果不需要连接到您的实例，可以选择 Do not assign a public IP address to any instances (不为任何实例分配公有 IP 地址）以防止组中的实例直接从互联网接收流量。在这种情况下，它们将仅从负载均衡器接收流量。
7. 对于安全组，从与负载均衡器相同的 VPC 中选择现有安全组。如果您保持选中创建新安全组选项，则会为运行 Linux 的 Amazon EC2 实例配置默认 SSH 规则。为运行 Windows 的亚马逊 EC2 实例配置了默认 RDP 规则。
8. 对于密钥对（登录），请选择密钥对选项下的选项。

   如果您已经配置了 Amazon EC2 实例密钥对，则可以在此处进行选择。

   如果您还没有 Amazon EC2 实例密钥对，请选择创建新的密钥对并为其指定一个可识别的名称。选择下载密钥对以将密钥对下载到您的计算机。

   > **Important**
   >
   > 如果需要连接到您的实例，不要选择在没有密钥对的情况下继续。

9. 选中确认复选框，然后选择 Create launch configuration（创建启动配置）。
10. 选中新启动配置名称旁边的复选框，然后选择操作、创建 Auto Scaling 组。

步骤 2：创建 Auto Scaling 组。

创建或选择启动模板或启动配置后，使用以下过程从中断的地方继续操作。

创建 Auto Scaling 组

1. 在 Choose launch template or configuration（选择启动模板或配置）页面上，对于 Auto Scaling group name（Auto Scaling 组名称），输入 Auto Scaling 组的名称。
2. [仅启动模板] 对于 Launch template（启动模板），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。
3. 选择 Next（下一步）。

   此时将显示页面 Choose instance launch options（选择实例启动选项），以便选择您希望 Auto Scaling 组使用的 VPC 网络设置，并提供启动按需型实例和竞价型实例的选项（如果您选择了启动模板）。
4. 在 Network（网络）部分中，对于 VPC，选择您用于负载均衡器的 VPC。如果您选择了默认 VPC，则它会自动配置为向您的实例提供 Internet 连接。此 VPC 在区域的每个可用区中均包含一个公有子网。
5. 对于 Availability Zones and subnets（可用区和子网），根据负载均衡器所在的可用区从要包含的每个可用区中选择一个或多个子网。有关更多信息，请参阅 选择 VPC 子网时的注意事项。
6. [仅限启动模板] 在 Instance type requirements（实例类型要求）部分中，使用默认设置简化此步骤。（请勿覆盖启动模板。）在本教程中，您将仅使用启动模板中指定的实例类型启动按需实例。
7. 选择 Next（下一步）转至 Configure advanced options（配置高级选项）页面。
8. 要将组连接到现有的负载均衡器，请在 Load balancing (负载均衡) 部分中选择 Attach to an existing load balancer (连接到现有负载均衡器)。您可以选择 Choose from your load balancer target groups (从负载均衡器目标组中进行选择) 或 Choose from Classic Load Balancers (从经典负载均衡器中选择)。然后，您可以为创建的 Application Load Balancer 或 Network Load Balancer 选择目标组的名称，或者选择经典负载均衡器的名称。
9. （可选）对于运行状况检查、其他运行状况检查类型，请选择启用 Elastic Load Balancing 运行状况检查。
10. （可选）对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
11. 完成 Auto Scaling 组的配置后，选择 Skip to review（跳过以审核）。

12. 在 Review（审核）页面上，审核 Auto Scaling 组的详细信息。您可以选择 Edit（编辑）进行更改。在完成后，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

在创建附加了负载均衡器的 Auto Scaling 组后，负载均衡器会在新实例联机时自动注册这些实例。此时，您只有一个实例，因此要注册的内容并不多。不过，您可以通过更新组的所需容量来添加其他实例。有关 step-by-step说明，请参阅更改您自动扩缩组的所需容量。

步骤 3：验证是否已附加您的负载均衡器

验证是否已附加您的负载均衡器

1. 在亚马逊 EC2 控制台的 A uto Scaling 群组页面上，选中 Auto Scaling 群组旁边的复选框。
2. 在详细信息选项卡上，负载均衡将显示任何附加的负载均衡器目标组或经典负载均衡器。
3. 在 Activity（活动）选项卡上，在Activity history（活动历史记录）中，您可以验证您的实例是否已成功启动。Status（状态）列显示 Auto Scaling 组是否具有已成功启动的实例。如果您的实例无法启动，您可以在 对 Amazon A EC2 uto Scaling 中的问题进行故障排除 中找到常见实例启动问题的故障排除思路。
4. 在 Instance management（实例管理）选项卡上的 Instances（实例）下，可以验证您的实例是否准备好接收流量。最初，您的实例处于 `Pending` 状态。在实例准备好接收流量时，其状态为 `InService`。“运行状况” 列显示对您的实例进行 Amazon A EC2 uto Scaling 运行状况检查的结果。尽管实例可能标记为运行状况良好，但负载均衡器只会向通过负载均衡器运行状况检查的实例发送流量。
5. 验证您已向负载均衡器注册您的实例。打开 Amazon EC2 控制台的 “目标群组” 页面。选择您的目标组，然后选择 Targets（目标）选项卡。如果实例的状态为 `initial`，这可能是因为它们仍在注册过程中，或者它们仍在进行运行状况检查。当实例状态为 `healthy` 时，即可供使用。

步骤 4：后续步骤

现在您已完成本教程，您可以了解更多信息：

- Amazon A EC2 uto Scaling 根据您的 Auto Scaling 组使用的运行状况检查的状态来确定实例是否运行正常。如果您启用负载均衡器运行状况检查，并且实例未通过运行状况检查，自动扩缩组即认为该实例运行状况不正常并进行替换。有关更多信息，请参阅 运行状况检查。
- 可以将应用程序扩展到同一区域中的其他可用区，来提高服务中断时的容错能力。有关更多信息，请参阅 添加可用区。

- 您可以将 Auto Scaling 组配置为使用目标跟踪扩展策略。这会在实例需求变化时自动增加或减少实例数量。这将允许该组处理应用程序接收的流量的变化。有关更多信息，请参阅 目标跟踪扩展策略。

第 5 步：清理

完成为本教程创建的资源后，应考虑清除这些资源，以免产生不必要的费用。

要删除 Auto Scaling 组

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。
3. 选择删除。
4. 当系统提示进行确认时，键入 **delete** 以确认删除指定自动扩缩组，然后选择 Delete（删除）。

   名称列中的加载图标指示 Auto Scaling 组正在被删除。发生删除时，Desired（所需）、Min（最小）和 Max（最大）列显示 Auto Scaling 组具有 0 个实例。终止实例并删除组需要几分钟时间。刷新列表以查看当前状态。

如果要保留启动模板，请跳过以下过程。

删除启动模板

1. 打开 Amazon EC2 控制台的启动模板页面。
2. 选择启动模板。
3. 选择 Actions（操作），然后选择 Delete template（删除模板）。
4. 当系统提示进行确认时，键入 **Delete** 以确认删除指定启动模板，然后选择 Delete（删除）。

如果您要保留启动配置，请跳过以下过程。

删除启动配置

1. 打开 Amazon EC2 控制台的启动配置页面。
2. 选择启动配置。
3. 依次选择 Actions（删除）和 Delete launch configuration（删除启动配置）。
4. 当系统提示进行确认时，选择 Delete（删除）。

如果您要保留负载均衡器供将来使用，请跳过以下步骤。

删除您的负载均衡器

1. 打开 Amazon EC2 控制台的负载均衡器页面。
2. 选择负载均衡器，然后依次选择 Actions（操作）和 Delete（删除）。
3. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

要删除目标组

1. 打开 Amazon EC2 控制台的 “目标群组” 页面。
2. 选择目标组，然后依次选择 Actions（操作）、Delete（删除）。
3. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

相关资源

借 Amazon CloudFormation助，您可以使用模板文件将资源集合作为一个单元（堆栈）一起创建和删除，从而可预测且重复地创建和配置 Amazon 基础架构部署。有关更多信息，请参阅 用户指南。Amazon CloudFormation

有关使用堆栈模板预配自动扩缩组和应用程序负载均衡器的演练，请参阅《Amazon CloudFormation 用户指南》中的演练：创建经扩展和负载均衡的应用程序。使用这些演练和示例模板作为起点来创建类似的模板以满足您的需求。

Auto Scaling 启动模板

启动模板类似于启动配置，因为它指定实例配置信息。它包括 Amazon 系统映像 (AMI) 的 ID、实例类型、密钥对、安全组和其他用于启动 EC2 实例的参数。但是，定义启动模板而非启动配置可让您有多个版本的启动模板。

利用启动模板的版本控制，您可以创建全套参数的子集。然后，您可以重复使用它来创建同一启动模板的其他版本。例如，您可以创建一个启动模板，用于定义无 AMI 或用户数据脚本的基本配置。创建启动模板后，您可以创建新版本并添加具有最新版本的应用程序的 AMI 和用户数据进行测试。这将生成两个版本的启动模板。存储基本配置可帮助您保持所需的常规配置参数。您可以随时根据基本配置创建新版本的启动模板。不再需要时，您也可以删除用于测试应用程序的版本。

我们建议您使用启动模板以确保您可以访问最新功能和改进。当您使用启动配置时，并非所有 Amazon A EC2 uto Scaling 功能都可用。例如，您无法创建 Auto Scaling 组来同时启动竞价型实例和按需型实例或者指定多个实例类型。您必须使用启动模板来配置这些功能。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

借助启动模板，您还可以使用 Amazon 的新功能 EC2。这包括 Systems Manager 参数 (AMI ID)、当前一代的 EBS 预配置 IOPS 卷 (io2)、EBS 卷标记、T2 Unlimited 实例、容量预留、Capacity Blocks，以及专用主机，仅举几例。

创建启动模板时，所有参数都是可选的。但是，如果启动模板未指定 AMI，则无法在创建您的 Auto Scaling 组时添加 AMI。如果您指定 AMI 但没有实例类型，则可以在创建您的 Auto Scaling 组时添加一个或多个实例类型。

内容

使用启动模板的权限

本节中的过程假定您已具有创建启动模板所需的权限。有关管理员如何向您授予权限的信息，请参阅 Amazon EC2 用户指南中的使用 IAM 权限控制启动模板的权限。

请注意，如果您没有足够的权限使用和创建启动模板中指定的资源，则当您尝试为自动扩缩组指定启动模板时，您会收到一条错误，指出您未获授权使用该启动模板。有关更多信息，请参阅 对 Amazon A EC2 uto Scaling 进行故障排除：启动模板。

有关允许您使用启动模板调用 `CreateAutoScalingGroup`、`UpdateAutoScalingGroup` 和 `RunInstances` API 操作的 IAM 策略示例，请参阅在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况。

启动模板支持的 API 操作

有关启动模板支持的 API 操作列表，请参阅《亚马逊 EC2 API 参考》中的亚马逊 EC2 操作。

为 Auto Scaling 组创建启动模板

在可以使用启动模板创建自动扩缩组之前，您必须创建启动模板，其中包含用于启动实例的配置信息，包括Amazon Machine Images (AMI)的 ID。

要创建新的启动模板，请使用以下过程。

内容

> **Important**
>
> 在创建启动模板时，不会完全验证启动模板参数。如果您为参数指定了错误的值，或者如果您未使用受支持的参数组合，则任何实例都无法通过此启动模板启动。确保为参数指定的值正

确，并使用支持的参数组合。例如，要使用基于 Arm 的 Amazon Graviton 或 Graviton2 AMI 启动实例，您必须指定一个与 Arm 兼容的实例类型。有关更多信息，请参阅 Amazon EC2 用户指南中的启动模板限制。

创建启动模板（控制台）

以下步骤说明了配置基本启动模板的过程：

- 指定要从其启动实例的 Amazon machine image（AMI）。
- 选择一个与您指定的 AMI 兼容的实例类型。
- 指定连接到实例时要使用的密钥对，例如使用 SSH。
- 添加一个或多个安全组以允许对实例的网络访问。
- 指定是否要为每个实例挂载附加卷。
- 将自定义标签（键值对）添加到实例和卷。

创建启动模板

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的实例下，选择启动模板。
3. 选择 Create launch template (创建启动模板)。为启动模板的初始版本输入名称并提供描述。
4. （可选）在 A uto Scaling 指导下，选中复选框，让亚马逊 EC2 提供指导，帮助创建用于 Amazon A EC2 uto Scaling 的模板。
5. 在 Launch template contents（启动模板内容）下，填写每个必填字段以及所有可选字段。

   a. Application and OS Images (Amazon Machine Image) [应用程序和操作系统镜像（Amazon Machine Image）]：（必填项）选择实例 AMI 的 ID。您可以搜索所有可用的 AMI AMIs，也可以从 “最近” 或 “快速入门” 列表中选择 AMI。如果您没有看到所需的 AMI，请选择 “浏览更多” AMIs 以浏览完整的 AMI 目录。

   要选择自定义 AMI，您必须首先从某个自定义实例创建该 AMI。有关更多信息，请参阅亚马逊用户指南中的创建由亚马逊 EBS 支持的 AMI。 EC2

   b. 对于 Instance type（实例类型），请选择与您指定的 AMI 兼容的单个实例类型。

   或者，要使用基于属性的实例类型选择，请选择高级、指定实例类型属性，然后指定以下选项：

- v 数 CPUs：输入 v 的最小和最大数CPUs。 要表示没有限制，请输入最小值 0，并将最大值保留为空。
- Amount of memory (MiB) [内存大小（MiB）]：输入最小和最大内存大小（以 MiB 为单位）。若要表示为无限制，请输入最小值为 0，然后将最大值留空。
- 展开 Optional instance type attributes（可选的实例类型属性），然后选择 Add attribute（添加属性）以进一步限制可用于满足所需容量的实例类型。有关每个属性的信息，请参阅 Amazon EC2 API 参考InstanceRequirementsRequest中的。
- 生成的实例类型：您可以查看符合指定计算要求的实例类型，例如 v CPUs、内存和存储。
- 要排除实例类型，请选择 Add Attribute（添加属性）。从 Attribute（属性）列表中，选择 Excluded instance types（排除的实例类型）。从 Attribute value（属性值）列表中，选择要排除的实例类型。

c. Key pair (login) [密钥对（登录）]：对于 Key pair name（密钥对名称），请选择一个现有密钥对，或选择 Create new key pair（创建新密钥对）以新建一个密钥对。有关更多信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 密钥对和 Linux 实例。

d. Network settings（网络设置）：对于 Firewall (security groups) [防火墙（安全组）]，请使用一个或多个安全组，或将此留空并将一个或多个安全组配置为网络接口的一部分。有关更多信息，请参阅亚马逊 EC2 用户指南中的适用于 Linux 实例的亚马逊 EC2安全组。

   如果您未在启动模板中指定任何安全组，Amazon 将 EC2 使用您的 Auto Scaling 组将启动实例的 VPC 的默认安全组。预设情况下，此安全组不允许来自外部网络的入站流量。有关更多信息，请参阅 Amazon VPC 用户指南 VPCs中的您的默认安全组。

e. 请执行以下操作之一：

   - 更改原定设置网络接口设置。例如，您可以启用或禁用公有 IPv4寻址功能，该功能会覆盖子网上的自动分配公有 IPv4 地址设置。有关更多信息，请参阅 更改默认网络接口设置（控制台）。
   - 跳过此步骤以保留原定设置网络接口设置。

f. 请执行以下操作之一：

   - 修改存储配置。有关更多信息，请参阅 修改存储配置（控制台）。
   - 跳过此步骤以保留原定设置存储配置。

g. 对于 Resource tags（资源标签），请提供键值组合以指定标签。如果您在启动模板中指定了实例标签，然后选择将 Auto Scaling 组的标签传播到其实例，则所有标签都会合并。如果为启动模板中的标签和 Auto Scaling 组中的标签指定了相同的标签键，则优先使用该组中的标签值。

6. （可选）配置高级设置。例如，您可以选择一个 IAM 角色，以供您的应用程序在访问其他 Amazon 资源或指定实例启动后可用于执行常见自动配置任务的实例用户数据。有关更多信息，请参阅 使用高级设置创建启动模板。
7. 准备好创建启动模板后，请选择 Create launch template（创建启动模板）。
8. 要创建 Auto Scaling 组，请从确认页面上选择创建 Auto Scaling 组。

更改默认网络接口设置（控制台）

网络接口提供与您的 VPC 中其他资源和互联网的连接。有关更多信息，请参阅 使用 Amazon VPC 为 Auto Scaling 实例提供网络连接。

这一部分说明了如何更改原定设置网络接口设置。例如，您可以定义是否要为每个实例分配公有 IPv4 地址，而不是默认使用子网上的自动分配公有 IPv4 地址设置。

注意事项和限制

更改原定设置网络接口设置时，请记住以下注意事项和限制：

- 您必须将安全组配置为该网络接口的一部分，而不是在模板的 Security groups（安全组）部分中配置。您不能在这两处指定安全组。
- 如果指定现有的网络接口 ID，则只能启动一个实例。为此，您必须使用 Amazon CLI 或 SDK 创建 Auto Scaling 组。创建组时，必须指定可用区，但不指定子网 ID。此外，仅当现有网络接口的设备索引为 0 时才可指定该接口。
- 如果您指定多个网络接口，则无法自动分配公共 IPv4 地址。也无法跨网络接口指定重复的设备索引。主网络接口和辅助网络接口都驻留在同一子网中。
- 实例启动时，系统会自动为每个网络接口分配一个私有地址。该地址来自启动实例的子网的 CIDR 范围。有关为 VPC 或子网指定 CIDR 范围（或 IP 地址范围）的信息，请参阅 Amazon VPC 用户指南。

更改原定设置网络接口设置

1. 在 Network settings（网络设置）下，展开 Advanced network configuration（高级网络配置）。
2. 选择 Add network interface（添加网络接口）以配置主网络接口，同时应注意以下字段：

   a. Device index（设备索引）：保留原定设置值 0 以将您的更改应用于主网络接口（eth0）。

b. 网络接口：保留默认值“新接口”，让 Amazon A EC2 uto Scaling 在实例启动时自动创建新的网络接口。您也可以选择一个设备索引为 0 的现有可用网络接口，但这将您的 Auto Scaling 组限定为单个实例。

c. Description（描述）：（可选）请输入一个描述性的名称。

d. Subnet（子网）：保留原定设置 Don't include in launch template（不包括在启动模板中）。

   如果 AMI 指定了子网的网络接口，则会导致错误。我们建议关闭 Auto Scaling guidance（Auto Scaling 指南）以解决此问题。进行此更改后，您将不会收到错误消息。但无论指定的子网来自何处，Auto Scaling 组的子网设置将会优先，且不能被覆盖。

e. 自动分配公有 IP：更改设备索引为 0 的网络接口是否接收公共 IPv4地址。默认情况下，默认子网中的实例会收到公有 IPv4 地址，而非默认子网中的实例则不会。选择启用或禁用可以覆盖子网的默认设置。

f. Security groups（安全组）：为网络接口选择一个或多个安全组。对于 Auto Scaling 组会将实例启动到其中的 VPC，必须为其配置各个安全组。有关更多信息，请参阅亚马逊 EC2 用户指南中的适用于 Linux 实例的亚马逊 EC2安全组。

g. Delete on termination（终止时删除）：选择 Yes（是）以在实例终止时删除网络接口，或选择 No（否）以保留网络接口。

h. Elastic Fabric Adapter：要支持高性能计算和机器学习使用案例，请将网络接口更改为某个 Elastic Fabric Adapter 网络接口。有关更多信息，请参阅 Amazon EC2 用户指南中的弹性结构适配器。

i. Network card index（网卡索引）：选择 0 以将主要网络接口挂载到设备索引为 0 的网卡。如果此选项不可用，则保留原定设置 Don't include in launch template（不包括在启动模板中）。将网络接口挂载到特定网卡的功能仅适用于支持的实例类型。有关更多信息，请参阅 Amazon EC2 用户指南中的网卡。

j. ENA Express：对于支持 ENA Express 的实例类型，您可以选择启用以启用 ENA Express，或者选择禁用将其禁用。有关更多信息，请参阅亚马逊 EC2 用户指南中的在 Linux 实例上使用 ENA Express 提高网络性能。

k. ENA Express UDP：如果您启用 ENA Express，则可以选择将其用于 UDP 流量。选择启用以启用 ENA Express UDP，或者选择禁用将其禁用。

3. 要添加辅助网络接口，请选择添加网络接口。

修改存储配置（控制台）

您可以修改从由 Amazon EBS-backed AMI 或实例存储支持的 AMI 启动的实例的存储配置。您还可以指定要挂载到实例的附加 EBS 卷。AMI 会包含一个或多个存储卷，包括根卷 [Volume 1 (AMI Root)] [卷 1（AMI 根）]。

修改存储配置

1. 在 Configure storage（配置存储）中，修改卷的大小或类型。

   如果指定的卷大小值超出相应卷类型的限制，或者小于快照大小，则会显示错误消息。此消息会提供有关该字段可以接受的最小值或最大值信息，以帮助您解决此问题。

   这时仅会显示与由 Amazon EBS-backed AMI 关联的卷。要显示从实例存储支持的 AMI 启动的实例的存储配置信息，请选择 Instance store volumes（实例存储卷）部分的 Show details（显示详细信息）。

   要指定所有 EBS 卷参数，请切换到右上角的 Advanced（高级）视图。
2. 对于高级选项，请展开要修改的卷并按如下方式配置该卷：

   a. Storage type（存储类型）：要与实例关联的卷类型（EBS 或临时卷）。仅当您选择支持实例存储（临时）卷类型的实例类型时，该卷类型才可用。有关更多信息，请参阅亚马逊 EBS 用户指南中的亚马逊 EBS 卷和亚马逊用户指南中的亚马逊 EC2 EC2 实例存储。

   b. Device name（设备名称）：从卷的可用设备名称列表中进行选择。

   c. Snapshot（快照）：选择要从其中创建卷的快照。您可以通过在 Snapshot（快照）字段中输入文本来搜索可用的共享快照和公有快照。

   d. Size (GiB)（大小 (GiB)）：对于 EBS 卷，您可以指定存储大小。如果您选择了有资格享用免费套餐的 AMI 和实例，请记住，若要享用免费套餐，您必须将总存储大小保持为 30GiB 以下。有关更多信息，请参阅《Amazon EBS User Guide》中的 Constraints on the size and configuration of an EBS volume。

   e. Volume type（卷类型）：对于 EBS 卷，请选择该卷类型。有关更多信息，请参阅《Amazon EBS 用户指南》中的 Amazon EBS 卷类型。

   f. IOPS：如果您已选择预置 IOPS SSD（io1 和 io2）以及通用型 SSD（gp3）卷类型，则您可以输入卷可支持的每秒输入/输出操作数（IOPS）。这对于 io1、io2 和 gp3 卷是必需的。gp2、st1、sc1 或标准卷不支持。

   g. Delete on termination（终止时删除）：对于 EBS 卷，选择 Yes（是）以在终止实例时删除此卷或选择 No（否）以保留此卷。

h. Encrypted（加密）：如果实例类型支持 EBS 加密，则可以选择 Yes（是）以为此卷启用加密。如果默认情况下在此区域中启用了加密，则会为您启用加密。有关更多信息，请参阅亚马逊 EBS 用户指南中的亚马逊 EBS 加密和默认启用亚马逊 EBS 加密。

设置此参数的默认效果会随所选卷源而异，如下表所述。在所有情况下，您都必须拥有使用指定内容的权限 Amazon KMS key。

加密结果

如果 **Encrypted** 参数设置为...	如果卷源...	则默认加密状态为...	备注
否	新（空）卷	未加密*	不适用
	您拥有的未加密快照	未加密*	
	您拥有的加密快照	按相同密钥加密	
	与您共享的未加密快照	未加密*	
	与您共享的加密快照	由默认 KMS 密钥加密	
是	新卷	由默认 KMS 密钥加密	要使用非原定设置 CMK 密钥，请为 KMS key（KMS 密钥）参数指定一个值。
	您拥有的未加密快照	由默认 KMS 密钥加密	
	您拥有的加密快照	按相同密钥加密	
	与您共享的未加密快照	由默认 KMS 密钥加密	
	与您共享的加密快照	由默认 KMS 密钥加密	

* 如果启用了原定设置加密，则所有新创建的卷 [不论是否将 Encrypted（加密）参数设置为 Yes（是）] 都将使用原定设置 KMS 密钥。如果您同时设置了 Encrypted（加密）和 KMS key（KMS 密钥）参数，则可指定非原定设置 KMS 密钥。

i. KMS key（KMS 密钥）：如果您为 Encrypted（加密）选择的是 Yes（是），则必须选择一个客户管理的密钥来加密该卷。如果默认情况下在此区域中启用了加密，则将为您选择默认的

客户托管密钥。您可以选择其他密钥，或指定您之前使用 Amazon Key Management Service 创建的任何客户管理的密钥的 ARN。

3. 要指定附加卷以挂载到此启动模板启动的实例，请选择 Add new volume（添加新卷）。

从现有实例创建启动模板（控制台）

通过现有实例创建启动模板

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格上的 Instances（实例）下，选择 Instances（实例）。
3. 选择实例，然后依次选择操作、图像和模板、从实例创建模板。
4. 提供名称和说明。
5. 在 Auto Scaling 指导下，选中复选框。
6. 根据需要调整任何参数，然后选择创建启动模板。
7. 要创建 Auto Scaling 组，请从确认页面上选择创建 Auto Scaling 组。

相关资源

我们提供了几个 JSON 和 YAML 模板片段，您可以使用它们来了解如何在 Amazon CloudFormation 堆栈模板中声明启动模板。有关更多信息，请参阅《用户指南》AWS::EC2::LaunchTemplate和《使用Amazon CloudFormation 用户指南》中的创建启动模板 Amazon CloudFormation部分。

有关启动模板的更多信息，请参阅 Amazon EC2 用户指南中的从启动模板启动实例。

限制

- 虽然可以在启动模板中指定子网，但如果您仅使用启动模板创建自动扩缩组，则无需这样做。您无法通过在启动模板中指定子网来为自动扩缩组指定子网。自动扩缩组的子网取自自动扩缩组自己的资源定义。
- 有关用户定义网络接口的其他限制，请参阅 更改默认网络接口设置（控制台）。

使用高级设置创建启动模板

本主题描述了如何从 Amazon Web Services Management Console使用高级设置创建启动模板。

使用高级设置创建启动模板

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格中的实例下，选择启动模板，然后选择创建启动模板。
3. 按照以下主题所述配置启动模板：
   - 必需的设置
   - 高级设置
4. 选择Create launch template（创建启动模板）。

必需的设置

当您创建启动模板时，必须包括以下必需的设置。

启动模板名称

输入描述启动模板的唯一名称。

应用程序和操作系统镜像（亚马逊机器映像）

选择要使用的亚马逊机器映像（AMI）。您可以搜索或浏览要使用的 AMI。为获得最佳扩缩效率，请选择自定义 AMI，其完全配置为使用您的应用程序代码启动实例，在启动时只需进行少量修改。

实例类型

选择与您的 AMI 兼容的实例类型。如果您计划使用自动扩缩组自有资源定义中嵌入的多种实例类型，则可以跳过向启动模板添加实例类型的操作。仅当您不打算创建混合实例组时才需要实例类型。

高级设置

高级设置是可选的。如果未配置任何高级设置，则不会将特定功能添加到您的实例中。

展开高级详细信息部分以查看高级设置。以下各节描述了为自动扩缩组创建启动模板时需要关注的最有用的高级设置。有关更多信息，请参阅 Amazon EC2 用户指南中的高级详情。

IAM 实例配置文件

实例配置文件包含您要使用的 IAM 角色。当您的 Auto Scaling 组启动 EC2 实例时，关联的 IAM 角色中定义的权限将授予在该实例上运行的应用程序。有关更多信息，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。

终止保护

启用后，此功能可防止用户使用 Amazon EC2 控制台、CLI 命令和 API 操作终止实例。终止保护为防止意外终止提供额外的保障。它不会阻止 Amazon A EC2 uto Scaling 终止实例。要控制 Amazon A EC2 uto Scaling 可以终止哪些实例，请参阅使用实例横向缩减保护以控制实例终止。

详细 CloudWatch 监控

您可以对您的 EC2 实例启用详细监控，允许它们以 1 分钟为间隔向 Amazon CloudWatch 发送指标数据。默认情况下， EC2 实例以 5 分钟为间隔 CloudWatch 向发送指标数据。将收取额外费用。有关更多信息，请参阅 配置 Auto Scaling 实例的监控。

积分规范

Amazon EC2 提供可突发性能实例，例如 T2、T3 和 T3a，允许应用程序在需要时突破基准 CPU 性能。默认情况下，在限制其 CPU 使用率之前，这些实例可以在有限的时间内突增。您可以选择启用无限模式，这样实例便可根据需要突增超出基准。这使应用程序能够在需要时保持较高的 CPU 性能。可能收取额外费用。有关更多信息，请参阅《亚马逊 EC2 用户指南》中的 “使用 Auto Scaling 群组以无限制模式启动可突发性能实例”。

置放群组名称

您可以指定置放群组，并使用集群或分区策略来影响您的实例在 Amazon 数据中心的物理位置。对于小型自动扩缩组，您也可以使用分布策略。有关更多信息，请参阅 Amazon EC2 用户指南中的置放群组。

将置放群组与自动扩缩组配合使用时，有一些注意事项：

- 如果在启动模板和自动扩缩组中都指定了一个置放群组，则优先使用自动扩缩组的置放群组。
- 在中 Amazon CloudFormation，如果您在启动模板中定义置放群组，请务必小心。Amazon A EC2 uto Scaling 将在指定的置放群组中启动实例。但是，如果您在 Auto Scaling 组中使用，则 CloudFormation 不会收到来自这些实例的信号（尽管将来这种情况可能会发生变化）。UpdatePolicy

购买选项

您可以选择请求竞价型实例以按照竞价型实例价格请求竞价型实例，以按需价格为上限；而选择自定义可更改默认竞价型实例设置。对于 Auto Scaling 组，必须指定不带结束日期的一次性请求（原定设置）。有关更多信息，请参阅 为容错和灵活的应用程序请求竞价型实例。此设置在特殊情况下可能很有用，但一般而言，最好将其保留为未指定，改为创建混合实例组。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

如果您在启动模板中指定竞价型实例请求，则无法创建混合实例组。如果您尝试使用向混合实例组请求竞价型实例的启动模板，则会收到以下错误消息：`Incompatible launch template:`

```
You cannot use a launch template that is set to request Spot Instances
(InstanceMarketOptions) when you configure an Auto Scaling group with a
mixed instances policy. Add a different launch template to the group and
try again.
```

Capacity Reservation

容量预留允许您在特定可用区内为 Amazon EC2 实例预留任意期限的容量。有关更多信息，请参阅 Amazon EC2 用户指南中的按需容量预留。

您可以选择是否在以下位置启动实例：

- 任何开放的容量预留（打开）
- 特定的容量预留 [目标（按 ID）]
- 一组容量预留 [目标（按组）]

要针对特定的容量预留，启动模板中的实例类型必须与预留的实例类型相匹配。创建自动扩缩组时，请使用与容量预留相同的可用区。根据 Amazon Web Services 区域 您选择的容量块，您可以选择改为瞄准容量块。有关更多信息，请参阅 使用 Capacity Blocks 适用于机器学习工作负载。

要针对一组容量预留，请参阅使用容量预留在特定可用区中预留容量 。通过针对一组容量预留，您可以将容量分配到多个可用区以提高故障恢复能力。

租赁

Amazon 为您的 EC2 实例的租赁 EC2 提供了三种选择：

- 共享（共享）：多个 Amazon Web Services 账户 可以共享相同的物理硬件。这是启动实例时的默认租赁选项。
- 专用实例（专用）：您的实例在单租户硬件上运行。没有其他 Amazon 客户共享同一台物理服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的专用实例。
- 专属主机（专属主机）：在专供您使用的物理服务器上运行的实例。使用专用主机可以更轻松地将具有专用硬件要求的自有许可证 (BYOL) 带到 EC2合规性用例中。如果选择此选项，则必须为租赁主机资源提供主机资源组。有关更多信息，请参阅 Amazon EC2 用户指南中的专用主机。

仅当指定主机资源组时才支持专属主机。您不能定位特定主机 ID 或使用主机放置关联。

- 如果尝试使用指定主机 ID 的启动模板，则您将收到以下错误消息：`Incompatible launch template: Tenancy host ID is not supported for Auto Scaling.`
- 如果尝试使用指定主机放置关联的启动模板，您将收到以下错误消息：`Incompatible launch template: Auto Scaling does not support host placement affinity.`

租赁主机资源组

使用 Amazon License Manager，您可以将自己的许可证带到 Amazon 并集中管理它们。主机资源组是一组专属主机，其链接到特定的 License Manager 许可证配置。主机资源组允许您在符合软件许可需求的专用主机上轻松启动 EC2 实例。您无需提前手动分配专属主机。它们会根据需要自动创建。请注意，当您将 AMI 与许可证配置关联时，该 AMI 一次只能与一个主机资源组关联。有关更多信息，请参阅 License Manager 用户指南中的 Amazon License Manager中的主机资源组。

许可证配置

使用此设置，您可以为实例指定许可证配置，而不必将其租赁限制为专属主机。许可证配置会跟踪部署在实例上的软件许可证，因此您可以监控许可证的使用情况和合规性。有关更多信息，请参阅《License Manager User Guide》中的 Create a self-managed license。

可访问的元数据

您可以选择是启用还是禁用对实例元数据服务的 HTTP 端点的访问。预设情况下，将启用 HTTP 终端节点。如果您选择禁用终端节点，则会关闭对实例元数据的访问。 IMDSv2 只有在启用 HTTP 终端节点时，您才能指定需要的条件。有关更多信息，请参阅 Amazon EC2 用户指南中的配置实例元数据选项。

元数据版本

在请求实例元数据时，您可以选择要求使用实例元数据服务版本 2 (IMDSv2)。如果未指定值，则默认为同时支持 IMDSv1 和 IMDSv2。有关更多信息，请参阅 Amazon EC2 用户指南中的配置实例元数据选项。

元数据令牌响应跃点限制

您可以为元数据令牌设置允许的网络跃点数。如果您未指定值，则原定设置为 1。有关更多信息，请参阅 Amazon EC2 用户指南中的配置实例元数据选项。

用户数据

您可以通过指定 shell 脚本或 cloud-init 指令作为用户数据，在启动时自定义并完成实例配置。用户数据在实例初始启动时运行，允许您在启动时自动安装应用程序、依赖项或自定义项。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Linux 实例启动时运行命令。

如果您的下载量较大或脚本很复杂，则会增加实例准备就绪所需的时间。在这种情况下，您可能需要配置生命周期挂钩，以延迟实例到达 `InService` 状态，直到其完全预置完毕。有关向自动扩缩组添加生命周期挂钩的更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

为容错和灵活的应用程序请求竞价型实例

在启动模板中，您可以选择请求没有结束日期或持续时间的竞价型实例。与 EC2 按需价格相比，Amazon EC2 Spot 实例是备用容量，可享受大幅折扣。如果能灵活控制应用程序的运行时间并且应用程序可以中断，竞价型实例就是经济实惠之选。有关创建请求竞价型实例的启动模板的更多信息，请参阅 使用高级设置创建启动模板。

> **Important**
>
> 竞价型实例通常用于补充按需实例。对于此情景，您可以将用于启动竞价型实例的相同设置指定为 Auto Scaling 组的设置的一部分。当您将设置指定为 Auto Scaling 组的一部分时，您只能在启动一定数量的按需实例后请求启动竞价型实例，然后在组扩展时继续启动按需实例和竞价型实例的某些组合。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

本主题介绍如何通过在启动模板中指定设置而不是在 Auto Scaling 组中指定设置，仅在 Auto Scaling 组中启动竞价型实例。本主题中的信息也适用于请求带启动模板的竞价型实例的 Auto Scaling 组。不同之处在于启动配置需要最高价，但对于启动模板，最高价是可选的。

在创建启动模板以仅启动竞价型实例时，请注意以下事项：

- Spot 价格。您只需为您启动的竞价型实例支付当前 Spot 价格。此定价会根据长期供需趋势缓慢发生变化。有关更多信息，请参阅 Amazon EC2 用户指南中的竞价型实例和定价与优惠。
- 设置您的最高价。您可以选择在启动模板中包含竞价型实例的每小时最高价。如果您的最高价格超过当前的竞价价格，Amazon EC2 Spot 服务将在容量可用时立即满足您的请求。如果竞价型实例的价格超过 Auto Scaling 组中正在运行的实例的最高价，它会终止实例。

  > **Warning**
  >
  > 如果您未收到任何竞价型实例 (例如当您的最高价太低时)，您的应用程序可能不运行。要尽可能长时间利用可用的竞价型实例，请将最高价设置为接近按需价格。

- 在可用区之间平衡。如果您指定多个可用区，Amazon A EC2 uto Scaling 会将竞价请求分发到指定的区域。如果您在一个可用区的最高价格过低，无法满足任何请求，Amazon A EC2 uto Scaling 会检查其他区域是否已完成请求。如果是这样，Amazon A EC2 uto Scaling 会取消失败的请求，并将这些请求重新分配到已完成请求的可用区。如果未完成请求的可用区内的价格下降到足以使未来的请求成功，则 Amazon A EC2 uto Scaling 会在所有可用区之间进行重新平衡。

- 竞价型实例终止。竞价型实例可以随时终止。当 EC2 竞价型实例的可用性或价格发生变化时，Amazon 竞价服务可以终止您的 Auto Scaling 组中的竞价型实例。在扩展或执行运行状况检查时，Amazon A EC2 uto Scaling 还可以像终止按需实例一样终止竞价型实例。当实例终止时，任何存储都将被删除。
- 保持所需容量。竞价型实例终止后，Amazon A EC2 uto Scaling 会尝试启动另一个竞价型实例以保持该组所需的容量。如果最高价高于当前 Spot 价格，则会启动竞价型实例。如果竞价型实例请求失败，它将继续尝试。
- 更改您的最高价。要更改最高价，请创建新的启动模板或使用新的最高价更新现有启动模板，然后将其与 Auto Scaling 组关联。只要现有竞价型实例所用的启动模板中指定的最高价高于当前 Spot 价格，这些实例就会继续运行。如果没有设置最高价，则默认最高价为按需价格。

使用 Capacity Blocks 适用于机器学习工作负载

Capacity Blocks 帮助您在将来的某个日期预留备受追捧的 GPU 实例，以支持您的短时机器学习 (ML) 工作负载。

有关以下内容的概述 Capacity Blocks 以及它们是如何工作的，请参阅 Capacity Blocks 在 Amazon EC2 用户指南中查看 ML。

要开始使用 Capacity Blocks，您可以在特定的可用区中创建容量预留。Capacity Blocks 在单个可用区中作为`targeted`容量预留交付。创建启动模板时，请指定容量块的预留 ID 和实例类型。然后，更新您的自动扩缩组，以使用您创建的启动模板和容量块的可用区。当您的容量块预留开始时，使用计划扩缩启动与容量块预留相同数量的实例。

> **Important**
>
> Capacity Blocks 仅适用于某些 Amazon EC2 实例类型和 Amazon Web Services 区域。有关更多信息，请参阅 Amazon EC2 用户指南中的先决条件。

内容

操作指导方针

以下是将容量块与自动扩缩组结合使用时应遵循的基本操作指导方针。

- 在容量块预留结束时间前 30 分钟以上，将自动扩缩组横向缩减到零。Amazon EC2 将在容量封锁结束前 30 分钟终止所有仍在运行的实例。
- 建议您在适当的预留时间使用计划的扩缩来横向扩展（添加实例）和横向缩减（移除实例）。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。
- 根据需要添加生命周期挂钩，以便在缩小实例时正常关闭实例内的应用程序。 EC2在 Amazon 在容量块预留结束前 30 分钟开始强制终止您的实例，请留出足够的时间让生命周期操作完成。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。
- 确保自动扩缩组在整个预留期间指向启动模板的正确版本。我们建议指向启动模板的特定版本，而不是 `$Default` 或 `$Latest` 版本。

> **Note**
>
> 如果您让容量块实例一直运行到预留结束并且 Amazon 收 EC2 回了该实例，则您的 Auto Scaling 组的扩展活动会将其显示`taken out of service in response to an EC2 health check that indicated it had been terminated or stopped`为“”，尽管它是在容量块结束时故意回收的。同样，Amazon A EC2 uto Scaling 将尝试替换实例，方法与替换任何未通过运行状况检查的实例的方式相同。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

在启动模板中指定容量块

要创建针对自动扩缩组特定容量块的启动模板，请使用以下方法之一：

Console

在启动模板中指定容量块（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在顶部导航栏上，选择您创建容量块 Amazon Web Services 区域 的位置。
3. 在导航窗格中的实例下，选择启动模板。
4. 选择创建启动模板，然后创建启动模板。根据需要包括Amazon Machine Images (AMI)的 ID、实例类型和任何其他启动模板设置。

5. 展开高级详细信息部分以查看高级设置。
6. 对于购买选项，选择容量块。
7. 对于容量预留，选择按 ID 定位，然后对于容量预留 - 按 ID 定位，选择现有容量块的容量预留 ID。
8. 完成后，选择创建启动模板。

有关使用启动模板创建自动扩缩组的帮助，请参阅使用启动模板创建 Auto Scaling 组。

Amazon CLI

要在启动模板中指定容量块 (Amazon CLI)

使用以下create-launch-template命令创建用于指定现有容量块预留 ID 的启动模板。将每个 *user input placeholder* 替换为您自己的信息。

```
aws ec2 create-launch-template --launch-template-name my-template-for-capacity-block
  \
  --version-description AutoScalingVersion1 --region us-east-2 \
  --launch-template-data file://config.json
```

> **Tip**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

config.json 的内容。

```
{
    "ImageId": "ami-04d5cc9b88example",
    "InstanceType": "p4d.24xlarge",
    "SecurityGroupIds": [
        "sg-903004f88example"
    ],
    "KeyName": "MyKeyPair",
    "InstanceMarketOptions": {
        "MarketType": "capacity-block"
    },
    "CapacityReservationSpecification": {
        "CapacityReservationTarget": {
            "CapacityReservationId": "cr-02168da1478b509e0"
```

```
        }
    }
}
```

下面是示例输出。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-capacity-block",
        "CreateTime": "2023-10-27T15:12:44.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

您可以使用以下describe-launch-template-versions命令来验证与启动模板关联的容量块预留 ID。

```
aws ec2 describe-launch-template-versions --launch-template-names my-template-for-capacity-block \
  --region us-east-2
```

以下是指定容量块预留的启动模板的示例输出。

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-068f72b724example",
            "LaunchTemplateName": "my-template-for-capacity-block",
            "VersionNumber": 1,
            "CreateTime": "2023-10-27T15:12:44.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-04d5cc9b88example",
                "InstanceType": "p5.48xlarge",
                "SecurityGroupIds": [
                    "sg-903004f88example"
                ],
                "KeyName": "MyKeyPair",
                "InstanceMarketOptions": {
```

```
                    "MarketType": "capacity-block"
                },
                "CapacityReservationSpecification": {
                    "CapacityReservationTarget": {
                        "CapacityReservationId": "cr-02168da1478b509e0"
                    }
                }
            }
        }
    ]
}
```

限制

- 对该项的支持 Capacity Blocks 仅当您的 Auto Scaling 组具有兼容的配置时才可用。不支持混合实例组和暖池。
- 您一次只能针对一个容量块。

相关资源

- 有关使用 P5 实例的先决条件和建议，请参阅 A mazon EC2 用户指南中的 P5 实例入门。
- 亚马逊 EKS 支持使用 Capacity Blocks 以支持 Amazon EKS 集群上的短时机器学习 (ML) 工作负载。有关更多信息，请参阅 Capacity Blocks在 Amazon EKS 用户指南中查看机器学习。
- 您可以使用 … Capacity Blocks 包含支持的实例类型和区域。但是，按需容量预留可以灵活地为其他实例类型和区域预留容量。有关演示如何使用按需容量预留选项的教程，请参阅使用容量预留在特定可用区中预留容量 。

将自动扩缩组迁移到启动模板

自 2023 年 1 月 1 日起，启动配置中不再支持新的实例类型。这适用于在初始区域启动 Amazon Web Services 区域 后添加到中的任何实例类型。此外，您还可以使用某个地区不再支持的实例类型创建启动配置。有关更多信息，请参阅 自动扩缩启动配置。

要将自动扩缩组从启动配置迁移到启动模板，请参阅以下步骤。

> **Important**
>
> 在继续操作之前，请确认您拥有使用启动模板所需的权限。有关更多信息，请参阅 使用启动模板的权限。

步骤 1：查找使用启动配置的自动扩缩组

要确定您的 Auto Scaling 组是否仍在使用启动配置，请使用运行以下describe-auto-scaling-groups命令 Amazon CLI。替换*REGION*为你的 Amazon Web Services 区域。

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

下面是示例输出。

```
[
    {
 "AutoScalingGroupName": "group-1",
 "AutoScalingGroupARN": "arn",
 "LaunchConfigurationName": "my-launch-config",
 "MinSize": 1,
 "MaxSize": 5,
 "DesiredCapacity": 2,
 "DefaultCooldown": 300,
 "AvailabilityZones": [
           "us-west-2a",
           "us-west-2b",
           "us-west-2c"
       ],
 "LoadBalancerNames": [],
 "TargetGroupARNs": [],
 "HealthCheckType": "EC2",
 "HealthCheckGracePeriod": 300,
 "Instances": [
           {
               "ProtectedFromScaleIn": false,
               "AvailabilityZone": "us-west-2a",
               "LaunchConfigurationName": "my-launch-config",
               "InstanceId": "i-05b4f7d5be44822a6",
               "InstanceType": "t3.micro",
               "HealthStatus": "Healthy",
```

```
                "LifecycleState": "InService"
            },
            {
                "ProtectedFromScaleIn": false,
                "AvailabilityZone": "us-west-2b",
                "LaunchConfigurationName": "my-launch-config",
                "InstanceId": "i-0c20ac468fa3049e8",
                "InstanceType": "t3.micro",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            }
 ],
 "CreatedTime": "2023-03-09T22:15:11.611Z",
 "SuspendedProcesses": [],
 "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
 "EnabledMetrics": [],
 "Tags": [
            {
  "ResourceId": "group-1",
  "ResourceType": "auto-scaling-group",
  "Key": "environment",
  "Value": "production",
  "PropagateAtLaunch": true
            }
        ],
 "TerminationPolicies": [
     "Default"
 ],
 "NewInstancesProtectedFromScaleIn": false,
 "ServiceLinkedRoleARN": "arn",
       "TrafficSources": []
    },

    ... additional groups ...

]
```

或者，要删除除输出中包含相应启动配置名称和标签的自动扩缩组名称之外的所有内容，请运行以下命令：

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`].{AutoScalingGroupName:
 AutoScalingGroupName, LaunchConfigurationName: LaunchConfigurationName, Tags: Tags}'
```

下面显示了示例输出。

```
[
    {
        "AutoScalingGroupName": "group-1",
        "LaunchConfigurationName": "my-launch-config",
        "Tags": [
            {
                "ResourceId": "group-1",
                "ResourceType": "auto-scaling-group",
                "Key": "environment",
                "Value": "production",
                "PropagateAtLaunch": true
            }
        ]
    },

    ... additional groups ...

]
```

有关筛选的更多信息，请参阅《Amazon Command Line Interface 用户指南》中的筛选 Amazon CLI 输出。

步骤 2：将启动配置复制到启动模板

您可以使用以下过程将启动配置复制到启动模板。然后，您可以将其添加到自动扩缩组。

复制多个启动配置会生成同名的启动模板。要在复制过程中更改启动模板的名称，必须逐个复制启动配置。

> **Note**
>
> 复制功能只能从控制台提供。

将启动配置复制到启动模板（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。

3. 在页面顶部附近，选择启动配置。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
4. 选择您要复制的启动配置，然后选择复制到启动模板，复制所选。这将设置一个与您所选启动配置具有相同名称和选项的新启动模板。
5. 对于新建启动模板名称，您可以使用启动配置的名称（默认值）或者输入新名称。启动模板名称必须是唯一的。
6. （可选）选择使用新模板创建自动扩缩组。

   您可以跳过此步骤以完成启动配置的复制。您无需创建新的自动扩缩组。
7. 选择复制。

将所有启动配置复制到启动模板（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的 Auto Scaling (自动调整) 上，选择 Launch Configurations (启动配置)。
3. 选择复制到启动模板，全部复制。这将当前地区中的每个启动配置复制到具有相同名称和选项的新启动模板。
4. 选择 Copy（复制）。

步骤 3：更新自动扩缩组以使用启动模板

创建启动模板后，您可以将其添加到自动扩缩组。

更新自动扩缩组以使用启动模板（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   将在页面底部打开一个拆分窗格，其中显示有关所选组的信息。
3. 在详细信息选项卡上，选择启动配置、编辑。
4. 选择切换到启动模板。
5. 对于启动模板，选择您的启动模板。
6. 对于版本，根据需要选择启动模板版本。在创建启动模板版本之后，您可以选择 Auto Scaling 组在扩展时是使用启动模板的默认版本还是最新版本。

7. 选择更新。

更新自动扩缩组以使用启动模板 (Amazon CLI)

以下update-auto-scaling-group命令更新指定的 Auto Scaling 组以使用指定启动模板的初始版本。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
```

有关使用 CLI 命令更新自动扩缩组以使用启动模板的更多示例，请参阅更新 Auto Scaling 组以使用启动模板。

步骤 4：替换实例

将启动配置替换为启动模板后，任何新实例都将使用新的启动模板。现有实例不会受到影响。

要更新现有的实例，您可以启动实例刷新替换自动扩缩组中的实例，而不是一次手动替换几个实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。如果组很大，则实例刷新可能会特别有用。

或者，您可以允许自动扩展以根据组的终止策略逐步将现有实例替换为新实例，也可以将其终止。手动终止会强制您的自动扩缩组启动新实例以保持该组的所需容量。有关更多信息，请参阅 Amazon EC2 用户指南中的终止实例。

其他信息

有关更多信息，请参阅 Amazon A EC2 uto Scaling 将不再为 Amazon 计算博客上的启动配置添加对新 EC2 功能的支持。

有关引导您了解如何将 Amazon CloudFormation 堆栈从启动配置迁移到启动模板的主题，请参阅将 Amazon CloudFormation 堆栈迁移到启动模板。

将 Amazon CloudFormation 堆栈迁移到启动模板

您可以将现有的 Amazon CloudFormation 堆栈模板从启动配置迁移到启动模板。为此，请将启动模板直接添加到现有堆栈模板中，然后将启动模板与堆栈模板中的自动扩缩组相关联。然后使用您的经过修改的模板更新您的堆栈。

迁移到启动模板时，本主题提供了将 CloudFormation 堆栈模板中的启动配置重写为启动模板的说明，从而为您节省时间。有关将启动配置迁移到启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

主题

查找使用启动配置的自动扩缩组

查找使用启动配置的自动扩缩组

- 使用以下describe-auto-scaling-groups命令列出在指定区域中使用启动配置的 Auto Scaling 组的名称。包括将结果范围缩小到与 CloudFormation 堆栈关联的组的`--filters`选项（通过按`aws:cloudformation:stack-name`标签键筛选）。

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --filters Name=tag-key,Values=aws:cloudformation:stack-name \
  --query 'AutoScalingGroups[?LaunchConfigurationName!
=`null`].AutoScalingGroupName'
```

下面显示了示例输出。

```
[
    "{stack-name}-group-1",
    "{stack-name}-group-2",
    "{stack-name}-group-3"
]
```

您可以找到其他有用的 Amazon CLI 命令，用于查找要迁移的 Auto Scaling 组和筛选输出将自动扩缩组迁移到启动模板。

> **Important**
>
> 如果您的堆栈资源名称AWSEB中有，则表示它们是通过创建的 Amazon Elastic Beanstalk。在这种情况下，您必须更新 Beanstalk 环境以指示 Elastic Beanstalk 删除启动配置并将其替换为启动模板。

更新堆栈以使用启动模板

请按照本部分中的步骤，来执行以下操作：

- 使用等效的启动模板属性将启动配置重写为启动模板。
- 关联新启动模板与自动扩缩组。
- 部署这些更新。

修改堆栈模板并更新堆栈

1. 按照Amazon CloudFormation 用户指南中修改堆栈模板的中所述的修改堆栈模板一般步骤进行操作。
2. 将启动配置重写为启动模板。请参见以下示例：

   示例：简单的启动配置

```
---
Resources:
  myLaunchConfig:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId: ami-02354e95b3example
      InstanceType: t3.micro
      SecurityGroups:
        - !Ref EC2SecurityGroup
      KeyName: MyKeyPair
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: 150
            DeleteOnTermination: true
      UserData:
        Fn::Base64: !Sub |
```

```
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource myASG
 --region ${AWS::Region}
```

示例：等效的启动模板

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        SecurityGroupIds:
          - Ref! EC2SecurityGroup
        KeyName: MyKeyPair
        BlockDeviceMappings:
          - DeviceName: /dev/xvda
            Ebs:
              VolumeSize: 150
              DeleteOnTermination: true
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash -x
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 myASG --region ${AWS::Region}
```

有关 Amazon EC2 支持的所有属性的参考信息，请参阅Amazon CloudFormation 用户指南中的。

请注意启动模板如何包含值为 `!Sub ${AWS::StackName}-launch-template` 的 `LaunchTemplateName` 属性。如果您要在启动模板的名称中包含堆栈名称，则必须执行此操作。

3. 如果启动配置中存在 **`IamInstanceProfile`** 属性，则必须将其转换为结构并指定实例配置文件的名称或 ARN。有关示例，请参阅AWS::EC2::LaunchTemplate。
4. 如果启动配置中存在 **`AssociatePublicIpAddress`**、**`InstanceMonitoring`** 或 **`PlacementTenancy`** 属性，则必须将其转换为结构。有关示例，请参阅 AWS::EC2::LaunchTemplate。

一个例外情况是，当您用于自动扩缩组的子网上的 MapPublicIpOnLaunch 属性值与启动配置中的 AssociatePublicIpAddress 属性值相匹配时。在这种情况下，您可以忽略 AssociatePublicIpAddress 属性。该AssociatePublicIpAddress属性仅用于覆盖该MapPublicIpOnLaunch属性以更改实例在启动时是否接收公共 IPv4 地址。

5. 您可以将安全组从 **SecurityGroups** 属性复制到启动模板中的两个位置之一。通常，您可以将安全组复制到 SecurityGroupIds 属性。但是，如果您在启动模板中创建 NetworkInterfaces 结构来指定 AssociatePublicIpAddress 属性，则必须将安全组复制到网络接口的 Groups 属性中。
6. 如果您的启动配置中存在任何**NoDevice**设置为的BlockDeviceMapping结构true，则必须在启动模板NoDevice中为指定一个空字符串，才能让 Amazon EC2 省略该设备。
7. 如果您的启动配置中存在 **SpotPrice** 属性，我们建议您将其从启动模板中省略。您的竞价型实例将以当前的 Spot 价格启动。该价格永远不会超过按需价格。

   要请求竞价型实例，您有两个互斥的选项：

   - 第一种是在启动模板中使用 InstanceMarketOptions 结构（不推荐）。有关更多信息，请参阅《Amazon CloudFormation 用户指南》。
   - 另一种方法是将 MixedInstancesPolicy 结构添加到自动扩缩组。这样做可以为你提供更多关于如何提出请求的选项。您的启动模板中的竞价型实例请求不支持每个自动扩缩组选择多个实例类型。但是，混合实例策略支持每个自动扩缩组选择多个实例类型。竞价型实例请求可以受益于有多个实例类型可供选择。有关更多信息，请参阅《Amazon CloudFormation 用户指南》MixedInstancesPolicy中的 MixedInstancesPolicyAWS::AutoScaling::AutoScaling群群组。
8. 从资源中移除该**LaunchConfigurationName**属性。将启动模板添加到其位置上。

   在以下示例中，Ref 内部函数获取具有逻辑 ID 的资源 ID。myLaunchTemplate该GetAtt函数获取该Version属性的启动模板的最新版本号（例如1）。

   示例：没有混合实例策略

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !GetAtt myLaunchTemplate.LatestVersionNumber
```

```
      ...
```

示例：使用混合实例策略

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref myLaunchTemplate
            Version: !GetAtt myLaunchTemplate.LatestVersionNumber
      ...
```

有关 Amazon A EC2 uto Scaling 支持的所有属性的参考信息，请参阅Amazon CloudFormation 用户指南中的AWS::AutoScaling::AutoScaling群。

9. 准备好部署这些更新后，请按照 CloudFormation步骤使用修改后的堆栈模板更新堆栈。有关更多信息，请参阅Amazon CloudFormation 用户指南中的修改堆栈模板。

理解堆栈资源的更新行为

CloudFormation 通过比较您提供的更新模板与您在先前版本的堆栈模板中描述的资源配置之间的更改来更新堆栈资源。尚未更改的资源配置在更新过程中不会受到影响。

CloudFormation 支持 Auto Scaling 组的UpdatePolicy属性。更新期间，如果设置`UpdatePolicy`为`AutoScalingRollingUpdate`，则在执行此过程中的步骤后 CloudFormation 替换`InService`实例。如果设置`UpdatePolicy`为`AutoScalingReplacingUpdate`，则 CloudFormation 替换 Auto Scaling 组及其温池（如果存在）。

如果您没有为 Auto Scaling 组指定`UpdatePolicy`属性，则会检查启动模板的正确性，但 CloudFormation 不会在 Auto Scaling 组中的实例之间部署任何更改。所有新实例都使用您的启动模板，但现有实例会继续使用其最初启动的配置运行（除非不存在启动配置）。例外情况是当您更改购买选项时，例如通过添加混合实例策略。在这种情况下，您的自动扩缩组会逐渐用新实例替换现有实例，以匹配新的购买选项。

如果您必须回滚更改才能从启动配置切换到启动模板，则请务必测试回滚操作。

跟踪迁移

跟踪迁移

1. 在Amazon CloudFormation 控制台中，选择已更新的堆栈，然后单击事件选项卡，查看堆栈事件。
2. 要使用最新事件更新事件列表，请选择 CloudFormation 控制台中的刷新按钮。
3. 在堆栈更新时，您会注意到每次资源更新都会发生多个事件。如果您在 “状态原因” 列中看到异常，表明在尝试创建启动模板时出现问题，请参阅 对 Amazon A EC2 uto Scaling 进行故障排除：启动模板 以了解潜在原因。
4. （可选）根据您对UpdatePolicy属性的使用情况，您可以从亚马逊 EC2 控制台的 Auto Scaling 群组页面监控 Auto Scaling 群组的进度。选择 Auto Scaling 组。在 Activity（活动）选项卡的 Activity history（活动历史记录）下，Status（状态）列显示您的 Auto Scaling 组是否已成功启动或终止实例，或者扩展活动是否仍在进行中。
5. 堆栈更新完成后， CloudFormation 发出UPDATE_COMPLETE堆栈事件。有关更多信息，请参阅Amazon CloudFormation 用户指南中的监控堆栈更新的进度。
6. 堆栈更新完成后，打开 Amazon EC2 控制台的启动模板页面和启动配置页面。您会注意到已创建新的启动模板，并且已删除启动配置。

启动配置映射参考

为了便于参考，下表列出了资源中的所有顶级属性及其在资源中的相应属性。

启动配置源属性	启动模板目标属性
AssociatePublicIpAddress	NetworkInterfaces.Associate PublicIpAddress
BlockDeviceMappings	BlockDeviceMappings
ClassicLinkVPCId	不可用[1]
ClassicLinkVPCSecurityGroups	不可用[1]
EbsOptimized	EbsOptimized

启动配置源属性	启动模板目标属性
IamInstanceProfile	选择在 IamInstanceProfile.Arn 或 IamInstanceProfile.Name ，但不能同时选择两者
ImageId	ImageId
InstanceId	InstanceId
InstanceMonitoring	Monitoring.Enabled
InstanceType	InstanceType
KernelId	KernelId
KeyName	KeyName
LaunchConfigurationName	LaunchTemplateName
MetadataOptions	MetadataOptions
PlacementTenancy	Placement.Tenancy
RamDiskId	RamDiskId
SecurityGroups	选择在 SecurityGroupIds 或 NetworkInterfaces.Groups ，但不能同时选择两者
SpotPrice	InstanceMarketOptions.SpotOptions.MaxPrice
UserData	UserData

[1] ClassicLinkVPCId 和ClassicLinkVPCSecurityGroups属性不可用于启动模板，因为 EC2-Classic 不再可用。

使用创建和管理启动模板的示例 Amazon CLI

您可以通过 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或创建和管理启动模板 SDKs。本节向您展示了从中创建和管理 Amazon A EC2 uto Scaling 启动模板的示例 Amazon CLI。

内容

示例用法

```
{
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "VersionDescription": "test description",
    "LaunchTemplateData": {
        "ImageId": "ami-04d5cc9b88example",
        "InstanceType": "t2.micro",
        "SecurityGroupIds": [
            "sg-903004f88example"
        ],
        "KeyName": "MyKeyPair",
        "Monitoring": {
            "Enabled": true
        },
        "Placement": {
```

```
            "Tenancy": "dedicated"
        },
        "CreditSpecification": {
            "CpuCredits": "unlimited"
        },
        "MetadataOptions": {
            "HttpTokens": "required",
            "HttpPutResponseHopLimit": 1,
            "HttpEndpoint": "enabled"
        }
    }
}
```

创建基本的启动模板

要创建基本的启动模板，请按以下方式使用create-launch-template命令，并进行以下修改：

- 将 ami-04d5cc9b88example 替换为要从其中启动实例的 AMI 的 ID。
- 将 t2.micro 替换为与指定的 AMI 兼容的实例类型。

此示例创建了一个名为的启动模板*my-template-for-auto-scaling*。如果在默认 VPC 中启动由此启动模板创建的实例，则默认情况下它们会收到一个公有 IP 地址。如果在非默认 VPC 中启动实例，则默认情况下它们不会收到一个公有 IP 地址。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

有关引用 JSON 格式参数的更多信息，请参阅 Amazon Command Line Interface 用户指南中的在Amazon CLI中将引号和字符串结合使用。

或者，您可以在配置文件中指定 JSON 格式参数。

以下示例创建了基本的启动模板，其中引用了启动模板参数值的配置文件。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data file://config.json
```

config.json 的内容：

```
{
    "ImageId":"ami-04d5cc9b88example",
    "InstanceType":"t2.micro"
}
```

指定在启动时标记实例的标签

以下示例在启动时将标签（例如，purpose=webserver）添加到实例。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"purpose","Value":"webserver"}]}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.
```

> **Note**
>
> 如果您在启动模板中指定了实例标签，然后选择将 Auto Scaling 组的标签传播到其实例，则所有标签都会合并。如果为启动模板中的标签和 Auto Scaling 组中的标签指定了相同的标签键，则优先使用该组中的标签值。

指定要传递到实例的 IAM 角色

以下示例指定启动时要传递给实例的与 IAM 角色关联的实例配置文件的名称。有关更多信息，请参阅适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data '{"IamInstanceProfile":{"Name":"my-instance-
profile"},"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

分配公有 IP 地址

以下create-launch-template示例将启动模板配置为向在非默认 VPC 中启动的实例分配公有地址。

> **Note**
>
> 指定网络接口时，请为 Groups 指定值（对应于 Auto Scaling 组将实例启动到其中的 VPC 的安全组）。指定 VPC 子网作为 Auto Scaling 组的属性。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

指定用于在启动时配置实例的用户数据脚本

以下示例将用户数据脚本指定为 base64 编码的字符串，用于在启动时配置实例。该create-launch-template命令需要使用 base64 编码的用户数据。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data
 '{"UserData":"IyEvYmluL2Jhc...","ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

指定块储存设备映射

以下create-launch-template示例创建了一个带有块储存设备映射的启动模板：一个 22 GB 的 EBS 卷映射到。/dev/xvdcz/dev/xvdcz 卷使用通用型 SSD (gp2) 卷类型，在它连接的实例终止时会被删除。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"BlockDeviceMappings":[{"DeviceName":"/dev/xvdcz","Ebs":
{"VolumeSize":22,"VolumeType":"gp2","DeleteOnTermination":true}}],"ImageId":"ami-04d5cc9b88exam
```

指定专属主机以从外部供应商获得软件许可证

如果您指定主机租赁，您可以指定主机资源组和 License Manager 许可证配置，以便从外部供应商获得合格的软件许可证。然后，您可以使用以下create-launch-template命令在 EC2 实例上使用许可证。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"Placement":
{"Tenancy":"host","HostResourceGroupArn":"arn"},"LicenseSpecifications":
[{"LicenseConfigurationArn":"arn"}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"
```

指定现有网络接口

以下create-launch-template示例将主网络接口配置为使用现有网络接口。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"NetworkInterfaceId":"eni-
b9a5ac93","DeleteOnTermination":false}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.mi
```

创建多个网络接口

以下create-launch-template示例添加了一个辅助网络接口。主网络接口的设备索引为 0，而辅助网络接口的设备索引为 1。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true},{"DeviceIndex":1,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

如果您使用的实例类型支持多个网卡和弹性结构适配器 (EFAs)，则可以使用以下create-launch-template命令向辅助网卡添加辅助接口并启用 EFA。有关更多信息，请参阅 Amazon EC2 用户指南中的向启动模板添加 EFA。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true}],"ImageId":"ami-09d95
```

> **Warning**
>
> p4d.24xlarge 实例类型产生的成本高于本节中的其他示例。有关 P4d 实例定价的更多信息，请参阅 A mazon EC2 P4 d 实例定价。

> **Note**
>
> 将来自同一子网的多个网络接口附加到一个实例可能会引入非对称路由，尤其是在使用非 Amazon Linux 变体的实例上。如果需要此类配置，则必须在操作系统中配置辅助网络接口。有关示例，请参阅如何使我的辅助网络接口在我的 Ubuntu EC2 实例中运行？ 在 Amazon 知识中心中。

管理启动模板

Amazon CLI 包括其他几个可帮助您管理启动模板的命令。

内容

列出并描述启动模板

您可以使用两个 Amazon CLI 命令来获取有关启动模板的信息：describe-launch-templates和describe-launch-template-versions。

该describe-launch-templates命令使您可以获取已创建的任何启动模板的列表。您可以使用选项筛选启动模板名称、创建时间、标签键或标记键值组合的结果。此命令可返回有关任何启动模板的汇总信息，包括启动模板标识符、最新版本和默认版本。

以下示例提供指定启动模板的汇总。

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

以下为响应示例。

```
{
    "LaunchTemplates": [
        {
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
```

```
            "CreateTime": "2020-02-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersionNumber": 1,
            "LatestVersionNumber": 1
        }
    ]
}
```

如果您未使用 `--launch-template-names` 选项将输出限制为一个启动模板，则将返回所有启动模板的相关信息。

以下describe-launch-template-versions命令提供描述指定启动模板版本的信息。

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

以下为响应示例。

```
{
    "LaunchTemplateVersions": [
        {
            "VersionDescription": "version1",
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "LaunchTemplateData": {
                "TagSpecifications": [
                    {
                        "ResourceType": "instance",
                        "Tags": [
                            {
                                "Key": "purpose",
                                "Value": "webserver"
                            }
                        ]
                    }
                ],
                "ImageId": "ami-04d5cc9b88example",
                "InstanceType": "t2.micro",
                "NetworkInterfaces": [
                    {
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
```

```
                        "Groups": [
                            "sg-903004f88example"
                        ],
                        "AssociatePublicIpAddress": true
                    }
                ]
            },
            "DefaultVersion": true,
            "CreateTime": "2020-02-28T19:52:27.000Z"
        }
    ]
}
```

创建启动模板版本

以下create-launch-template-version命令基于启动模板的版本 1 创建新的启动模板版本并指定不同的 AMI ID。

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --
version-description version2 \
  --source-version 1 --launch-template-data "ImageId=ami-c998b6b2example"
```

要设置启动模板的默认版本，请使用modify-launch-template命令。

删除启动模板版本

以下delete-launch-template-versions命令删除指定的启动模板版本。

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --
versions 1
```

删除启动模板

如果您不再需要启动模板，则可以使用以下delete-launch-template命令将其删除。如果删除启动模板，则会删除该模板的所有版本。

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

更新 Auto Scaling 组以使用启动模板

您可以使用update-auto-scaling-group命令将启动模板添加到现有 Auto Scaling 组中。

更新 Auto Scaling 组以使用最新版本的启动模板

以下update-auto-scaling-group命令更新指定的 Auto Scaling 组，使其使用指定启动模板的最新版本。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

更新 Auto Scaling 组以使用特定版本的启动模板

以下update-auto-scaling-group命令更新指定的 Auto Scaling 组，使其使用指定启动模板的特定版本。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

IDs 在启动模板中使用 Amazon Systems Manager 参数而不是 AMI

本节向您演示如何创建启动模板，该模板指定引用亚马逊机器映像（AMI）ID 的 Amazon Systems Manager 参数。您可以使用存储在您的同一个参数中的参数 Amazon Web Services 账户、从另一个 Amazon Web Services 账户共享的参数或由维护的公共 AMI 的公共参数 Amazon。

使用 Systems Manager 参数，您可以更新 Auto Scaling 组以使用新的 AMI，而 IDs 无需在每次更改 AMI ID 时创建新的启动模板或新版本的启动模板。它们 IDs 可能会定期更改，例如在 AMI 使用最新操作系统或软件更新进行更新时。

您可以使用参数存储器创建、更新或删除自己的 Systems Manager 参数，该功能为 Amazon Systems Manager。必须先创建 Systems Manager 参数，然后才能在启动模板中使用该参数。首先，您可以创建数据类型为 `aws:ec2:image` 的参数，并输入 AMI ID 作为其值。AMI ID 的格式为 `ami-<identifier>`，例如，`ami-123example456`。AMI ID 是否正确取决于您在其中启动自动扩缩组的实例类型和 Amazon Web Services 区域 。

有关为 AMI ID 创建有效参数的更多信息，请参阅创建 Systems Manager 参数。

创建指定 AMI 参数的启动模板

要创建指定 AMI 参数的启动模板，请使用以下方法之一：

Console

使用 Amazon Systems Manager 参数创建启动模板

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。

2. 在导航窗格中，选择启动模板，然后选择创建启动模板。
3. 对于设备模板名称，请为您的启动模板输入描述性名称。
4. 在 “应用程序和操作系统映像（Amazon 系统映像）” 下，选择 “浏览更多” AMIs。
5. 选择搜索栏右侧的箭头按钮，然后选择指定自定义值/Systems Manager 参数。
6. 在指定自定义值或 Systems Manager 参数对话框中，执行以下操作：

   a. 对于 AMI ID 或 Systems Manager 参数字符串，使用以下格式之一输入 Systems Manager 参数名称：

      要引用公有参数，请执行以下操作：

      - **resolve:ssm:*public-parameter***

      要引用存储在同一账户中的参数，请执行以下操作：

      - **resolve:ssm:*parameter-name***
      - **resolve:ssm:*parameter-name*:*version-number***
      - **resolve:ssm:*parameter-name*:*label***

      要引用其他 Amazon Web Services 账户共享的参数，请执行以下操作：

      - **resolve:ssm:*parameter-ARN***
      - **resolve:ssm:*parameter-ARN*:*version-number***
      - **resolve:ssm:*parameter-ARN*:*label***

   b. 选择保存。
7. 根据需要配置任何其他启动模板设置，然后选择创建启动模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

Amazon CLI

要创建指定 Systems Manager 参数的启动模板，您可以使用以下示例命令之一。将每个 *user input placeholder* 替换为您自己的信息。

示例：创建一个启动模板，该模板指定了 Amazon自有的 public 参数

请使用以下语法：resolve:ssm:*public-parameter*，其中 resolve:ssm 是标准前缀，*public-parameter* 是公有参数的路径和名称。

在此示例中，启动模板使用 Amazon提供的公共参数，使用中为您的配置文件配置的最新 Amazon Linux 2 AMI 启动实例。 Amazon Web Services 区域

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
  --launch-template-data file://config.json
```

config.json 的内容：

```
{
    "ImageId":"resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2",
    "InstanceType":"t2.micro"
}
```

以下为响应示例。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-089c023a30example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-28T19:52:27.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

示例：创建指定存储在同一账户中的参数的启动模板

使用以下语法：resolve:ssm:*parameter-name*，其中 resolve:ssm 是标准前缀，*parameter-name* 是 Systems Manager 参数名称。

以下示例可创建一个从名为 *golden-ami* 的现有 Systems Manager 参数获取 AMI ID 的启动模板。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling \
  --launch-template-data file://config.json
```

config.json 的内容：

```
{
    "ImageId":"resolve:ssm:golden-ami",
    "InstanceType":"t2.micro"
}
```

如果未指定，则参数的默认版本为最新版本。

以下示例引用 *golden-ami* 参数的特定版本。该示例使用 *golden-ami* 参数的版本 *3*，但您可以使用任何有效的版本号。

```
{
    "ImageId":"resolve:ssm:golden-ami:3",
    "InstanceType":"t2.micro"
}
```

以下类似示例引用了映射到 *golden-ami* 参数特定版本的参数标签 *prod*。

```
{
    "ImageId":"resolve:ssm:golden-ami:prod",
    "InstanceType":"t2.micro"
}
```

下面是示例输出。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-27T17:11:21.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

示例：创建一个启动模板，该模板指定了与另一个共享的参数 Amazon Web Services 账户

使用以下语法：resolve:ssm:*parameter-ARN*，其中 resolve:ssm 是标准前缀，而 *parameter-ARN* 是 Systems Manager 参数的 ARN。

以下示例可创建一个从 ARN 为 *arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter* 的现有 Systems Manager 参数获取 AMI ID 的启动模板。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
  --launch-template-data file://config.json
```

config.json 的内容：

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter",
    "InstanceType":"t2.micro"
}
```

如果未指定，则参数的默认版本为最新版本。

以下示例引用 *MyParameter* 参数的特定版本。该示例使用 *MyParameter* 参数的版本 *3*，但您可以使用任何有效的版本号。

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:3",
    "InstanceType":"t2.micro"
}
```

以下类似示例引用了映射到 *MyParameter* 参数特定版本的参数标签 *prod*。

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:prod",
    "InstanceType":"t2.micro"
}
```

以下为响应示例。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-00f93d4588example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
```

```
        "CreateTime": "2024-01-08T12:43:21.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

要在启动模板中指定来自 Parameter Store 的参数，您必须拥有指定参数的 `ssm:GetParameters` 权限。使用启动模板的任何人也需要 `ssm:GetParameters` 权限才能验证参数值。有关更多信息，请参阅《Amazon Systems Manager 用户指南》中的使用 IAM 策略限制对 Systems Manager 参数的访问。

验证启动模板是否获得正确的 AMI ID

使用describe-launch-template-versions命令并添加将参数解析为实际的 AMI ID 的`--resolve-alias`选项。

```
aws ec2 describe-launch-template-versions --launch-template-name my-template-for-auto-scaling \
   --versions 1 --resolve-alias
```

该示例返回 `ImageId` 的 AMI ID。使用此启动模板启动实例时，AMI ID 解析为 `ami-0ac394d6a3example`。

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-089c023a30example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreateTime": "2022-12-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-0ac394d6a3example",
                "InstanceType": "t2.micro",
            }
        }
    ]
}
```

相关资源

有关在启动模板中指定 Systems Manager 参数的更多详情，请参阅亚马逊 EC2 用户指南中的使用系统管理器参数代替 AMI ID。

有关使用 Systems Manager 参数的更多信息，请参阅 Systems Manager 文档中的以下参考资料。

- 要创建参数版本和标签，请参阅使用参数版本和使用参数标签。
- 有关如何查找 Amazon 支持的 AMI 公共参数的信息 EC2，请参阅调用 AMI 公共参数。
- 有关与其他 Amazon 账户共享参数或通过共享参数的信息 Amazon Organizations，请参阅使用共享参数。
- 有关监控参数是否成功创建的信息，请参阅 Amazon 系统映像的原生参数支持 IDs。

限制

使用 Systems Manager 参数时，请注意以下限制：

- Amazon A EC2 uto Scaling 仅支持将 AMI 指定 IDs 为参数。
- 不支持使用指定 Systems Manager 参数的启动模板创建或更新具有基于属性的实例类型选择的混合实例组。
- 如果您的自动扩缩组使用指定 Systems Manager 参数的启动模板，则您将无法使用所需的配置或使用跳过匹配开始实例刷新。
- 如果您的 Auto Scaling 组使用的启动模板指定 Systems Manager 参数，则不支持温池。
- 每次调用创建或更新您的 Auto Scaling 群组时，Amazon A EC2 uto Scaling 都会解析启动模板中的 Systems Manager 参数。如果您使用的是高级参数或更高的吞吐量限制，则频繁调用 Parameter Store（即 `GetParameters` 操作）可能会增加 Systems Manager 的成本，因为每次 Parameter Store API 交互都会产生费用。有关更多信息，请参阅 Amazon Systems Manager 定价。

自动扩缩启动配置

Important

限制：

- 自 2023 年 1 月 1 日起，启动配置中不再支持新的 Amazon EC2 实例类型。这包括支持在初始区域启动 Amazon Web Services 区域 后添加到中的任何实例类型。
- 2023 年 6 月 1 日当天或之后创建的账户无法使用控制台创建新的启动配置。
- 2024 年 10 月 1 日当天或之后创建的账户无法使用任何方法（控制台 Amazon CLI、API 或 CloudFormation）创建新的启动配置。

迁移到启动模板，确保现在或将来都不需要创建新的启动配置。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

Note

您也许能够使用某个地区不再支持的实例类型创建启动配置。我们建议您迁移到启动模板。

我们为尚未从启动配置迁移到启动模板的客户提供有关启动配置的信息。启动配置是 Auto Scaling 组用来启动 EC2 实例的实例配置模板。在创建启动配置时，您需要指定实例的信息。包括 Amazon Machine Image (AMI) 的 ID、实例类型、密钥对、一个或多个安全组以及块储存设备映射。如果您之前启动过 EC2 实例，则指定了相同信息以启动该实例。

您可以为多个 Auto Scaling 组指定启动配置。但是一次只能为一个 Auto Scaling 组指定一个启动配置，而且启动配置在创建后不能修改。要更改 Auto Scaling 组的启动配置，必须创建启动配置，然后用该配置更新您的 Auto Scaling 组。

内容

创建启动配置

Important

限制：

- 自 2023 年 1 月 1 日起，启动配置中不再支持新的 Amazon EC2 实例类型。这包括支持在初始区域启动 Amazon Web Services 区域 后添加到中的任何实例类型。
- 2023 年 6 月 1 日当天或之后创建的账户无法使用控制台创建新的启动配置。
- 2024 年 10 月 1 日当天或之后创建的账户无法使用任何方法（控制台 Amazon CLI、API 或 CloudFormation）创建新的启动配置。

迁移到启动模板，确保现在或将来都不需要创建新的启动配置。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

Note

您也许能够使用某个地区不再支持的实例类型创建启动配置。我们建议您迁移到启动模板。

本主题描述了如何创建启动配置。我们为尚未从启动配置迁移到启动模板的客户提供有关启动配置的信息。

创建启动配置后，您无法对其进行修改。相反，必须创建新的启动配置。

要将新的启动配置与现有自动扩缩组相关联，请参阅更改 Auto Scaling 组的启动配置。要创建新自动扩缩组，请参阅使用启动配置创建 Auto Scaling 组。

内容

创建启动配置

创建启动配置（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在顶部导航栏上，选择您所在 Amazon 的地区。
3. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。
4. 在页面顶部附近，选择启动配置。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
5. 选择创建启动配置，然后为您的启动配置输入名称。
6. 对于 Amazon Machine Image (AMI)，选择 AMI。要查找特定 AMI，您可以查找合适的 AMI，记下其 ID，然后输入 ID 作为搜索条件。

   要获取 Amazon Linux 2 AMI 的 ID，请执行以下操作：

   a. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
   b. 在导航窗格中的 实例下，选择 实例，然后选择 启动实例。
   c. 在选择 Amazon Machine Image 页面的 Quick Start 选项卡上，请注意 Amazon Linux 2 AMI (HVM) 旁边的 AMI 的 ID。
7. 对于实例类型，为您的实例选择硬件配置。
8. 在其他配置下，请注意以下字段：

   a. （可选）对于购买选项，您可以选择请求竞价型实例以按需价格为上限的 Spot 价格请求竞价型实例。（可选）您可以指定您的竞价型实例的每实例小时最高价。

   > **Note**
   >
   > 如果您能灵活控制应用程序的运行时间并且应用程序可以中断，那么相对于按需型实例，竞价型实例是经济实惠之选。有关更多信息，请参阅 为容错和灵活的应用程序请求竞价型实例。

   b. （可选）对于 IAM 实例配置文件，选择要与实例关联的角色。有关更多信息，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。
   c. （可选）对于监控，选择是否允许实例每隔 1 分钟向 Amazon 发布指标数据， CloudWatch 方法是启用详细监控。将收取额外费用。有关更多信息，请参阅 配置 Auto Scaling 实例的监控。

d. （可选）对于高级详细信息、用户数据，您可以指定用户数据在启动过程中配置实例或在实例启动后运行配置脚本。

e. （可选）对于高级详细信息、IP 地址类型，选择是否将公有 IP 地址分配到组的实例。如果未设置值，则原定设置为使用启动实例的子网的自动分配公有 IP 设置。

9. （可选）对于存储（卷），如果您不需要额外存储，则可以跳过此部分。否则，除了 AMI 指定的卷以外，要指定要附加到实例的卷，请选择添加新卷。然后选择所需的选项和设备、快照、大小、卷类型、IOPS、吞吐量、终止时删除和已加密的关联值。

10. 对于安全组，创建或选择要与组实例关联的安全组。如果您将创建新安全组选项保留为选中状态，则会为运行 Linux 的 Amazon EC2 实例配置默认 SSH 规则。为运行 Windows 的亚马逊 EC2 实例配置了默认 RDP 规则。

11. 对于密钥对（登录），请选择密钥对选项下的选项。

    如果您已经配置了 Amazon EC2 实例密钥对，则可以在此处进行选择。

    如果您还没有 Amazon EC2 实例密钥对，请选择创建新的密钥对并为其指定一个可识别的名称。选择下载密钥对以将密钥对下载到您的计算机。

    > **Important**
    >
    > 如果需要连接到您的实例，不要选择在没有密钥对的情况下继续。

12. 选中确认复选框，然后选择 Create launch configuration（创建启动配置）。

从现有启动配置创建启动配置（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在顶部导航栏上，选择您所在 Amazon 的地区。
3. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。
4. 在页面顶部附近，选择启动配置。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
5. 选择启动配置，选择 Actions（操作），然后单击 Copy launch configuration（复制启动配置）。这将设置与原启动配置选项相同的新启动配置，但在名称中会增加“Copy”文本。
6. 在 Copy Launch Configuration（复制启动配置）页面上，根据需要编辑配置选项，然后选择 Create launch configuration（创建启动配置）。

使用命令行创建启动配置

您可以使用以下任一命令：

- create-launch-configuration (Amazon CLI)
- 新增-ASLaunch 配置 (Amazon Tools for Windows PowerShell)

配置实例元数据选项

Amazon A EC2 uto Scaling 支持在启动配置中配置实例元数据服务 (IMDS)。这使您可以选择使用启动配置将 Auto Scaling 组中的 Amazon EC2 实例配置为需要实例元数据服务版本 2 (IMDSv2)，这是一种用于请求实例元数据的面向会话的方法。有关优势 IMDSv2的详细信息，请参阅 Amazon 博客上的这篇文章，内容涉及为 EC2 实例元数据服务添加深度防御的增强功能。

您可以将 IMDS 配置为同时支持 IMDSv2 和 IMDSv1 （默认），或者要求使用。 IMDSv2如果您使用 Amazon CLI 或其中一个 SDKs 来配置 IMDS，则必须使用最新版本的 Amazon CLI 或 SDK 才能要求使用。 IMDSv2

您可以针对以下内容配置启动配置：

- 要求在请求实例元数据 IMDSv2 时使用
- 指定 PUT 响应跃点限制
- 关闭对实例元数据的访问

您可以在以下主题中找到有关配置实例元数据服务的更多详细信息：Amazon EC2 用户指南中的配置实例元数据服务。

使用以下步骤在启动配置中配置 IMDS 选项。创建启动配置后，您可以将其与 Auto Scaling 组关联。如果将启动配置与现有 Auto Scaling 组关联，现有启动配置将与 Auto Scaling 组取消关联，并且现有实例需要替换才能使用您在新启动配置中指定的 IMDS 选项。有关更多信息，请参阅 更改 Auto Scaling 组的启动配置。

在启动配置中配置 IMDS（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在顶部导航栏上，选择您所在 Amazon 的地区。
3. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。

4. 在页面顶部附近，选择启动配置。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
5. 选择创建启动配置，然后按照常规方式创建启动配置。包括 Amazon Machine Image (AMI) 的 ID、实例类型和可选的密钥对、一个或多个安全组以及实例的任何其他EBS 卷或实例存储卷。
6. 要为与此启动配置关联的所有实例配置实例元数据选项，请在其他配置中的高级详细信息下，执行以下操作：

   a. 对于元数据可访问，选择是启用还是禁用对实例元数据服务的 HTTP 终端节点的访问。预设情况下，将启用 HTTP 终端节点。如果您选择禁用终端节点，则会关闭对实例元数据的访问。 IMDSv2 只有在启用 HTTP 终端节点时，您才能指定需要的条件。

   b. 对于元数据版本，您可以选择在请求实例元数据时要求使用实例元数据服务版本 2 (IMDSv2)。如果未指定值，则默认为同时支持 IMDSv1 和 IMDSv2。

   c. 对于元数据标记响应跃点限制，您可以为元数据标记设置允许的网络跃点数。如果您未指定值，则原定设置为 1。
7. 完成后，选择创建启动配置。

要要求在启动配置 IMDSv2 中使用，请使用 Amazon CLI

使用以下create-launch-configuration命令，`--metadata-options`将设置为`HttpTokens=required`。在为 `HttpTokens` 指定值时，还必须将 `HttpEndpoint` 设置为已启用。由于元数据检索请求将安全令牌标头设置为必填项，因此在请求实例元数据 IMDSv2 时会选择要求使用。

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imdsv2 \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
      ...
  --metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

关闭对实例元数据的访问

使用以下create-launch-configuration命令关闭对实例元数据的访问。您可以稍后使用modify-instance-metadata-options命令重新开启访问权限。

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imds-disabled \
  --image-id ami-01e24be29428c15b2 \
```

```
--instance-type t2.micro \
    ...
--metadata-options "HttpEndpoint=disabled"
```

使用 EC2 实例创建启动配置

您还可以选择使用正在运行的 EC2 实例的属性来创建启动配置。

从头开始创建启动配置和从现有 EC2 实例创建启动配置有区别。如果从头创建启动配置，需要指定映像 ID、实例类型、可选资源（如存储设备）和可选设置（如监控）。当您从正在运行的实例创建启动配置时，Amazon A EC2 uto Scaling 会从指定实例中获取启动配置的属性。属性也可能来自从中启动实例的 AMI 的块储存设备映射，从而忽略在启动后添加的任何其他块储存设备。

如果使用正在运行的实例创建启动配置，可以通过将以下属性指定为相同请求的一部分，来覆盖这些属性：AMI、块储存设备、密钥对、实例配置文件、实例类型、内核、实例监控、部署租用、虚拟磁盘、安全组、最高 Spot 价格、用户数据、该实例是否有公有 IP 地址，以及该实例是否经过了 EBS 优化。

> **Note**
>
> 如果指定的实例具有启动配置当前不支持的属性，则 Auto Scaling 组启动的实例可能与原始 EC2 实例不同。

> **Important**
>
> 用于启动指定实例的 AMI 必须仍然存在。

主题

- 从 EC2 实例创建启动配置 (Amazon CLI)
- 从实例创建启动配置以及覆盖块储存设备 (Amazon CLI)
- 创建启动配置和覆盖实例类型 (Amazon CLI)

从 EC2 实例创建启动配置 (Amazon CLI)

可以使用以下 create-launch-configuration 命令从一个实例中创建启动配置，它使用与该实例相同的属性。将忽略在启动后添加的任何块储存设备。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance --instance-id i-a8e09d9c
```

您可以使用以下describe-launch-configurations命令来描述启动配置并验证其属性是否与实例的属性相匹配。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance
```

以下为响应示例。

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

从实例创建启动配置以及覆盖块储存设备 (Amazon CLI)

默认情况下，Amazon A EC2 uto Scaling 使用您指定的 EC2 实例中的属性来创建启动配置。不过，块储存设备来自用于启动实例的 AMI，而不是来自实例。要将块储存设备添加到启动配置，请覆盖该启动配置的块储存设备映射。

使用以下create-launch-configuration命令使用 EC2 实例创建启动配置，但使用自定义块储存设备映射。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-bdm --instance-id i-a8e09d9c \
  --block-device-mappings "[{\"DeviceName\":\"/dev/sda1\",\"Ebs\":{\"SnapshotId\":\"snap-3decf207\"}},{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":\"snap-eed6ac86\"}}]"
```

使用以下describe-launch-configurations命令描述启动配置并验证其是否使用您的自定义块储存设备映射。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-bdm
```

下面的示例响应描述了该启动配置。

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-c49c0dac",
            "CreatedTime": "2015-01-07T14:51:26.065Z",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "SnapshotId": "snap-3decf207"
                    }
                },
                {
                    "DeviceName": "/dev/sdf",
                    "Ebs": {
                        "SnapshotId": "snap-eed6ac86"
                    }
                }
            ],
```

```
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8637d3e3"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-bdm",
            "KernelId": null,
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

创建启动配置和覆盖实例类型 (Amazon CLI)

默认情况下，Amazon A EC2 uto Scaling 使用您指定的 EC2 实例中的属性来创建启动配置。根据您的要求，您可能需要覆盖实例中的属性并使用所需的值。例如，您可以覆盖实例类型。

使用以下create-launch-configuration命令使用实例创建启动配置，但 EC2 实例类型（例如`t2.medium`）与实例类型不同（例如`t2.micro`）。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-changetype \
  --instance-id i-a8e09d9c --instance-type t2.medium
```

使用以下describe-launch-configurations命令描述启动配置并验证实例类型是否已被覆盖。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-changetype
```

下面的示例响应描述了该启动配置。

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
```

```
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-changetype",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t2.medium",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

更改 Auto Scaling 组的启动配置

> **Important**
>
> 我们为尚未从启动配置迁移到启动模板的客户提供有关启动配置的信息。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

本主题描述了如何将不同的启动配置与您的自动扩缩组关联。

更改启动配置后，将使用新的配置选项启动所有新实例，但现有实例不受影响。有关更多信息，请参阅更新自动扩缩实例。

要更改 Auto Scaling 组的启动配置（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在详细信息选项卡上，选择启动配置、编辑。
5. 对于启动配置，选择启动配置。
6. 完成后，选择更新。

使用命令行更改自动扩缩组的启动配置

您可以使用以下任一命令：

- update-auto-scaling-group (Amazon CLI)
- 更新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

自动扩缩组

> **Note**
>
> 如果您不熟悉自动扩缩组，则请完成创建您的第一个自动扩缩组教程中的步骤以开始使用，并了解当组中的实例终止时自动扩缩组如何响应。

A uto Scaling 组包含一组 EC2 实例，这些实例被视为用于自动扩展和管理的逻辑分组。Auto Scaling 组还允许您使用 Amazon A EC2 uto Scaling 功能，例如运行状况检查替换和扩展策略。Amazon Auto Scaling 服务的核心功能是保持 Auto Scaling 组中的实例数量和 EC2 自动扩展。

Auto Scaling 组的大小取决于您设置为所需容量的实例数量。您可以通过手动方式或使用自动扩展调整其大小以满足需求。

Auto Scaling 组会首先启动足够实例以达到需要的容量。它通过对组中实例定期执行运行状况检查来保持实例数量。即使某个实例运行状况不佳，Auto Scaling 组也会继续保持固定数量的实例。如果某个实例运行状况不佳，则该组终止运行状况不佳的实例，并启动其他实例来替换它。有关更多信息，请参阅自动扩缩组中实例的运行状况检查。

您可以使用扩展策略动态增加或减少组中的实例数量，以符合不断变化的条件。扩展策略生效时，Auto Scaling 组会在您指定的最小和最大容量值之间调整组的所需容量，并根据需要启动或终止实例。您还可以按计划进行扩展。有关更多信息，请参阅 选择您的扩缩方法。

自动扩缩组可以启动按需型实例和/或竞价型实例。只有当您使用 Auto Scaling 群组时，您才能为 Auto Scaling 群组指定多个购买选项 launch template。 有关更多信息，请参阅Auto Scaling 组具有多个实例类型和购买选项。

竞价型实例可让您以相对于按需价格的大幅折扣访问未使用的 EC2 容量。有关更多信息，请参阅 Amazon EC2 竞价型实例。竞价型实例和按需型实例之间存在着关键区别：

- 竞价型实例的价格因需求而异
- 当竞价型实例的可用性或价格发生变化时，Amazon EC2 可以终止单个竞价型实例

竞价型实例终止后，Auto Scaling 组会尝试启动替代实例来保持该组的所需容量。

实例启动时，如果您指定多个可用区，会为这些可用区分配所需容量。如果发生扩展操作，Amazon A EC2 uto Scaling 会自动在您指定的所有可用区域之间保持平衡。

内容

使用启动模板创建自动扩缩组

如果您已创建启动模板，则可以创建一个 Auto Scaling 组，该组使用启动模板作为其 EC2 实例的配置模板。启动模板可以为实例指定一些信息，例如，AMI ID、实例类型、密钥对、安全组和块储存设备映射。有关创建启动模板的信息，请参阅为 Auto Scaling 组创建启动模板。

您必须具有足够的权限来创建自动扩缩组。您还必须拥有足够的权限才能创建 Amazon A EC2 uto Scaling 用来代表您执行操作的服务相关角色（如果该角色尚不存在）。有关管理员在向您授予权限时可参考的 IAM policy 示例，请参阅 基于身份的策略示例 和 在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况。

内容

使用启动模板创建 Auto Scaling 组

创建 Auto Scaling 组时，必须指定必要的信息，以配置 Amazon EC2 实例、实例的可用区和 VPC 子网、所需容量以及最小和最大容量限制。

要配置由您的 Auto Scaling 组启动的 Amazon EC2 实例，您可以指定启动模板或启动配置。以下过程演示如何使用启动模板创建 Auto Scaling 组。

先决条件

- 你必须已创建启动模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

使用启动模板创建 Auto Scaling 组（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏上，选择与创建启动模板时相同的 Amazon Web Services 区域 模板。
3. 选择 Create an Auto Scaling group (创建 Auto Scaling 组)。
4. 在选择启动模板或配置页面上，执行以下操作：
   a. 在 Auto Scaling 组名称中，输入 Auto Scaling 组的名称。
   b. 对于启动模板，请选择现有启动模板。
   c. 对于 Launch template version（启动模板版本），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。
   d. 验证您的启动模板是否支持您计划使用的所有选项，然后选择下一步。
5. 在选择实例启动选项页面上，如果您没有使用多个实例类型，则可以跳过实例类型要求部分，使用启动模板中指定的 EC2 实例类型。

   如需使用多种实例类型，参阅Auto Scaling 组具有多个实例类型和购买选项。
6. 在 Network (网络) 下，对于 VPC，选择相应的 VPC。必须在您于启动模板中指定的安全组所在的 VPC 中创建 Auto Scaling 组。
7. 对于 (子网) Availability Zones and subnets (可用区和子网)，选择指定 VPC 中的一个或多个子网。可以在多个可用区中使用子网以提供高可用性。有关更多信息，请参阅 选择 VPC 子网时的注意事项。
8. 对于可用区域分发，请选择一种分配策略。有关更多信息，请参阅 Auto Scaling 组可用区域分布。
9. 如果您创建了指定实例类型的启动模板，则可以继续下一步，创建使用启动模板中实例类型的 Auto Scaling 组。

或者，如果在启动模板中未指定实例类型，或者您想使用多种实例类型进行自动扩展，您也可以选择 Override launch template (覆盖启动模板) 选项。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

10. 选择 Next (下一步) 以继续下一步。

    或者，您可接受其余默认值，然后选择 Skip to review (跳到审核)。

11. （可选）在与其他服务集成页面上，配置以下选项，然后选择下一步：

    a. 对于负载平衡，请选择是否将 Auto Scaling 组连接到负载均衡器。有关更多信息，请参阅 Elastic Load Balancing。

    b. 对于 VPC 莱迪思集成选项，请选择是否使用 VPC 莱迪思。有关更多信息，请参阅 使用 VPC Lattice 目标组来管理流量。

    c. 对于 Amazon 应用程序恢复控制器 (ARC) 区域切换，请选中该复选框以启用区域移动。有关更多信息，请参阅 Auto Scaling 组的区域偏移。

       - 如果启用区域移动，则对于健康检查行为，请选择忽略不健康状态或替换不健康状态。有关更多信息，请参阅 Auto Scaling 群组的区域偏移是如何运作的。

    d. 在 “健康检查” 下，对于其他运行状况检查类型，选择开启 Amazon EBS 运行状况检查。有关更多信息，请参阅 使用运行状况检查监控具有受损 Amazon EBS 卷的 Auto Scaling 实例。

    e. 对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

12. （可选）在配置组大小和缩放比例页面上，配置以下选项，然后选择下一步：

    a. 在组大小下，对于所需容量，请输入要启动的实例的初始数量。

    b. 在 “扩展”、“扩展限制” 下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。

    c. 对于自动扩缩，请选择是否要创建目标跟踪扩展策略。您也可以在创建自动扩缩组后再创建此策略。

       如果您选择目标跟踪扩展策略，请按照 创建目标跟踪扩缩策略 中的说明创建策略。

    d. 在实例维护策略下，选择是否要创建实例维护策略。您也可以在创建自动扩缩组后再创建此策略。要创建策略，请按照设置实例维护政策中的指导操作。

e. 在“其他容量设置”的“容量预留”首选项下，选择是否要使用容量预留首选项。有关更多信息，请参阅 使用容量预留在特定可用区中预留容量 。

f. 在“其他设置”的实例缩减保护下，选择是否启用实例缩容保护。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

g. 对于监控，选择是否启用 CloudWatch 群组指标收集。这些指标提供的测量值可以指示潜在的问题，例如终止实例的数量或挂起实例的数量。有关更多信息，请参阅 CloudWatch 监控您的 Auto Scaling 组和实例的指标。

h. 对于默认实例预热，请选择此选项并选择应用程序的预热时间。如果您正在创建具有扩展策略的 Auto Scaling 组，则默认实例预热功能会改进用于动态扩展的 Amazon CloudWatch 指标。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

13. （可选）在添加通知页面上，配置通知，然后选择下一步。有关更多信息，请参阅 亚马逊 Auto Scaling 的亚马 EC2 逊 SNS 通知选项。

14. （可选）在添加标签页面上，选择添加标签，为每个标签提供标签键和值，然后选择下一步。有关更多信息，请参阅 为 Auto Scaling 组和实例添加标签。

15. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

使用命令行创建 Auto Scaling 组

您可以使用以下任一命令：

- create-auto-scaling-group (Amazon CLI)
- 新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

使用亚马逊 EC2 启动向导创建 Auto Scaling 群组

以下过程说明如何使用亚马逊 EC2 控制台中的启动实例向导创建 Auto Scaling 组。该选项自动使用启动实例向导中的特定配置详细信息来填充启动模板。

> **Note**
>
> 该向导不会使用您指定的实例数填充 Auto Scaling 组；它只会使用 Amazon Machine Image (AMI) ID 和实例类型填充启动模板。使用创建 Auto Scaling 组向导以指定要启动的实例数量。AMI 包含配置实例所需的信息。在需要具有相同配置的多个实例时，您可以从单个 AMI 启动多个实例。我们建议使用已在其上安装应用程序的自定义 AMI，以避免在重启属于 Auto Scaling

组的实例时终止实例。要在 Amazon A EC2 uto Scaling 中使用自定义 AMI，您必须先从自定义实例创建 AMI，然后使用 AMI 为您的 Auto Scaling 组创建启动模板。

先决条件

- 您必须已经在计划创建 Auto Scaling 组的 Amazon Web Services 区域 位置创建了自定义 AMI。有关更多信息，请参阅《亚马逊 EC2 用户指南》中的创建 AMI。

使用自定义 AMI 作为模板

在本节中，您将使用 Amazon EC2 启动向导自动填充您的自定义 AMI 的启动模板。或者，要从头开始设置启动模板或要进一步了解可以为启动模板配置的参数，请参阅 创建启动模板（控制台）。

使用自定义 AMI 作为模板

1. 打开亚马逊 EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在屏幕顶部的导航栏上，显示 Amazon Web Services 区域 当前。选择要在其中启动 Auto Scaling 组的区域。
3. 在导航窗格中，选择实例。
4. 选择 Launch instance（启动实例），然后执行以下操作：

   a. 在 Name and tags（名称和标签）下，将 Name（名称）留空。该名称不是用于创建启动模板的数据的一部分。

   b. 在 “应用程序和操作系统映像（Amazon 系统映像）” 下，选择 “浏览更多” AMIs 以浏览完整的 AMI 目录。

   c. 选择 “我的” AMIs，找到您创建的 AMI，然后选择 “选择”。

   d. 在 Instance type（实例类型）下，选择一个实例类型。

   > **Note**
   >
   > 选择创建 AMI 时使用的相同实例类型或更强大的实例类型。

   e. 在屏幕右侧的 Summary（摘要）下，对于 Number of instances（实例的数量），输入一个任意数字。此处输入的数字并不重要。在创建 Auto Scaling 组时，您将会指定要启动的实例数。

在 “实例数” 字段下，将显示一条消息，上面写着 “启动多个实例时，请考虑 EC2 Auto Scaling”。

f. 选择考虑 A EC2 uto Scaling 超链接文本。

g. 在 Launch into Auto Scaling Group（启动至 Auto Scaling 组）确认对话框中，选择 Continue（继续）以转至 Create launch template（创建启动模板）页面，其中已经填充了您在启动实例向导中选择的 AMI 和实例类型。

选择 Continue (继续) 之后，Create launch template (创建启动模板) 页面随即打开。按照此程序结束创建启动模板。

创建启动模板

1. 在 Launch template name and description（启动模板名称和说明）下，输入新启动模板的名称和描述。
2. （可选）在 Key pair (login) [密钥对（登录）] 下，对于 Key pair name（密钥对名称），选择先前创建以在连接到实例时（例如使用 SSH）使用的密钥对的名称。
3. （可选）在 Network settings（网络设置）下，对于 Security groups（安全组），选择一个或多个之前创建的安全组。
4. （可选）在 Configure storage（配置存储）下，更新存储配置。默认存储配置由 AMI 和实例类型确定。
5. 配置完启动模板后，选择 Create launch template (创建启动模板)。
6. 在确认页面上，选择 Create Auto Scaling group (创建 Auto Scaling 组)。

创建 自动扩缩组

> **Note**
>
> 本主题的其余内容介绍创建 Auto Scaling 组的基本步骤。有关可以为 Auto Scaling 组配置的参数的更多说明，请参阅 使用启动模板创建 Auto Scaling 组。

选择 Create Auto Scaling group (创建 Auto Scaling 组) 之后，Create Auto Scaling group (创建 Auto Scaling 组) 向导随即打开。按照程序创建 Auto Scaling 组。

创建 Auto Scaling 组

1. 在 Choose launch template or configuration (选择启动模板或配置) 页面上，输入 Auto Scaling 组的名称。
2. 已经自动选择您创建的启动模板。

   对于 Launch template version（启动模板版本），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。
3. 选择 Next (下一步) 以继续下一步。
4. 在选择实例启动选项页面上，如果您没有使用多个实例类型，则可以跳过实例类型要求部分，使用启动模板中指定的 EC2 实例类型。

   如需使用多种实例类型，参阅Auto Scaling 组具有多个实例类型和购买选项。
5. 在 Network (网络) 下，对于 VPC，选择相应的 VPC。必须在启动模板中指定的安全组所在的 VPC 中创建 Auto Scaling 组。

   > **Tip**
   >
   > 如果您没有在启动模板中指定安全组，则将使用指定 VPC 中的默认安全组启动您的实例。预设情况下，此安全组不允许来自外部网络的入站流量。
6. 对于 (子网) Availability Zones and subnets (可用区和子网)，选择指定 VPC 中的一个或多个子网。
7. 对于可用区域分发，请选择一种分配策略。有关更多信息，请参阅 Auto Scaling 组可用区域分布。
8. 选择 Next（下一步）两次以转至 Configure group size and scaling policies（配置组大小和扩展策略）页面。
9. 在组大小下，指定所需容量（创建自动扩缩组后立即启动的初始实例数量）。
10. 在扩展部分的扩展限制下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
11. 选择 Skip to review (跳转以查看)。
12. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

后续步骤

您可以通过查看活动历史记录来检查是否已正确创建了 Auto Scaling 组。在 Activity (活动) 选项卡上的 Activity history (活动历史记录) 下，Status (状态) 列显示您的 Auto Scaling 组是否已成功启动实例。如果实例无法启动，或者它们启动但随后立即终止，请参阅以下主题了解可能的原因和解决方法：

- 对 Amazon A EC2 uto Scaling 进行故障排除： EC2 实例启动失败
- 对 Amazon A EC2 uto Scaling 进行故障排除：AMI 问题
- 对 Amazon A EC2 uto Scaling 中运行不正常的实例进行故障排除

如果需要，您现在可以在 Auto Scaling 组所在的同一区域中连接负载均衡器。有关更多信息，请参阅使用 Elastic Load Balancing 在 Auto Scaling 组中分配传入的应用程序流量 。

Auto Scaling 组具有多个实例类型和购买选项

您可以启动并自动扩展单个 Auto Scaling 组中的一组按需实例和竞价型实例。除了享受使用竞价型实例的折扣外，您还可以使用预留实例或节省计划获得常规按需型实例定价的折扣。这些因素可以帮助您优化 EC2 实例的成本节约，并为您的应用程序获得所需的规模和性能。

竞价型实例是备用容量，与 EC2 按需价格相比，折扣幅度很大。如果能灵活控制应用程序的运行时间并且应用程序可以中断，竞价型实例就是经济实惠之选。这些实例可用于各种容错和灵活的应用程序。示例包括无状态 Web 服务器、API 端点、大数据和分析应用程序、容器化工作负载、CI/CD pipelines, high performance and high throughput computing (HPC/HTC)、渲染工作负载以及其他灵活的工作负载。

有关更多信息，请参阅 Amazon EC2 用户指南中的实例购买选项。

主题

- 创建混合实例组的设置概述
- 多种实例类型的分配策略
- 使用基于属性的实例类型选择创建混合实例组
- 通过手动选择实例类型来创建混合实例组
- 配置自动扩缩组以使用实例权重
- 为实例类型使作不同的启动模板

创建混合实例组的设置概述

本主题提供创建混合实例自动扩缩的概述和最佳实践。

内容

概览

要创建混合实例组，您有两个选项：

- 基于属性的实例类型选择：定义您的计算要求，以便根据其特定实例属性自动选择实例类型。
- 手动实例类型选择：手动选择适合您工作负载的实例类型。

Manual selection

以下步骤介绍如何通过手动选择实例类型来创建混合实例组：

1. 选择具有启动 EC2 实例的参数的启动模板。启动模板中的参数是可选的，但是如果启动模板中缺少亚马逊系统映像 (AMI) ID，Amazon A EC2 uto Scaling 将无法启动实例。
2. 选择选项以覆盖启动模板。
3. 手动选择适合您的工作负载的实例类型。
4. 指定要启动的按需型实例和竞价型实例的百分比。
5. 从可能的实例类型中选择分配策略，以确定 Amazon A EC2 uto Scaling 如何满足您的按需和竞价容量。
6. 选择要在其中启动实例的可用区和 VPC 子网。
7. 指定组的初始大小（所需容量）以及该组的最小和最大大小。

必须进行覆盖，才能覆盖启动模板中声明的实例类型并使用嵌入在自动扩缩组自有资源定义中的多个实例类型。有关可用实例类型的更多信息，请参阅 Amazon EC2 用户指南中的实例类型。

您还可以为每种实例类型配置以下可选参数：

- `LaunchTemplateSpecification`：您可以根据需要为实例类型分配不同的启动模板。此选项目前在控制台中不可用。有关更多信息，请参阅 为实例类型使作不同的启动模板。
- `WeightedCapacity`：与组中其余实例相比，您可以决定该实例在所需容量中所占的比例。如果您为一种实例类型指定 `WeightedCapacity` 值，则必须为所有实例类型指定 `WeightedCapacity` 值。默认情况下，每个实例在您的所需容量中计为一个。有关更多信息，请参阅 配置自动扩缩组以使用实例权重。

Attribute-based selection

要让 Amazon A EC2 uto Scaling 根据实例的特定属性自动选择您的实例类型，请使用以下步骤通过指定您的计算要求来创建混合实例组：

1. 选择具有启动 EC2 实例的参数的启动模板。启动模板中的参数是可选的，但是如果启动模板中缺少亚马逊系统映像 (AMI) ID，Amazon A EC2 uto Scaling 将无法启动实例。
2. 选择选项以覆盖启动模板。
3. 指定与您的计算要求相匹配的实例属性，例如 v CPUs 和内存要求。
4. 指定要启动的按需型实例和竞价型实例的百分比。
5. 从可能的实例类型中选择分配策略，以确定 Amazon A EC2 uto Scaling 如何满足您的按需和竞价容量。
6. 选择要在其中启动实例的可用区和 VPC 子网。
7. 指定组的初始大小（所需容量）以及该组的最小和最大大小。

必须进行覆盖，才能覆盖启动模板中声明的实例类型并使用一组实例属性来描述您的计算要求。有关支持的属性，请参阅《Amazon A EC2 uto Scaling API 参考》InstanceRequirements中的。或者，您可以使用已有实例属性定义的启动模板。

您也可以在覆盖结构中配置 `LaunchTemplateSpecification` 参数，以便根据需要为一组实例要求分配不同的启动模板。此选项目前在控制台中不可用。有关更多信息，请参阅LaunchTemplateOverrides《Amazon A EC2 uto Scaling API 参考》。

默认情况下，您将自动扩缩组的所需容量值设置为实例的数量。

或者，您可以将所需容量的值设置为 v 数CPUs 或内存量。为此，请使用 `CreateAutoScalingGroup` API 操作中的 `DesiredCapacityType` 属性或 Amazon Web Services Management Console中的所需容量类型下拉字段。这是实例权重的实用替代方案。

实例类型灵活性

要提高可用性，请跨多种实例类型部署应用程序。最佳实践是使用多种实例类型来满足容量要求。这样，如果您选择的可用区域中的实例容量不足，Amazon A EC2 uto Scaling 就可以启动另一种实例类型。

如果竞价型实例的实例容量不足，Amazon A EC2 uto Scaling 会继续尝试从其他竞价型实例池启动。（它使用的池由您选择的实例类型和分配策略决定。）Amazon A EC2 uto Scaling 通过启动竞价型实例代替按需实例，帮助您利用竞价型实例节省的成本。

我们建议灵活地为每种工作负载在至少 10 种实例类型之间进行选择。选择实例类型时，不要局限于最热门的新实例类型。选择老一代实例类型往往会导致竞价型中断较少，因为按需客户对它们的需求较少。

可用区灵活性

我们强烈推荐您在多个可用区之间跨越您的自动扩缩组。利用多个可用区，您可以设计在可用区之间自动实现故障转移的应用程序，从而提高弹性。

此外，与单个可用区中的群组相比，您可以访问更深的 Amazon EC2 容量池。由于可用区内每个实例类型的容量分开波动，您可以灵活使用实例类型和可用区，通常可以获得更多计算容量。

有关使用多可用区的详细信息，请参阅 示例：在可用区之间分配实例。

Spot 最高价

当你使用 Amazon CLI 或软件开发工具包创建 Auto Scaling 组时，你可以指定`SpotMaxPrice`参数。`SpotMaxPrice` 参数确定您愿意为竞价型实例一小时支付的最高价。

当您在覆盖（或 `"DesiredCapacityType": "vcpu"` 或在组级别的 `"DesiredCapacityType": "memory-mib"`）中指定 `WeightedCapacity` 参数时，最高价格代表的是最高单价，而不是整个实例的最高价格。

我们强烈建议您不要指定最高价。如果您未收到任何竞价型实例 (例如当您的最高价太低时)，您的应用程序可能不运行。如果未指定最高价，则默认最高价为按需价格。您只需为您启动的竞价型实例支付 Spot 价格。您仍然可以享受竞价型实例提供的大幅折扣。这些折扣得以实现，是由于借助竞价型定价模型带来了稳定的竞价型定价。有关更多信息，请参阅 Amazon EC2 用户指南中的定价和优惠。

主动容量再平衡

如果您的用例允许，我们建议您进行容量再平衡。容量再平衡通过主动替换存在中断风险的竞价型实例来帮助您保持工作负载的可用性。

启用容量再平衡后，Amazon A EC2 uto Scaling 会尝试主动替换已收到 EC2 实例再平衡建议的竞价型实例。这提供了一个机会，可以将您的工作负载重新平衡到中断风险不高的新 Spot 实例。

有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

扩展行为

当您创建混合实例组时，它默认使用按需型实例。要使用竞价型实例，您必须修改要启动的按需型实例启动的百分比。您可以指定 0 到 100 之间的任意数字来作为按需百分比。

或者，您也可以指定要开始使用的按需型实例的基本数量。如果您这样做，Amazon A EC2 uto Scaling 会等待竞价型实例的启动，直到按需实例组扩展时启动按需实例的基本容量。超出基本容量的任何内容都使用按需百分比来确定要启动多少按需型实例和竞价型实例。

Amazon A EC2 uto Scaling 将百分比转换为等值的实例数。如果结果创建一个小数，向上舍入为下一个整数，以支持按需型实例。

下表展示了自动扩缩组随着规模的增加和减少而采取的行为。

示例：扩展行为

购买选项	各购买选项的运行实例的组大小和数量			
	10	20	30	40
示例 1：以 10 为基准，50/ 50% 按需/竞价型				
按需实例（基本金额）	10	10	10	10
按需型实例	0	5	10	15
竞价型实例	0	5	10	15

购买选项	各购买选项的运行实例的组大小和数量			
示例 2：以 0 为基准，0/ 100% 按需/竞价型				
按需实例（基本金额）	0	0	0	0
按需型实例	0	0	0	0
竞价型实例	10	20	30	40
示例 3：以 0 为基准，60/ 40% 按需/竞价型				
按需实例（基本金额）	0	0	0	0
按需型实例	6	12	18	24
竞价型实例	4	8	12	16
示例 4：以 0 为基准，100/ 0% 按需/竞价型				
按需实例（基本金额）	0	0	0	0
按需型实例	10	20	30	40
竞价型实例	0	0	0	0
示例 5：以 12 为基准，0/ 100% 按需/竞价型				

购买选项	各购买选项的运行实例的组大小和数量			
按需实例（基本金额）	10	12	12	12
按需型实例	0	0	0	0
竞价型实例	0	8	18	28

当群组规模增加时，Amazon A EC2 uto Scaling 会尝试在您指定的可用区域内均衡您的容量。然后，它根据指定的分配策略启动实例类型。

当群组规模缩小时，Amazon A EC2 uto Scaling 会首先确定应终止两种类型（竞价型或按需型）中的哪一种。然后，它会尝试以平衡的方式终止您指定的可用区的实例。它还倾向于以更接近您的分配策略的方式终止实例。有关终止策略的更多信息，请参阅 为 Amazon A EC2 uto Scaling 配置终止策略。

实例类型的区域可用性

EC2 实例类型的可用性因您而异 Amazon Web Services 区域。例如，最新一代实例类型可能尚未在给定区域中可用。由于不同区域的实例可用性存在差异，如果您的覆盖中的多个实例类型在您所在的地区不可用，则在提出编程请求时可能会遇到问题。使用您所在地区不可用的多个实例类型可能会导致请求完全失败。要解决此问题，请使用不同的实例类型重试请求，确保每个实例类型在该区域中都可用。要搜索按位置提供的实例类型，请使用describe-instance-type-offerings命令。有关更多信息，请参阅亚马逊 EC2 用户指南中的查找亚马逊 EC2 实例类型。

相关资源

有关竞价型实例的更多最佳实践，请参阅 Amazon EC2 用户指南中的 EC2 竞价型最佳实践。

限制

使用混合实例策略向自动扩缩组添加覆盖后，您可以通过 UpdateAutoScalingGroup API 调用更新覆盖，但无法将其删除。要完全移除覆盖，必须先将自动扩缩组切换为使用启动模板或启动配置，而不是混合实例策略。然后，您可以再次添加混合实例策略，而无需任何覆盖。

多种实例类型的分配策略

当您使用多种实例类型时，您可以根据可能的实例类型管理 Amazon A EC2 uto Scaling 如何满足您的按需容量和竞价。为此，您需要指定多个分配策略。

要查看混合实例组的最佳实践，请参阅创建混合实例组的设置概述。

内容

竞价型实例

Amazon A EC2 uto Scaling 为竞价型实例提供了以下分配策略：

`price-capacity-optimized`（推荐）

价格和容量优化分配策略同时考虑价格和容量，以选择中断可能性最小、价格尽可能低的竞价型实例池。

我们建议您在入门时使用此策略。有关更多信息，请参阅 Amazon 博客中的 EC2 竞价型实例 price-capacity-optimized分配策略简介。

`capacity-optimized`

Amazon A EC2 uto Scaling 从池中请求您的竞价型实例，其容量最适合正在启动的实例数量。

使用竞价型实例，定价会根据长期供需趋势缓慢发生变化。但是，容量会实时波动。`capacity-optimized` 策略通过查看实时容量数据并预测可用性最高的池，自动在可用性最高的池中启动 Spot 实例。这有助于最大限度减少工作负载的可能中断，这些工作负载可能会因重启工作和检查点操作而导致更高的中断成本。要使某些实例类型有更高的首先启动机会，请使用 `capacity-optimized-prioritized`。

`capacity-optimized-prioritized`

设置启动模板覆盖的实例类型顺序，从最高优先级到最低优先级（从列表中的第一项到最后一项降序列出）。Amazon A EC2 uto Scaling 会尽最大努力尊重实例类型优先级，但会首先针对容量进行优化。对于必须最大限度地减少中断可能性，但对某些实例类型的偏好也很重要的工作负载来说，这是一个不错的选择。如果将按需型分配策略设置为 `prioritized`，则在满足按需型容量时将应用相同的优先级。

`lowest-price`（不推荐）

Amazon A EC2 uto Scaling 在您为最低价格池设置指定的 N 个竞价池中，使用可用区内价格最低的池请求您的竞价型实例。例如，如果您指定四种实例类型和四个可用区，则您的自动扩缩组可以

访问最多 16 个竞价型池。（每个可用区内四个。）如果您为分配策略指定两个竞价型池 (N=2)，则您的自动扩缩组可以使用每个可用区的两个价格最低的池来满足竞价型容量。

由于此策略仅考虑实例价格而不考虑容量可用性，因此可能会导致较高的中断率。

Amazon A EC2 uto Scaling 会努力从您指定的 N 个池中提取竞价型实例。但是，如果池在满足所需容量之前用完竞价容量，Amazon A EC2 uto Scaling 会继续通过从价格第二低的池中提取资金来满足您的请求。为了达到您所需的容量，您可能会从超过指定的 N 个池接收竞价型实例。同样，如果大多数池没有竞价容量，则您可能会从少于指定的 N 个池接收完整所需容量。

> **Note**
>
> 如果将竞价型实例配置为启动并开启 AMD SEV-SNP，您需要按小时支付额外的使用费，费率为所选实例类型按需小时费率的 10%。如果分配策略使用价格作为输入，则 Amazon A EC2 uto Scaling 不包括这笔额外费用；只使用现货价格。

按需型实例

Amazon A EC2 uto Scaling 提供了以下可用于按需实例的分配策略：

`lowest-price`

Amazon A EC2 uto Scaling 会根据当前的按需价格自动在每个可用区部署价格最低的实例类型。

为了满足您的所需容量，您可能会在每个可用区中获得一种以上实例类型的按需型实例。这取决于您请求多大容量。

`prioritized`

在满足按需容量时，Amazon A EC2 uto Scaling 会根据启动模板替代列表中实例类型的顺序确定首先使用哪种实例类型。例如，假设您按以下顺序指定三个启动模板覆盖：`c5.large`、`c4.large` 和 `c3.large`。在您的按需型实例启动时，自动扩缩组满足按需型容量的顺序是从 `c5.large` 开始，然后是 `c4.large`，最后是 `c3.large`。

管理按需实例的优先级顺序时，请考虑以下事项：

- 您可以通过使用 Savings Plans 或预留实例预先支付使用费用，以获得按需实例的大幅折扣。有关更多信息，请参阅 Amazon EC2 定价页面。
- 对于预留实例，如果 Amazon A EC2 uto Scaling 启动匹配的实例类型，则按常规按需实例定价的折扣费率适用。因此，如果存在未使用的 `c4.large` 预留实例，则可以设置实例类型优先级，

将预留实例的最高优先级赋予 c4.large 实例类型。当 c4.large 实例启动时，您可享受预留实例定价。

- 使用 Savings Plans，在使用亚马逊实例储蓄计划或计算储蓄计划时，您的常规按需 EC2实例定价折扣费率适用。通过 Savings Plans，您可以更灵活地确定实例类型的优先级。只要您使用 Savings Plan 涵盖的实例类型，就可以按任何优先级顺序设置它们。您也可以偶尔更改实例类型的整个顺序，同时仍可享受 Savings Plan 折扣费率。有关 Savings Plans 的更多信息，请参阅 Savings Plans 用户指南。

分配策略如何与权重配合使用

当您在覆盖（或在组级别的 "DesiredCapacityType": "vcpu" 或 "DesiredCapacityType": "memory-mib"）中指定 WeightedCapacity 参数时，分配策略的工作方式与用于其他自动扩缩组时完全相同。

假设您有一个 Auto Scaling 组，该组包含多个具有不同数量的 v 的实例类型CPUs。 您使用lowest-price现货和按需分配策略。如果您选择根据每种实例类型的 vCPU 数量分配权重，Amazon A EC2 uto Scaling 将在配送时根据您分配的权重值（例如，每个 vCPU）启动价格最低的实例类型。如果是 Spot 实例，那么这意味着每个 vCPU 的最低 Spot 价格。如果是按需型实例，那么这意味着每个 vCPU 的最低按需价格。

有关更多信息，请参阅 配置自动扩缩组以使用实例权重。

使用基于属性的实例类型选择创建混合实例组

在创建混合实例组时手动选择实例类型有一个替代方法，您可以指定一组用于描述计算要求的实例属性。当 Amazon A EC2 uto Scaling 启动实例时，Auto Scaling 组使用的任何实例类型都必须与您所需的实例属性相匹配。这称为基于属性的实例类型选择。

这种方法非常适合于可以灵活处理所使用实例类型的工作负载和框架，例如容器、大数据和 CI/CD。

选择基于属性的实例类型具有以下优势：

- 竞价型实例的最佳灵活性 — Amazon A EC2 uto Scaling 可以从多种实例类型中进行选择，用于启动竞价型实例。这符合灵活处理实例类型的 Spot 最佳实践，这使得 Amazon EC2 Spot 服务更有可能找到和分配所需的计算容量。
- 轻松使用正确的实例类型 – 有如此多的实例类型可供使用，因此找到适用于您的工作负载的实例类型可能非常耗时。当您指定实例属性时，实例类型将自动具有工作负载所需的属性。
- 自动使用新实例类型：您的自动扩缩组可以在发布新一代实例类型时使用这些实例类型。新一代实例类型将自动投入使用，如果它们符合您的要求并与您为 Auto Scaling 组选择的分配策略保持一致。

主题

基于属性的实例类型选择的工作原理

通过基于属性的实例类型选择，您无需提供特定实例类型列表，而是提供实例所需的实例属性列表，例如：

- vCPU 计数-每个实例的最小和最大 v CPUs 数。
- 内存-每个实例的最小 GiBs 和最大内存。
- 本地存储 - 是使用 EBS 还是实例存储卷作为本地存储。
- 可突增性能 - 是否使用 T 实例系列（包括 T4g、T3a、T3 和 T2 类型）。

有许多选项可用于定义您的实例要求。有关每个选项和默认值的描述，请参阅InstanceRequirements《Amazon A EC2 uto Scaling API 参考》。

当您的自动扩缩组需要启动实例时，它将搜索与您指定的属性匹配且在该可用区中可用的实例类型。然后，分配策略确定启动哪种匹配的实例类型。默认情况下，基于属性的实例类型选择启用了价格保护功能，以防止您的自动扩缩组启动超出预算阈值的实例类型。

默认情况下，在设置自动扩缩组所需的容量时，使用实例数作为计量单位，这意味着每个实例算作一个单位。

或者，您可以将所需容量的值设置为 v 数CPUs 或内存量。为此，请在或 UpdateAutoScalingGroup API 操作中使用 Amazon Web Services Management Console 或DesiredCapacityType属性的所需容量类型下拉字段。CreateAutoScalingGroup然后，Amazon A EC2 uto Scaling 会启动满足所需的 vCPU 或内存容量所需的实例数量。例如，如果您

使用 v CPUs 作为所需容量类型，并使用CPUs 每个容量为 2 v 的实例，则所需容量为 10 v CPUs 将启动 5 个实例。这是实例权重的实用替代方案。

价格保护

借助价格保护，您可以指定愿意为 Auto Scaling 组启动的 EC2 实例支付的最高价格。价格保护是一项功能，可以防止自动扩缩组使用您认为成本过高的实例类型，即使其恰好适合您指定的属性。

默认已启用价格保护，价格保护为按需型实例和竞价型实例设置了单独的价格阈值。当 Amazon A EC2 uto Scaling 需要启动新实例时，任何定价高于相关阈值的实例类型都不会启动。

主题

- 按需价格保护
- 竞价价格保护
- 自定义价格保护

按需价格保护

对于按需型实例，您可以将愿意支付的最高按需价格定义为高于已确定按需价格的百分比。已确定按需价格是具有指定属性的、价格最低的最新一代 C、M 或 R 实例类型的价格。

如果未明确定义按需价格保护值，则将使用比已确定按需价格高 20% 的默认最高按需价格。

竞价价格保护

默认情况下，Amazon A EC2 uto Scaling 将自动应用最佳竞价型实例价格保护，以便始终如一地从各种实例类型中进行选择。您也可以自行手动设置价格保护。但是，让 Amazon A EC2 uto Scaling 帮你做这件事可以提高竞价容量得到满足的可能性。

您可以使用以下选项之一，手动指定价格保护。如果您手动设置价格保护，则建议您使用第一个选项。

- 已确定按需价格的百分比：已确定按需价格是具有指定属性的、价格最低的最新一代 C、M 或 R 实例类型的价格。
- 高于已确定竞价价格的百分比：已确定竞价价格是具有指定属性的、价格最低的最新一代 C、M 或 R 实例类型的价格。不建议使用此选项，因为竞价价格可能会波动，因此您的价格保护阈值也可能会波动。

自定义价格保护

您可以在 Amazon A EC2 uto Scaling 控制台中或使用或自定义价格保护阈值 SDKs。 Amazon CLI

- 在控制台中，使用其他实例属性中的按需价格保护和竞价价格保护设置。
- 在InstanceRequirements结构中，要指定按需实例价格保护阈值，请使用OnDemandMaxPricePercentageOverLowestPrice属性。要指定竞价型实例价格保护阈值，请使用 MaxSpotPriceAsPercentageOfOptimalOnDemandPrice 或 SpotMaxPricePercentageOverLowestPrice 属性。

如果您将 “所需容量类型” (DesiredCapacityType) 设置为 v CPUs 或内存 GiB，则价格保护将根据每个 vCPU 或每个内存的价格而非每个实例的价格适用。

您也可以关闭价格保护。要表明无价格保护阈值，请指定一个较高的百分比值，如 999999。

> **Note**
>
> 如果没有与指定属性匹配的最新一代 C、M 或 R 实例类型，则价格保护仍然适用。如果找不到匹配项，则已确定价格来自匹配指定属性的、价格最低的最新一代实例类型；如果没有，则来自价格最低的上一代实例类型。

性能保护

性能保护是一项功能，可确保您的 Auto Scaling 组使用的实例类型与指定性能基准相似或超过指定性能基准。要使用性能保护，请指定一个实例系列作为基准参考。指定的实例系列的功能确立了可接受的最低性能水平。Auto Scaling 在选择实例类型时，会考虑您指定的属性和性能基准。低于性能基准的实例类型将被自动排除在选择范围之外，即使它们与您指定的其他属性相匹配。这可确保所有选定的实例类型都提供与指定实例系列建立的基准相似或更好的性能。Auto Scaling 使用此基准来指导实例类型选择，但不能保证所选实例类型始终会超过每个应用程序的基准。

目前，此功能仅支持将 CPU 性能作为基准性能因素。指定实例系列的 CPU 性能作为性能基准，可确保所选实例类型与该基准相似或超过该基准。具有相同 CPU 处理器的实例系列会产生相同的筛选结果，即使其网络或磁盘性能有所不同。例如，将 c6in 或 c6i 指定为基准参考将产生相同的基于性能的筛选结果，因为两个实例系列使用相同的 CPU 处理器。

不支持的实例系列

以下实例系列不支持性能保护：

- c1
- g3 | g3s

- hpc7g
- m1 | m2
- mac1 | mac2 | mac2-m1ultra | mac2-m2 | mac2-m2pro
- p3dn | p4d | p5
- t1
- u-12tb1 | u-18tb1 | u-24tb1 | u-3tb1 | u-6tb1 | u-9tb1 | u7i-12tb | u7in-16tb | u7in-24tb | u7in-32tb

如果您通过指定支持的实例系列来启用性能保护，则返回的实例类型将排除上述不受支持的实例系列。

示例：设置 CPU 性能基准

在以下示例中，实例要求是使用具有与 c6i 实例系列一样性能的 CPU 核心的实例类型启动。这将筛选出性能较低的 CPU 处理器的实例类型，即使它们满足其他指定的实例要求，例如 v CPUs 的数量。例如，如果您指定的实例属性包括 4 v CPUs 和 16 GB 的内存，则具有这些属性但 CPU 性能低于该值的实例类型c6i将被排除在选择范围之外。

```
"BaselinePerformanceFactors": {
        "Cpu": {
            "References": [
                {
                    "InstanceFamily": "c6i"
                }
            ]
        }
```

注意事项

使用性能保护时，请考虑以下几点：

- 您可以指定实例类型或实例属性，但不能同时指定两者。
- 在一个请求配置中，最多可以指定四个 InstanceRequirements 结构。

先决条件

- 创建启动模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。
- 验证启动模板尚未请求竞价型实例。

使用基于属性的实例类型选择创建一个混合实例组（控制台）

通过以下过程使用基于属性的实例类型选择创建混合实例组。为了帮助您高效地完成这些步骤，我们跳过了一些可选部分。

对于大多数通用工作负载，只需指定所需的 v 数量CPUs 和内存就足够了。对于高级使用案例，您可以指定存储类型、网络接口、CPU 制造商和加速器类型等属性。

要查看混合实例组的最佳实践，请参阅创建混合实例组的设置概述。

创建一个混合实例组

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择在创建启动模板时使用的同一 Amazon Web Services 区域 。
3. 选择 Create an Auto Scaling group (创建 Auto Scaling 组)。
4. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，输入 Auto Scaling 组的名称。
5. 要选择启动模板，请执行以下操作：

   a. 对于启动模板，请选择现有启动模板。

   b. 对于 Launch template version（启动模板版本），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。

   c. 验证您的启动模板是否支持您计划使用的所有选项，然后选择下一步。
6. 在选择实例启动选项页面上，执行以下操作：

   a. 对于 Instance type requirements（实例类型要求），请选择 Override launch template（覆盖启动模板）。

   > **Note**
   >
   > 如果您选择的启动模板已经包含一组实例属性（例如 v CPUs 和内存），则会显示实例属性。这些属性已添加到 Auto Scaling 组属性中，您可以随时从 Amazon A EC2 uto Scaling 控制台对其进行更新。

   b. 在 “指定实例属性” 下，首先输入您的 v CPUs 和内存要求。

   - 对于 v CPUs，输入所需的最小值和最大值 v CPUs。 要指定无限制，请选择 “无最小值”、“无最大值” 或两者兼而有之。

- 对于 Memory (GiB)（内存 (GiB)），输入所需的最小和最大内存量。要指定没有限制，请选择 No minimum（没有最小值）和/或 No maximum（没有最大值）。

c. （可选）对于 Additional instance attributes（其它实例属性），您可以选择指定一个或多个属性以更详细地表达计算要求。每个额外属性都会进一步增加对您的请求的限制。

d. （可选）展开预览匹配的实例类型以查看具有指定属性的实例类型。

e. 在 实例购买选项下，对于 实例分配，指定要启动的按需型实例和竞价型实例在该组中所占的百分比。如果您的应用程序无状态、容错，并且可以处理中断的实例，则可以指定更高的竞价型实例百分比。

f. （可选）如果您指定了一个竞价型实例百分比，请选中包括按需型基本容量旁边的复选框，然后指定按需型实例在自动扩缩组的最小初始容量中必须达到的数量。超出基本容量的任何内容都使用百分比来确定要启动多少按需实例和 Spot 实例。

g. 在 Allocation strategies（分配策略）下，已为 On-Demand allocation strategy（按需分配策略）自动选择 Lowest price（最低价格），无法对其进行更改。

h. 对于 Spot allocation strategy（竞价型分配策略），选择分配策略。默认情况下，将选择 Price capacity optimized（价格容量优化）。默认情况下，将隐藏 Lowest price（最低价格），仅在您选择 Show all strategies（显示所有策略）时才会出现。如果您选择最低价格，则请输入价格最低的池的数量，以跨价格最低的池实现多样化。

i. 对于容量再平衡，选择启用还是禁用容量再平衡。使用容量再平衡功能，以自动响应竞价型实例因竞价中断而即将终止的情况。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

j. 在 Network (网络) 下，对于 VPC，选择相应的 VPC。自动扩缩组必须与您在启动模板中指定的安全组创建在相同的 VPC 中。

k. 对于 Availability Zones and subnets (可用区和子网)，选择指定 VPC 中的一个或多个子网。可以在多个可用区中使用子网以提供高可用性。有关更多信息，请参阅 选择 VPC 子网时的注意事项。

l. 选择下一步、下一步。

7. 对于 Configure group size and scaling policies（配置组大小和扩缩策略）步骤，请执行以下操作：

a. 如果您希望以实例以外的其他单位来衡量所需容量，请为组大小、所需容量类型选择合适的选项。支持单位CPUs、v 和内存 GiB。默认情况下，Amazon A EC2 uto Scaling 会指定单位，单位转换为实例的数量。

b. 为自动扩缩组设置初始 所需容量。

c. 在扩展部分的扩展限制下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。有关更多信息，请参阅为自动扩缩组设置扩缩限制。

8. 选择 Skip to review (跳转以查看)。
9. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

使用基于属性的实例类型选择创建一个混合实例组（Amazon CLI）

使用命令行创建一个混合实例组

使用以下命令之一：

- create-auto-scaling-group (Amazon CLI)
- 新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

示例配置

要使用创建具有基于属性的实例类型选择的 Auto Scaling 组 Amazon CLI，请使用以下create-auto-scaling-group命令。

指定了以下实例属性：

- VCpuCount— 实例类型必须最小为四 vCPUs，最大为八 v CPUs。
- MemoryMiB – 实例类型的最小内存必须为 16384 MiB。
- CpuManufacturers - 实例类型必须具有英特尔制造的 CPU。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

下面是一个 config.json 示例文件。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacityType": "units",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
```

```
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [{
                "InstanceRequirements": {
                    "VCpuCount": {"Min": 4, "Max": 8},
                    "MemoryMiB": {"Min": 16384},
                    "CpuManufacturers": ["intel"]
                }
            }]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 4,
    "DesiredCapacityType": "units",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

要将所需容量的值设置为 v 数CPUs 或内存量，请在文件"DesiredCapacityType": "memory-mib"中指定"DesiredCapacityType": "vcpu"或。默认的所需容量类型为 units，它将所需容量的值设置为实例数。

YAML

或者，您可以使用以下create-auto-scaling-group命令创建 Auto Scaling 组。这将引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

下面是一个 config.yaml 示例文件。

```
---
AutoScalingGroupName: my-asg
DesiredCapacityType: units
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
```

```
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceRequirements:
        VCpuCount:
          Min: 2
          Max: 4
        MemoryMiB:
          Min: 2048
        CpuManufacturers:
        - intel
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 0
MaxSize: 100
DesiredCapacity: 4
DesiredCapacityType: units
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

要将所需容量的值设置为 v 数CPUs 或内存量，请在文件DesiredCapacityType: memory-mib中指定DesiredCapacityType: vcpu或。默认的所需容量类型为 units，它将所需容量的值设置为实例数。

预览您的实例类型

您可以预览符合计算要求的实例类型而无需启动它们，并在必要时调整要求。在 Amazon Auto Scaling 控制台中创建 A EC2 uto Scaling 组时，实例类型的预览会显示在选择实例启动选项页面的预览匹配的实例类型部分。

或者，您可以使用 Amazon CLI 或软件开发工具包调用 Amazon EC2 GetInstanceTypesFromInstanceRequirementsAPI 来预览实例类型。在请求中传递 InstanceRequirements 参数，其格式与用于创建或更新 Auto Scaling 组的格式一致。有关更多信息，请参阅 Amazon EC2 用户指南中的预览具有指定属性的实例类型。

相关资源

要详细了解基于属性的实例类型选择，请参阅博客上的 Auto Scaling EC2 和 EC2 Fleet 基于属性的实例类型选择。 Amazon

当您使用 Amazon CloudFormation创建一个自动缩放组时，您可以声明基于属性的实例类型选择。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的 自动扩缩模板片段部分中的示例片段。

通过手动选择实例类型来创建混合实例组

本主题介绍如何通过手动选择实例类型在单个自动扩缩组中启动多种实例类型。

如果您希望使用实例属性作为选择实例类型的标准，请参阅 使用基于属性的实例类型选择创建混合实例组。

内容

前提条件

- 创建启动模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。
- 验证启动模板尚未请求竞价型实例。

创建混合实例组（控制台）

使用以下过程创建混合实例组，手动选择您的组可以启动的实例类型。为了帮助您高效地完成这些步骤，我们跳过了一些可选部分。

要查看混合实例组的最佳实践，请参阅创建混合实例组的设置概述。

创建一个混合实例组

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择在创建启动模板时使用的同一 Amazon Web Services 区域 。
3. 选择 Create an Auto Scaling group (创建 Auto Scaling 组)。
4. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，输入 Auto Scaling 组的名称。
5. 要选择启动模板，请执行以下操作：

    a. 对于启动模板，请选择现有启动模板。

    b. 对于 Launch template version（启动模板版本），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。

c. 验证您的启动模板是否支持您计划使用的所有选项，然后选择 Next（下一步）。

6. 在选择实例启动选项页面上，执行以下操作：

a. 对于 Instance type requirements（实例类型要求），选择 Override launch template（覆盖启动模板），然后选择 Manually add instance types（手动添加实例类型）。

b. 选择您的实例类型。您可以使用我们的建议作为起点。默认情况下，将选择 Family and generation flexible（系列和世代灵活）。

- 要更改实例类型的顺序，请使用箭头。如果您选择支持优先级排序的分配策略，则实例类型顺序将设置其启动优先级。
- 要删除实例类型，请选择 X。
- （可选）对于权重列中的方框，为每个实例类型分配一个相对权重。要执行此操作，请输入该类型的实例计入组所需容量的单元数量。如果实例类型提供不同的 vCPU、内存、存储或网络带宽功能，则执行此操作可能很有用。有关更多信息，请参阅 配置自动扩缩组以使用实例权重。

  请注意，如果您选择使用大小灵活建议，则属于本部分的所有实例类型都将自动获得权重值。如果您不想指定任何权重，则请清除 Weight（权重）列中所有实例类型的方框。

c. 在 Instance purchase options（实例购买选项）下，对于 Instances distribution（实例分配），分别指定要启动的按需型实例和竞价型实例在该组中所占的百分比。如果您的应用程序无状态、容错，并且可以处理中断的实例，则可以指定更高的竞价型实例百分比。

d. （可选）如果您指定了一个竞价型实例百分比，请选中包括按需型基本容量旁边的复选框，然后指定按需型实例在自动扩缩组的最小初始容量中必须达到的数量。超出基本容量的任何内容都使用实例分配设置来确定要启动多少按需实例和 Spot 实例。

e. 在 Allocation strategies (分配策略) 下，对于 On-Demand allocation strategy (按需分配策略)，选择分配策略。当您手动选择实例类型时，默认情况下将选择 Prioritized（优先）。

f. 对于 Spot allocation strategy（竞价型分配策略），选择分配策略。默认情况下，将选择 Price capacity optimized（价格容量优化）。默认情况下，将隐藏 Lowest price（最低价格），仅在您选择 Show all strategies（显示所有策略）时才会出现。

- 如果您选择最低价格，则请输入价格最低的池的数量，以跨价格最低的池实现多样化。
- 如果您选择容量优化，则可以选择选中优先考虑实例类型复选框，让 Amazon A EC2 uto Scaling 根据您的实例类型的列出顺序选择首先启动的实例类型。

g. 对于容量再平衡，选择启用还是禁用容量再平衡。使用容量再平衡功能，以自动响应竞价型实例因竞价中断而即将终止的情况。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

h. 在 Network (网络) 下，对于 VPC，选择相应的 VPC。自动扩缩组必须与您在启动模板中指定的安全组创建在相同的 VPC 中。

i. 对于 Availability Zones and subnets (可用区和子网)，选择指定 VPC 中的一个或多个子网。可以在多个可用区中使用子网以提供高可用性。有关更多信息，请参阅 选择 VPC 子网时的注意事项。

j. 选择下一步、下一步。

7. 对于 Configure group size and scaling policies（配置组大小和扩缩策略）步骤，请执行以下操作：

   a. 在组大小下，对于所需容量，请输入要启动的实例的初始数量。

      默认情况下，所需容量以实例数量表示。如果您为实例类型分配了权重，则必须将此值转换为用于分配权重的相同计量单位，例如 v 的数量CPUs。

   b. 在扩展部分的扩展限制下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。有关更多信息，请参阅为自动扩缩组设置扩缩限制。

8. 选择 Skip to review (跳转以查看)。

9. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

创建混合实例组 (Amazon CLI)

使用命令行创建一个混合实例组

使用以下命令之一：

- create-auto-scaling-group (Amazon CLI)
- 新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

示例配置

以下示例配置显示如何使用不同的竞价分配策略创建混合实例组。

> **Note**
>
> 这些示例显示如何使用 JSON 或 YAML 格式的配置文件。如果使用 Amazon CLI 版本 1，则必须指定 JSON 格式的配置文件。如果您使用 Amazon CLI 版本 2，则可以指定格式为 YAML 或 JSON 的配置文件。

示例

- 示例 1：使用 capacity-optimized 分配策略启动竞价型实例
- 示例 2：使用 capacity-optimized-prioritized 分配策略启动竞价型实例
- 示例 3：使用在两个池之间不同的 lowest-price 分配策略启动竞价型实例
- 示例 4：使用 price-capacity-optimized 分配策略启动 Spot 实例

示例 1：使用 **capacity-optimized** 分配策略启动竞价型实例

以下create-auto-scaling-group命令创建一个 Auto Scaling 组，该组指定了以下内容：

- 作为按需型实例 (0) 启动的组的百分比以及以 (1) 开头的按需实例的基本数量。
- 按优先级顺序（c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.larg启动的实例类型。
- 要在其中启动实例的子网（subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782）。每个子网都对应不同的可用区。
- 启动模板 (my-launch-template) 和启动模板版本 ($Default)。

当 Amazon A EC2 uto Scaling 尝试满足您的按需容量时，它会首先启动c5.large实例类型。竞价型实例来自每个可用区中基于竞价型实例容量的最佳竞价池。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下代码。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
```

```
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

或者，您可以使用以下create-auto-scaling-group命令创建 Auto Scaling 组。这将引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml 文件包含以下内容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

示例 2：使用 **capacity-optimized-prioritized** 分配策略启动竞价型实例

以下create-auto-scaling-group命令创建一个 Auto Scaling 组，该组指定了以下内容：

- 作为按需型实例 (0) 启动的组的百分比以及以 (1) 开头的按需实例的基本数量。
- 按优先级顺序（c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.larg启动的实例类型。

- 要在其中启动实例的子网（subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782）。每个子网都对应不同的可用区。
- 启动模板 (my-launch-template) 和启动模板版本 ($Latest)。

当 Amazon A EC2 uto Scaling 尝试满足您的按需容量时，它会首先启动c5.large实例类型。当 Amazon A EC2 uto Scaling 尝试满足您的竞价容量时，它会尽力满足实例类型的优先级。但它首先会针对容量进行优化。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下代码。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
```

```
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized-prioritized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

或者，您可以使用以下create-auto-scaling-group命令创建 Auto Scaling 组。这将引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml 文件包含以下内容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
```

```
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

示例 3：使用在两个池之间不同的 **lowest-price** 分配策略启动竞价型实例

以下create-auto-scaling-group命令创建一个 Auto Scaling 组，该组指定了以下内容：

- 作为按需实例启动的组的百分比 (50)。（这不会指定要开始使用的按需型实例的基本数量。）
- 按优先级顺序（c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.larg启动的实例类型。
- 要在其中启动实例的子网（subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782）。每个子网都对应不同的可用区。
- 启动模板 (my-launch-template) 和启动模板版本 ($Latest)。

当 Amazon A EC2 uto Scaling 尝试满足您的按需容量时，它会首先启动c5.large实例类型。对于您的竞价容量，Amazon A EC2 uto Scaling 会尝试在每个可用区的两个价格最低的池中均匀启动竞价型实例。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下代码。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
```

```
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "lowest-price",
            "SpotInstancePools": 2
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

或者，您可以使用以下create-auto-scaling-group命令创建 Auto Scaling 组。这将引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml 文件包含以下内容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

示例 4：使用 **price-capacity-optimized** 分配策略启动 Spot 实例

以下create-auto-scaling-group命令创建一个 Auto Scaling 组，该组指定了以下内容：

- 作为按需实例启动的组的百分比 (30)。（这不会指定要开始使用的按需型实例的基本数量。）
- 按优先级顺序（c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.larg启动的实例类型。

- 要在其中启动实例的子网（subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782）。每个子网都对应不同的可用区。
- 启动模板 (my-launch-template) 和启动模板版本 ($Latest)。

当 Amazon A EC2 uto Scaling 尝试满足您的按需容量时，它会首先启动c5.large实例类型。对于您的竞价型容量，Amazon A EC2 uto Scaling 会尝试以尽可能低的价格从竞价型实例池中启动竞价型实例，但也要根据正在启动的实例数量提供最佳容量。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下代码。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
```

```
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 30,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

或者，您可以使用以下create-auto-scaling-group命令创建 Auto Scaling 组。这将引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml 文件包含以下内容。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
```

```
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 30
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

配置自动扩缩组以使用实例权重

当您使用多种实例类型时，可以指定与每种实例类型关联的单位数量，然后使用相同的度量单位来指定组的容量。此容量规格选项称为权重。

例如，假设您运行的计算密集型应用程序在至少有 8 v CPUs 和 15 GiB 的 RAM 时性能最佳。如果您使用`c5.2xlarge`作为基本单元，则以下任何一种 EC2 实例类型都将满足您的应用程序需求。

实例类型示例

实例类型	vCPU	内存 (GiB)
`c5.2xlarge`	8	16
`c5.4xlarge`	16	32
`c5.12xlarge`	48	96
`c5.18xlarge`	72	144
`c5.24xlarge`	96	192

默认情况下，无论大小如何，所有实例类型的权重都相同。换句话说，无论是 Amazon A EC2 uto Scaling 启动大型实例类型还是小型实例类型，每个实例在 Auto Scaling 组所需容量中的计数都是相同的。

但是，通过权重，您可以分配一个数值，该值指定与每种实例类型关联的单位数量。例如，如果实例大小不相同，`c5.2xlarge` 实例的权重可能为 2，而 `c5.4xlarge`（大两倍）的权重可能为 4，依此类推。然后，当 Amazon A EC2 uto Scaling 扩展该组时，这些权重会转换为每个实例计入所需容量的单位数。

权重不会改变 Amazon A EC2 uto Scaling 选择启动的实例类型；相反，分配策略会做到这一点。有关更多信息，请参阅 多种实例类型的分配策略。

> **Important**
>
> 要使用 v 的数量CPUs 或每种实例类型的内存量配置 Auto Scaling 组以满足其所需容量，我们建议使用基于属性的实例类型选择。设置 `DesiredCapacityType` 参数会根据您为此参数设置的值自动指定与每种实例类型关联的单位数量。有关更多信息，请参阅 使用基于属性的实例类型选择创建混合实例组。

内容

注意事项

本节讨论有效实施权重的关键注意事项。

- 选择几种符合应用程序性能需求的实例类型。根据每种实例类型的功能，确定其应计入自动扩缩组所需容量的权重。这些权重应用于当前和未来实例。
- 避免权重之间的差距过大。例如，当下一个较大的实例类型的权重为 200 时，不要将实例类型的权重指定为 1。此外，最小权重和最大权重之间的差异不能太极端。过大的权重差异会对性价比优化产生负面影响。
- 以单位（而不是实例数）指定组的所需容量。例如，如果使用基于 vCPU 的权重，请设置所需核心数以及最小核心数和最大核心数。
- 设置权重和所需容量，使所需容量至少比最大权重大两到三倍。

更新现有群组时请注意以下几点：

- 向现有组添加权重时，应包括当前正在使用的所有实例类型的权重。
- 当您添加或更改权重时，Amazon A EC2 uto Scaling 将根据新的权重值启动或终止实例以达到所需的容量。
- 如果您删除了实例类型，则该类型正在运行的实例将保留其上次权重，即使不再定义。

实例权重行为

当您使用实例权重时，Amazon A EC2 uto Scaling 的行为方式如下：

- 当前容量等于或高于所需容量。如果启动的实例超出剩余的所需容量单位，则当前容量可能会超过所需容量。例如，假设您指定了两个实例类型 `c5.2xlarge` 和 `c5.12xlarge`，并且您为 `c5.2xlarge` 分配了实例权重 2，为 `c5.12xlarge` 分配了实例权重 12。如果还有五个单位可以满足所需容量，且 Amazon A EC2 uto Scaling 配置了 `ac5.12xlarge`，则所需容量将超过七个单位。
- 启动实例时，Amazon A EC2 uto Scaling 优先考虑在可用区之间分配容量并遵守分配策略，而不是超过所需容量。
- 使用您的首选分配策略，Amazon A EC2 uto Scaling 可以超过最大容量限制，以保持可用区域间的平衡。Amazon A EC2 uto Scaling 强制执行的硬限制是您的所需容量加上您的最大重量。

配置自动扩缩组以使用权重

您可以将自动扩缩组配置为使用权重，如以下 Amazon CLI 示例所示。有关如何使用控制台的说明，请参阅通过手动选择实例类型来创建混合实例组。

配置新自动扩缩组以使用权重 (Amazon CLI)

使用 create-auto-scaling-group 命令。例如，以下命令创建一个新的自动扩缩组，并通过指定以下内容分配权重：

- 作为按需实例启动的组的百分比 (`0`)
- 每个可用区中竞价型实例的分配策略(`capacity-optimized`)。
- 按优先级顺序启动（`m4.16xlarge`、`m5.24xlarge`）的实例类型
- 实例权重对应于实例类型 (16,CPUs) 之间的相对大小差 (v24)
- 要在其中启动实例的子网（`subnet-5ea0c127`、`subnet-6194ea3b`、`subnet-c934b782`），每个子网对应于不同的可用区
- 启动模板 (`my-launch-template`) 和启动模板版本 (`$Latest`)

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 文件包含以下代码。

```
{
```

```
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "m4.16xlarge",
                    "WeightedCapacity": "16"
                },
                {
                    "InstanceType": "m5.24xlarge",
                    "WeightedCapacity": "24"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 160,
    "MaxSize": 720,
    "DesiredCapacity": 480,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

配置现有自动扩缩组以使用权重 (Amazon CLI)

使用 update-auto-scaling-group 命令。例如，以下命令通过指定以下内容向现有自动扩缩组中的实例类型分配权重：

- 按优先级顺序启动（c5.18xlarge、c5.24xlarge、c5.2xlarge、c5.4xlarge）的实例类型
- 实例权重对应于实例类型 (18、、242、CPUs) 之间的相对大小差 (v4)
- 新增加的所需容量，大于最大权重

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下代码。

```
{
    "AutoScalingGroupName": "my-existing-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "Overrides": [
                {
                    "InstanceType": "c5.18xlarge",
                    "WeightedCapacity": "18"
                },
                {
                    "InstanceType": "c5.24xlarge",
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceType": "c5.2xlarge",
                    "WeightedCapacity": "2"
                },
                {
                    "InstanceType": "c5.4xlarge",
                    "WeightedCapacity": "4"
                }
            ]
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 100
}
```

使用命令行验证权重

使用以下命令之一：

- describe-auto-scaling-groups (Amazon CLI)
- 获取-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

每单位小时 Spot 价格示例

下表比较美国东部（弗吉尼亚州北部）不同可用区中竞价型实例的每小时价格和同一区域中按需型实例的价格。显示的价格是示例定价，而不是当前定价。这些价格是您每实例小时的成本。

示例：每实例小时的竞价定价

实例类型	us–east–1a	us–east–1b	us–east–1c	按需定价
c5.2xlarge	0.180 USD	0.191 USD	0.170 USD	0.34 USD
c5.4xlarge	0.341 USD	0.361 USD	0.318 USD	0.68 USD
c5.12xlarge	0.779 USD	0.777 USD	0.777 USD	2.04 USD
c5.18xlarge	1.207 USD	1.475 USD	1.357 USD	3.06 USD
c5.24xlarge	1.555 USD	1.555 USD	1.555 USD	4.08 USD

通过实例权重，您可以根据每单位小时的使用量来评估您的成本。您可以将某种实例类型的价格除以它表示的单位数来确定每单位小时价格。对于按需实例，部署一个实例类型时的每单位小时 价格与部署不同大小的相同实例类型时的价格相同。但是，相比之下，每单位小时 的 Spot 价格因竞价池而异。

以下示例显示了如何使用实例权重计算每单位小时竞价价格。为便于计算，假定您只想启动 us-east-1a 中的竞价型实例。下表列出了每单位小时的价格。

示例：每单位小时的现货价格

实例类型	us–east–1a	实例权重	每单位小时价格
c5.2xlarge	0.180 USD	2	0.090 USD
c5.4xlarge	0.341 USD	4	0.085 USD
c5.12xlarge	0.779 USD	12	0.065 USD
c5.18xlarge	1.207 USD	18	0.067 USD
c5.24xlarge	1.555 USD	24	0.065 USD

为实例类型使作不同的启动模板

除了使用多种实例类型外，您还可以使用多种启动模板。

例如，假设您为计算密集型应用程序配置自动扩缩组，并希望混合 C5、C5a 和 C6g 实例类型。但是，C6g 实例采用基于 64 位 A Amazon rm 架构的 Graviton 处理器，而 C5 和 C5a 实例则在 64 位英特尔 x86 处理器上运行。 AMIs 适用于 C5 和 C5a 的实例都适用于其中的每一个实例，但不能在 C6g 实例上运行。要解决此问题，请对 C6g 实例使用不同的启动模板。您仍然可以对 C5 和 C5a 实例使用相同的启动模板。

本节包含使用 Amazon CLI 来执行与使用多个启动模板相关的任务的过程。目前，仅当您使用 Amazon CLI 或 SDK 时此功能才可用，并且不可从控制台使用。

内容

配置自动扩缩组以使用多个启动模板

您可以将自动扩缩组配置为使用多个启动模板，如以下示例所示。

要将新的自动扩缩组配置为使用多个启动模板 (Amazon CLI)

使用 create-auto-scaling-group 命令。例如，以下命令将创建一个新的自动扩缩组。它指定 `c5.large`、`c5a.large` 和 `c6g.large` 实例类型，并为 `c6g.large` 实例类型定义新的启动模板，以确保使用适当的 AMI 启动 Arm 实例。Amazon A EC2 uto Scaling 使用实例类型的顺序来确定在满足按需容量时应首先使用哪种实例类型。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 文件包含以下代码。

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{
      "LaunchTemplateSpecification":{
        "LaunchTemplateName":"my-launch-template-for-x86",
        "Version":"$Latest"
```

```
      },
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
    },
    "InstancesDistribution":{
      "OnDemandBaseCapacity": 1,
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize":1,
  "MaxSize":5,
  "DesiredCapacity":3,
  "VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags":[ ]
}
```

将现有自动扩缩组配置为使用多个启动模板 (Amazon CLI)

使用 update-auto-scaling-group 命令。例如，以下命令将名为 *my-launch-template-for-arm* 的自动扩缩组的 *c6g.large* 实例类型分配名为 *my-asg* 的启动模板。

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 文件包含以下内容。

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
```

```
    "LaunchTemplate":{
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
    }
  }
}
```

验证 Auto Scaling 组的启动模板

使用以下命令之一：

- describe-auto-scaling-groups (Amazon CLI)
- 获取-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

相关资源

你可以在 re: Post 的模板中找到使用基于属性的实例类型选择来指定多个启动 Amazon CloudFormation 模板的示例。Amazon

使用启动配置创建自动扩缩组

Important

限制：

- 自 2023 年 1 月 1 日起，启动配置中不再支持新的 Amazon EC2 实例类型。这包括支持在初始区域启动 Amazon Web Services 区域 后添加到中的任何实例类型。

- 2023 年 6 月 1 日当天或之后创建的账户无法使用控制台创建新的启动配置。
- 在 2024 年 10 月 1 日当天或之后创建的账户无法使用任何方法（控制台 Amazon CLI、API 或 CloudFormation）创建新的启动配置。

迁移到启动模板，确保现在或将来都不需要创建新的启动配置。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

如果您已创建启动配置或 EC2 实例，则可以创建一个 Auto Scaling 组，该组使用启动配置作为其 EC2 实例的配置模板。启动配置可以为实例指定一些信息，例如，AMI ID、实例类型、密钥对、安全组和块储存设备映射。有关创建启动配置的信息，请参阅创建启动配置。

您必须具有足够的权限来创建自动扩缩组。您还必须拥有足够的权限才能创建 Amazon A EC2 uto Scaling 用来代表您执行操作的服务相关角色（如果该角色尚不存在）。有关管理员在向您授予权限时可参考的 IAM policy 示例，请参阅 基于身份的策略示例。

内容

- 使用启动配置创建 Auto Scaling 组
- 使用现有实例创建 Auto Scaling 组 Amazon CLI

使用启动配置创建 Auto Scaling 组

Important

我们为尚未从启动配置迁移到启动模板的客户提供有关启动配置的信息。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

创建 Auto Scaling 组时，必须指定必要的信息，以配置 Amazon EC2 实例、实例的可用区和 VPC 子网、所需容量以及最小和最大容量限制。

以下过程演示如何使用启动配置创建 Auto Scaling 组。您无法在创建启动配置后进行修改，但可以替换 Auto Scaling 组的启动配置。有关更多信息，请参阅 更改 Auto Scaling 组的启动配置。

先决条件

- 您必须已创建启动配置。有关更多信息，请参阅 创建启动配置。

使用启动配置创建 Auto Scaling 组（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏上，选择与创建启动配置时相同的 Amazon Web Services 区域 配置。
3. 选择 Create an Auto Scaling group (创建 Auto Scaling 组)。
4. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，输入 Auto Scaling 组的名称。
5. 要选择启动配置，请执行以下操作：

   a. 对于 Launch Template (启动模板)，选择 Switch to launch configuration (切换以启动配置)。

   b. 对于 Launch configuration (启动配置)，请选择现有启动配置。

   c. 验证您的启动配置是否支持您计划使用的所有选项，然后选择 Next (下一步)。
6. 在 (配置设置) Configure instance launch options (配置实例启动选项) 页面的 Network (网络) 下方，对于 VPC，选择相应的 VPC。必须在您于启动配置中指定的安全组所在的 VPC 中创建 Auto Scaling 组。
7. 对于 (子网) Availability Zones and subnets (可用区和子网)，选择指定 VPC 中的一个或多个子网。可以在多个可用区中使用子网以提供高可用性。有关更多信息，请参阅 选择 VPC 子网时的注意事项。
8. 选择下一步。

   或者，您可接受其余默认值，然后选择 Skip to review (跳到审核)。
9. （可选）在 Configure advanced options（配置高级选项）页面上，配置以下选项，然后选择 Next（下一步）：

   a. （可选）对于运行状况检查、其他运行状况检查类型，选择开启 Amazon EBS 运行状况检查。有关更多信息，请参阅 使用运行状况检查监控具有受损 Amazon EBS 卷的 Auto Scaling 实例。

   b. （可选）对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

   c. 在 “其他设置” 下的 “监控” 下，选择是否启用 CloudWatch 群组指标收集。这些指标提供的测量值可以指示潜在的问题，例如终止实例的数量或挂起实例的数量。有关更多信息，请参阅 CloudWatch 监控您的 Auto Scaling 组和实例的指标。

d. 对于启用默认实例预热，选择此选项并选择应用程序的预热时间。如果您正在创建具有扩展策略的 Auto Scaling 组，则默认实例预热功能会改进用于动态扩展的 Amazon CloudWatch 指标。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

10. （可选）在 Configure group size and scaling policies (配置组大小和扩展策略) 页面上，配置以下选项，然后选择 Next (下一步)：

    a. 在组大小下，对于所需容量，请输入要启动的实例的初始数量。

    b. 在扩展部分的扩展限制下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。有关更多信息，请参阅为自动扩缩组设置扩缩限制。

    c. 对于自动扩缩，请选择是否要创建目标跟踪扩展策略。您也可以在创建自动扩缩组后再创建此策略。

       如果您选择目标跟踪扩展策略，请按照 创建目标跟踪扩缩策略 中的说明创建策略。

    d. 对于实例维护策略，请选择是否要创建实例维护策略。您也可以在创建自动扩缩组后再创建此策略。要创建策略，请按照设置实例维护政策中的指导操作。

    e. 在 Instance scale-in protection（实例缩减保护）下，选择是否启用实例缩减保护。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

11. （可选）要接收通知，请为 Add notification（添加通知）配置通知，然后选择 Next（下一步）。有关更多信息，请参阅 亚马逊 Auto Scaling 的亚马 EC2 逊 SNS 通知选项。

12. （可选）要添加标签，请选择 Add tag（添加标签），为每个标签提供标签键和值，然后选择 Next（下一步）。有关更多信息，请参阅 为 Auto Scaling 组和实例添加标签。

13. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

使用命令行创建 Auto Scaling 组

您可以使用以下任一命令：

- create-auto-scaling-group (Amazon CLI)
- 新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

使用现有实例创建 Auto Scaling 组 Amazon CLI

> **Important**
>
> 我们为尚未从启动配置迁移到启动模板的客户提供有关启动配置的信息。有关为自动扩缩组创建启动模板的更多信息，请参阅 将自动扩缩组迁移到启动模板。

如果这是您首次创建 Auto Scaling 组，我们建议您使用控制台从现有 EC2 实例创建启动模板。然后使用启动模板创建新的 Auto Scaling 组。有关此步骤，请参阅 使用亚马逊 EC2 启动向导创建 Auto Scaling 群组。

以下程序演示了如何通过如下方法创建 Auto Scaling 组：指定要用作启动其他实例基础的现有实例。创建 EC2 实例需要多个参数，例如 Amazon 系统映像 (AMI) ID、实例类型、密钥对和安全组。当需要扩展时，Amazon A EC2 uto Scaling 还会使用所有这些信息代表您启动实例。此信息存储在启动模板或启动配置中。

当您使用现有实例时，Amazon A EC2 uto Scaling 会创建一个 Auto Scaling 组，该组根据同时创建的启动配置启动实例。Auto Scaling 组的名称与 Auto Scaling 组相同，并且包括来自已识别实例的某些配置详细信息。

以下配置详细信息会从已识别的实例复制到启动配置中：

- AMI ID
- 实例类型
- 密钥对
- 安全组
- IP 地址类型（公有或私有）
- IAM 实例配置文件（如果适用）
- 监控（true 或 false）
- EBS 优化（true 或 false）
- 租期设置 (如果在 VPC (共享或专用) 中启动)
- 内核 ID 和 RAM 磁盘 ID (如果适用)
- 用户数据，如果指定
- Spot（最高）价格

VPC 子网和可用区将从已识别的实例复制到自动扩缩组自己的资源定义中。

如果已识别的实例位于置放群组中，则新 Auto Scaling 组将在与已识别实例相同的置放群组中启动实例。由于启动配置设置不允许指定置放群组，因此将置放群组复制到新 Auto Scaling 组的 `PlacementGroup` 属性。

不会从已识别实例中复制以下配置详细信息，

- 存储：不会从已识别的实例中复制块储存设备（EBS 卷和实例存储卷）。相反，作为创建 AMI 的一部分而创建的块储存设备映射决定了使用哪些设备。
- 网络接口数量：网络接口未从已识别的实例中复制。相反，Amazon A EC2 uto Scaling 使用其默认设置来创建一个网络接口，即主网络接口 (eth0)。
- 实例元数据选项：不会从已识别的实例中复制元数据可访问、元数据版本和令牌响应跃点数限制设置。相反，Amazon A EC2 uto Scaling 使用其默认设置。有关更多信息，请参阅 配置实例元数据选项。
- 负载均衡器：如果识别的实例适用一个或多个负载均衡器进行注册，则有关负载均衡器的信息不会复制到负载均衡器或新 Auto Scaling 组的目标组属性。
- 标签：如果识别的实例有标签，标签不会复制到新 Auto Scaling 组的 `Tags` 属性。

先决条件

该 EC2 实例必须满足以下标准：

- 实例不是其他 Auto Scaling 组的成员。
- 实例处于 `running` 状态。
- 用于启动实例的 AMI 必须仍然存在。

通过 EC2实例创建 Auto Scaling 组 (Amazon CLI)

以下过程向您展示如何使用 CLI 命令从 EC2 实例创建 Auto Scaling 组。

此程序不会将实例添加到 Auto Scaling 组中。要连接实例，必须在创建 Auto Scaling 组之后运行 attach-instances 命令。

在开始之前，请使用亚马逊 EC2 控制台或 desc ribe- instances 命令查找 EC2 实例的 ID。

将当前实例用作模板

- 使用以下create-auto-scaling-group命令从该 EC2 实例创建 Auto Scaling 组i-123456789abcdefg0。my-asg-from-instance

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
  --instance-id i-123456789abcdefg0 --min-size 1 --max-size 2 --desired-capacity 2
```

验证 Auto Scaling 组已启动实例

- 使用以下describe-auto-scaling-groups命令验证 Auto Scaling 组是否已成功创建。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

以下示例响应显示该组的所需容量为 2，该组有 2 个正在运行的实例，启动配置命名为 my-asg-from-instance。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg-from-instance",
      "AutoScalingGroupARN":"arn",
      "LaunchConfigurationName":"my-asg-from-instance",
      "MinSize":1,
      "MaxSize":2,
      "DesiredCapacity":2,
      "DefaultCooldown":300,
      "AvailabilityZones":[
        "us-west-2a"
      ],
      "LoadBalancerNames":[],
      "TargetGroupARNs":[],
      "HealthCheckType":"EC2",
      "HealthCheckGracePeriod":0,
      "Instances":[
        {
          "InstanceId":"i-34567890abcdef012",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
```

```
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-012345abcdefg6789",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        }
      ],
      "CreatedTime":"2020-10-28T02:39:22.152Z",
      "SuspendedProcesses":[ ],
      "VPCZoneIdentifier":"subnet-0abc1234",
      "EnabledMetrics":[ ],
      "Tags":[ ],
      "TerminationPolicies":[
        "Default"
      ],
      "NewInstancesProtectedFromScaleIn":false,
      "ServiceLinkedRoleARN":"arn",
      "TrafficSources":[]
    }
  ]
}
```

查看启动配置

- 使用以下describe-launch-configurations命令查看启动配置的详细信息。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-asg-from-instance
```

下面是示例输出：

```
{
  "LaunchConfigurations":[
    {
```

```
      "LaunchConfigurationName":"my-asg-from-instance",
      "LaunchConfigurationARN":"arn",
      "ImageId":"ami-234567890abcdefgh",
      "KeyName":"my-key-pair-uswest2",
      "SecurityGroups":[
        "sg-12abcdefgh3456789"
      ],
      "ClassicLinkVPCSecurityGroups":[ ],
      "UserData":"",
      "InstanceType":"t2.micro",
      "KernelId":"",
      "RamdiskId":"",
      "BlockDeviceMappings":[ ],
      "InstanceMonitoring":{
        "Enabled":true
      },
      "CreatedTime":"2020-10-28T02:39:22.321Z",
      "EbsOptimized":false,
      "AssociatePublicIpAddress":true
    }
  ]
}
```

终止实例

- 如果您不再需要实例，可终止它。以下 terminate-instances 命令可终止实例i-123456789abcdefg0。

  ```
  aws ec2 terminate-instances --instance-ids i-123456789abcdefg0
  ```

  终止 Amazon EC2 实例后，您将无法重启该实例。终止后，卷上的数据都不复存在，并且再也不能附加到任何实例。要了解有关终止实例的更多信息，请参阅 Amazon EC2 用户指南中的终止实例。

更新自动扩缩组

您可以更新自动扩缩组的大部分详细信息。您无法更新 Auto Scaling 组的名称或更改其名称 Amazon Web Services 区域。

更新自动扩缩组（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择您的自动扩缩组以显示有关该组的信息，其中包含详细信息、活动、自动扩缩、实例管理、监控和实例刷新选项卡。
3. 选择您感兴趣的配置区域的选项卡，然后根据需要更新设置。对于您编辑的每个设置，请选择更新以保存对自动扩缩组配置所做的更改。

   - 详细信息选项卡

     这些是自动扩缩组的常规设置。您可以像创建自动扩缩组时那样编辑和管理这些设置。

     高级配置部分包含一些在创建组时不可用的选项，例如终止策略、冷却时间、暂停的进程和最大实例生命周期。您也可以查看但不能编辑自动扩缩组的置放群组和服务相关角色。

   - “集成” 选项卡
     - 负载平衡 — Elastic 负载平衡

       如果该组与 Elastic Load Balancing 资源相关联，请在更改可用区之前参阅 添加可用区。对负载均衡器的某些限制可能会阻止您将该组的可用区更改应用于负载均衡器的可用区。

     - VPC 莱迪思集成选项 — VPC 莱迪思
     - ARC 区域偏移 — A uto Scaling 组区域偏移
   - 自动扩缩选项卡
     - 动态扩缩策略：动态扩缩策略
     - 预测性扩展策略：预测性扩展策略
     - 计划操作：计划操作
   - 实例管理选项卡
     - 生命周期挂钩：生命周期挂钩
     - 暖池：暖池
   - 活动选项卡
     - 活动通知：Amazon SNS 通知
   - 监控选项卡
     - 此选项卡中只有一个选项，允许您启用或禁用CloudWatch群组指标收集。

使用命令行来更新自动扩缩组

您可以使用以下任一命令：

- update-auto-scaling-group (Amazon CLI)
- 更新-ASAuto ScalingGroup (Amazon Tools for Windows PowerShell)

更新自动扩缩实例

如果您将新的启动模板或启动配置与自动扩缩组相关联，那么所有新实例都将获得更新后的配置。现有实例继续采用它们最初启动时采用的配置运行。要将更改应用于现有实例，您有以下选项：

- 启动实例刷新以替换旧实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。
- 根据终止策略等待扩缩活动逐步使用较新的实例替换较旧的实例。
- 手动终止它们，这样它们就会被您的自动扩缩组所替代。

> **Note**
>
> 您可以通过将以下实例属性指定为启动模板或启动配置的一部分来更改这些属性：
>
> - 亚马逊机器映像 (AMI)
> - 块储存设备
> - 密钥对
> - 实例类型
> - 安全组
> - 用户数据
> - 监控
> - IAM 实例配置文件
> - 部署租期
> - kernel
> - 虚拟磁盘
> - 实例是否有公有 IP 地址
> - 可用区分发策略

Auto Scaling 群组分配策略和容量变化

当您更改 Auto Scaling 组分配策略时，不会替换现有实例。由于扩展事件而启动的任何新实例都将遵循新的分配策略。任何未来规模的事件都将遵循终止政策，如果终止政策设置为`Default`或，则使用新的分配策略`AllocationStrategy`。例如，如果您将分配策略从更改`lowest-price`为`price-capacity-optimized`，则可能不会终止任何实例，但任何新实例都将使用新的分配策略启动。实例类型更改不会影响现有实例。

当您更改某些参数（例如OnDemandBaseCapacity或）时OnDemandPercentageAboveBaseCapacity，如果按需实例和竞价型实例的百分比与新规格不匹配，Auto Scaling 将自动重新平衡。例如，假设一个 Auto Scaling 组将按需实例`OnDemandPercentageAboveBaseCapacity`设置为 50% 和 50% 竞价型实例。然后，将`OnDemandPercentageAboveBaseCapacity`按需实例增加到 100%。Auto Scaling 组将通过启动新的按需实例和终止竞价型实例来主动进行再平衡。您定义的实例维护策略决定了启动和终止活动的顺序。

为 Auto Scaling 组和实例添加标签

标签是您分配或分配给 Amazon 资源的自定义属性标签。 Amazon 每个标签具有两个部分：

- 标签键（例如，`costcenter`、`environment` 或 `project`）
- 一个称为标签值的可选字段（例如，`111122223333` 或 `production`）

标签可帮助您：

- 追踪您的 Amazon 成本。您可以在 Amazon Billing and Cost Management 控制面板上激活这些标签。 Amazon 使用标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅 Amazon Billing 用户指南中的使用成本分配标签。
- 根据标签控制对 Auto Scaling 组的访问。您可以使用 IAM policy 中的条件根据该组上的标签控制对自动扩缩组的访问。有关更多信息，请参阅 安全性标签。
- 根据您添加的标签筛选和搜索自动扩缩组。有关更多信息，请参阅 使用标签筛选 Auto Scaling 组。
- 识别和整理您的 Amazon 资源。许多都 Amazon Web Services 服务 支持标记，因此您可以为来自不同服务的资源分配相同的标签，以表明这些资源是相关的。

您可以标记新的或现有 Auto Scaling 组。您还可以将标签从 Auto Scaling 组传播到该组启动的 EC2 实例。

标签不会传播到 Amazon EBS 卷。要向 Amazon EBS 卷添加标签，请在启动模板中指定标签。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

您可以通过 Amazon Web Services Management Console Amazon CLI、或创建和管理标签 SDKs。

内容

标签命名和使用限制

下面是适用于标签的基本限制：

- 每个资源的最大标签数是 50。
- 可以使用单个调用添加或删除的标签的最大数目为 25。
- 最大键长度为 128 个 Unicode 字符。
- 最大值长度为 256 个 Unicode 字符。
- 标签键和值区分大小写。最佳实践是，决定利用标签的策略并在所有资源类型中一致地实施该策略。
- 请勿在标签名称或值中使用`aws:`前缀，因为它是保留供 Amazon 使用的。您不能编辑或删除具有此前缀的标签名称或值，它们不计入每个资源配额的标签数限制。

EC2 实例标记生命周期

如果您选择将标签传播到您的 EC2 实例，则标签的管理方式如下：

- 当 Auto Scaling 组启动实例时，它会在资源创建期间向实例添加标记，而不是在创建资源之后。
- Auto Scaling 组会自动向实例添加带有 `aws:autoscaling:groupName` 键和 Auto Scaling 组名称值的标签。

- 如果您在启动模板中指定了实例标签，并且选择将组的标签传播到其实例，则所有标签都会合并。如果为启动模板中的标签和 Auto Scaling 组中的标签指定了相同的标签键，则优先使用该组中的标签值。
- 您在附加现有实例时，Auto Scaling 组就会向这些实例添加标签，覆盖具有相同标签关键字的现有标签。它还添加键为 `aws:autoscaling:groupName`、值为 Auto Scaling 组名称的标签。
- 将实例从 Auto Scaling 组中分离时，它仅删除 `aws:autoscaling:groupName` 标签。

标记 Auto Scaling 组

当您向 Auto Scaling 组添加标签时，可以指定是否应将其添加到 Auto Scaling 组中启动的实例。如果修改标签，在更改后，标签的更新版本将添加到在 Auto Scaling 组中启动的实例。如果创建或修改 Auto Scaling 组的标签，不会对已经在 Auto Scaling 组中运行的实例进行这些更改。

内容

添加或修改标签（控制台）

创建时标记 Auto Scaling 组

使用亚马逊 EC2 控制台创建 Auto Scaling 组时，可以在创建 Auto Scaling 组向导的添加标签页面上指定标签键和值。要将标签传播到在 Auto Scaling 组中启动的实例，请确保该标签的 Tag new instances（标记新实例）选项保持选中状态。否则，您可以取消选择它。

添加或修改现有 Auto Scaling 组的标签

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Tags（标签）、Edit（编辑）。
4. 要修改现有标签，请编辑 Key（键）和 Value（值）。

5. 要添加新标签，请选择 Add tag（添加标签），然后编辑 Key（键）和 Value（值）。您可以使标记新实例保持选中状态，以便自动将标签添加到在 Auto Scaling 组启动的实例，否则取消选中它。
6. 添加完标签后，选择 Update（更新）。

添加或修改标签 (Amazon CLI)

以下示例说明如何在创建 Auto Scaling 组时使用添加标签，以及如何为现有 Auto Scaling 组添加或修改标签。 Amazon CLI

创建时标记 Auto Scaling 组

使用create-auto-scaling-group命令创建新的 Auto Scaling 组，然后向 Auto Scaling 组添加标签**environment=production**，例如添加标签。该标签还会添加到在 Auto Scaling 组中启动的任何实例。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --tags Key=environment,Value=production,PropagateAtLaunch=true
```

创建或修改现有 Auto Scaling 组的标签

可以使用 create-or-update-tags 命令创建或修改标签。例如，以下命令将添加 **Name=my-asg** 和 **costcenter=cc123** 标签。在进行该更改后，该标签还会添加到在 Auto Scaling 组中启动的任何实例。如果具有任一键的标签已经存在，则会替换现有标签。Amazon EC2 控制台将每个实例的显示名称与为Name密钥指定的名称相关联（区分大小写）。

```
aws autoscaling create-or-update-tags \
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-
asg,PropagateAtLaunch=true \
  ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

描述 Auto Scaling 组的标签 (Amazon CLI)

如果您要查看应用于特定的 Auto Scaling 组的标签，可以使用以下任一命令：

- describe-tags：您提供自动扩缩组的名称，以查看指定组的标签列表。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

以下为响应示例。

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "production",
            "Key": "environment"
        }
    ]
}
```

- describe-auto-scaling-groups— 您可以提供 Auto Scaling 组名称以查看指定组的属性，包括任何标签。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下为响应示例。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
```

```
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        }
    ]
}
```

删除标签

您可以随时删除与 Auto Scaling 组关联的标签。

内容

- 删除标签（控制台）
- 删除标签 (Amazon CLI)

删除标签（控制台）

删除标签

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Tags（标签）、Edit（编辑）。
4. 选择标签旁边的 Remove（删除）。
5. 选择 Update（更新）。

删除标签 (Amazon CLI)

使用 delete-tags 命令删除标签。例如，以下命令删除键为 **environment** 的标签。

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=environment"
```

您必须指定标签键，但无需指定值。如果您指定了一个值，并且该值不正确，则不会删除标签。

安全性标签

使用标签来验证请求者（例如 IAM 用户或角色）是否有权创建、修改或删除特定自动扩缩组。使用下面的一个或多个条件键，在 IAM policy 的条件元素中提供标签信息：

- 使用 `autoscaling:ResourceTag/tag-key: tag-value` 可允许（或拒绝）带特定标签的 Auto Scaling 组上的用户操作。
- 使用 `aws:RequestTag/tag-key: tag-value` 要求在请求中存在（或不存在）特定标签。
- 使用 `aws:TagKeys [tag-key, ...]` 要求在请求中存在（或不存在）特定标签键。

例如，您可能拒绝对包含具有键 **environment** 和值 **production** 的标签的 Auto Scaling 组的访问，如以下示例所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"autoscaling:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

有关使用条件键控制自动扩缩组访问的更多信息，请参阅 Amazon A EC2 uto Scaling 如何与 IAM 配合使用。

控制对标签的访问

使用标签来验证请求者（例如 IAM 用户或角色）是否有权添加、修改或删除自动扩缩组的标签。

以下 IAM policy 示例授予主体权限仅从自动扩缩组中删除带有 **`temporary`** 密钥的标签。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "autoscaling:DeleteTags",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
            }
        }
    ]
}
```

有关对自动扩缩组指定的标签实施限制的 IAM policy 的更多示例，请参阅 控制可以使用哪些标签键和标签值。

> **Note**
>
> 在实例启动后，即使您制定限制您的用户对 Auto Scaling 组执行标记（或取消标记）操作的策略，这也不会防止他们手动更改实例上的标签。有关控制实例标签访问权限的示 EC2 例，请参阅 Amazon EC2 用户指南中的示例：标记资源。

使用标签筛选 Auto Scaling 组

以下示例向您展示了如何使用带describe-auto-scaling-groups命令的过滤器来描述带有特定标签的 Auto Scaling 组。按标签筛选仅限于 Amazon CLI 或 SDK，无法通过控制台进行筛选。

筛选注意事项

- 您可以在单一请求中指定多个筛选条件和多个筛选条件值。
- 您不可以将通配符与筛选值一同使用。

- 筛选值区分大小写。

示例：使用特定标签键和值对描述 Auto Scaling 组

以下命令展示了如何筛选结果以仅显示具有 **environment=production** 的标签键和值对的 Auto Scaling 组。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

以下为响应示例。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        },

    ... additional groups ...

    ]
}
```

或者，您也可以使用 `tag:<key>` 筛选条件指定标签。例如，以下命令展示了如何筛选结果以仅显示具有 **environment=production** 的标签键和值对的 Auto Scaling 组。此筛选条件的格式如下所示：`Name=tag:<key>,Values=<value>`，其中采用代表标签键值对的 **<key>** 和 **<value>**。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production
```

您也可以使用`--query`选项筛选 Amazon CLI 输出。以下示例说明如何仅将前一个命令的 Amazon CLI 输出限制为组名、最小大小、最大大小和所需的容量属性。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production \
  --query "AutoScalingGroups[].{AutoScalingGroupName: AutoScalingGroupName, MinSize:
 MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity}"
```

以下为响应示例。

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "MinSize": 0,
        "MaxSize": 10,
        "DesiredCapacity": 1
    },

    ... additional groups ...

]
```

有关筛选的更多信息，请参阅《Amazon Command Line Interface 用户指南》中的筛选 Amazon CLI 输出。

示例：描述带有与指定标签键匹配的标签的 Auto Scaling 组

以下命令演示了如何筛选结果以仅显示带有 **environment** 标签的 Auto Scaling 组，并且不考虑标签值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment
```

示例：描述带有与指定标签键集匹配的标签的 Auto Scaling 组

以下命令显示如何筛选结果以仅显示带有 **environment** 和 **project** 的标签的 Auto Scaling 组，并且不考虑标签值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

示例：描述具有与至少一个指定标签键匹配的标签的 Auto Scaling 组

以下命令显示如何筛选结果以仅显示带有 **environment** 或 **project** 的标签的 Auto Scaling 组，并且不考虑标签值。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment,project
```

示例：描述带有指定标签值的 Auto Scaling 组

以下命令展示了如何筛选结果以仅显示标签值为 **production** 的 Auto Scaling 组，并且不考虑标签键。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production
```

示例：描述带有指定标签值集的 Auto Scaling 组

以下命令显示如何筛选结果以仅显示具有标签值 **production** 和 **development** 的 Auto Scaling 组，并且不考虑标签键。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production Name=tag-value,Values=development
```

示例：描述带有与至少一个指定标签值匹配的标签的 Auto Scaling 组

以下命令展示了如何筛选结果以仅显示标签值为 **production** 或**development** 的 Auto Scaling 组，并且不考虑标签键。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production,development
```

示例：描述带有与多个标签键和值匹配的标签的 Auto Scaling 组

您也可以组合过滤器来创建自定义 AND 以及 OR 逻辑来进行更复杂的过滤。

以下命令显示如何筛选结果以仅显示具有特定标签集的 Auto Scaling 组。一个标签密钥是 **environment** AND 标签值是 (**production** OR **development**) AND 另一个标签密钥是 **costcenter** AND 标签值为**cc123**。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production,development
 Name=tag:costcenter,Values=cc123
```

实例维护策略

您可以为自动扩缩组配置实例维护策略，以满足导致实例被替换的事件（例如实例刷新或运行状况检查过程）期间的特定容量要求。

例如，假设您有一个自动扩缩组，该组具有少量实例。当运行状况检查显示实例受损时，您需要避免因终止实例然后更换实例而造成的潜在中断。通过实例维护策略，您可以确保 Amazon A EC2 uto Scaling 首先启动一个新实例，然后等待其完全准备就绪，然后再终止运行状况不佳的实例。

实例维护策略还可以帮助您在同时更换多个实例时最大限度地减少任何潜在的中断。您可以为策略设置最低和最高运行正常百分比参数，并且在替换实例时，自动扩缩组只能在该最小-最大范围内增加和减少容量。范围越大，可以同时替换的实例的数量就会增加。

内容

自动扩缩组的实例维护策略

本主题概述了可用选项，并介绍了创建实例维护策略时需要考虑的内容。

内容

概览

当您为 Auto Scaling 组创建实例维护策略时，该策略会影响导致实例被替换的 Amazon A EC2 uto Scaling 事件。这样可以在同一自动扩缩组中实现更一致的替换行为。它还允许您根据需要优化群组的可用性或成本。

在控制台中，有以下配置选项可用：

- 终止前启动 – 必须先配置新实例，然后才能终止现有实例。对于偏向于可用性而不是成本节约的应用程序来说，这种方法是一个不错的选择。
- 终止并启动 – 在终止现有实例的同时配置新实例。对于偏向于节省成本而不是可用性的应用程序来说，这种方法是一个不错的选择。对于启动容量不应超过当前可用容量的应用程序来说，这也是一个不错的选择，即使在替换实例时也是如此。
- 自定义策略 – 此选项允许您在替换实例时使用自定义的最小和最大容量范围来设置策略。这种方法可以帮助您在成本和可用性之间取得适当的平衡。

自动扩缩组的默认设置是没有实例维护策略，这会使它以默认行为响应实例维护事件。下表描述了默认行为。

实例维护事件默认行为

事件	描述	默认行为
运行状况检查失败	当实例未通过运行状况检查时自动发生。Amazon A EC2 uto Scaling 会替换运行状况检查失败的实例。有关运行状况检查失败的原因，请参阅 自动扩缩组中实例的运行状况检查。	终止并启动。
实例刷新	在启动实例刷新时发生。根据您的配置，实例刷新可以一次替换一个实例，一次替换多个实例，也可以一次替换全部实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。	终止并启动。

事件	描述	默认行为
最大实例生命周期	当实例达到您为自动扩缩组指定的最大实例生命周期时，会自动发生。Amazon A EC2 uto Scaling 会替换已达到最大实例生命周期的实例。有关更多信息，请参阅 基于最大实例生命周期替换 Auto Scaling 实例。	终止并启动。
再平衡	如果存在导致组不平衡的潜在变化，则会自动发生。在以下情况下，Amazon A EC2 uto Scaling 会重新平衡群组： • 以前容量不足的可用区会恢复，或者您可以从组中添加或删除可用区。发生这种情况时，您的自动扩缩组会尝试在可用区域之间均衡自己。有关更多信息，请参阅 再平衡活动。 • 您在自动扩缩组上启用容量再平衡，它会尝试启动新的竞价型实例，然后随着竞价型实例的可用性发生变化，现有竞价型实例被中断。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。 • 您更新您的自动扩缩组，它会逐渐替换实例，以匹配您在更新混合实例策略时选择的新购买选项。有关更多信息，请参阅 更新自动扩缩组。	在终止之前启动。 Amazon A EC2 uto Scaling 最多可以超出群组最大容量的 10%。但是，如果您使用容量再平衡，则最多只能超过所需容量的 10%。

在以下情况下，Amazon A EC2 uto Scaling 将继续默认终止并启动。因此，当其中一种情况发生时，您的组容量可能会低于您的实例维护策略的下限阈值。

- 当实例意外终止时，例如由于人为行为。Amazon A EC2 uto Scaling 会立即替换不再运行的实例。有关更多信息，请参阅 Amazon EC2 健康检查。
- 在 Amazon A EC2 uto Scaling 启动替换实例之前，当亚马逊作为计划事件的一部分 EC2 重启、停止或停用实例时。有关这些事件的更多信息，请参阅 Amazon EC2 用户指南中的实例计划事件。
- 当 Amazon EC2 竞价服务启动竞价型实例时，竞价型实例会被强制终止。

对于竞价型实例，如果您在自动扩缩组上启用了容量再平衡，则该实例可能已经有一个来自我们在启动竞价中断之前启动的不同竞价池中的待处理实例。有关容量重新平衡的工作方式的详细信息，请参阅在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

但是，由于不能保证竞价型实例保持可用状态，并且可以在两分钟内发出竞价型实例中断通知后终止，因此，如果实例在新实例启动之前中断，则可能会超过您的实例维护政策的下限阈值。

核心概念

在您开始之前，请熟悉以下核心概念和术语：

所需容量

所需容量是自动扩缩组在创建时的容量。这也是该组在没有附加任何扩展条件时尝试保持的容量。

实例维护政策

实例维护策略控制是否在因实例维护事件而终止现有实例之前先配置实例。它还决定了您的自动扩缩组可能在多大程度上低于或超过所需容量才能同时替换多个实例。

最高运行正常百分比

最高运行正常百分比是替换实例时您的自动扩缩组可以增加到的所需容量的百分比。它表示组中可以处于运行状态且运行状况良好或待处理以支持您的工作负载的最大百分比。在控制台中，使用终止前启动选项或自定义策略选项时，您可以设置最高运行正常百分比。有效值为 100–200%。

最低运行正常百分比

最低运行正常百分比是在替换实例时保持正常运行、运行良好且随时可用于支持您的工作负载的所需容量的百分比。成功完成首次运行状况检查并且经过指定的预热时间后，该实例被视为运行状况良好，可以随时使用。在控制台中，使用终止并启动选项或自定义策略选项时，您可以设置最低运行正常百分比。有效值为 0–100%。

> **Note**
>
> 要更快地替换实例，您可以指定较低的最低运行正常百分比。但是，如果运行正常的实例不足，则可用性可能会降低。我们建议选择一个合理的值，以便在需要替换多个实例的情况下保持可用性。

实例预热

如果您的实例在进入`InService`状态后需要时间进行初始化，请为您的自动扩缩组启用默认实例预热。使用默认实例预热，您可以防止实例在准备就绪之前计入最低运行正常百分比。这可确保 Amazon A EC2 uto Scaling 在终止现有实例之前考虑需要多长时间才能有足够的容量来支持工作负载。

此外，启用默认实例预热功能后，您可以改进用于动态扩展的 Amazon CloudWatch 指标。如果您的 Auto Scaling 组有任何扩展策略，则当该组向外扩展时，它将使用相同的默认预热期，以防止在实例完成初始化之前将其计入 CloudWatch 指标。

有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

运行状况检查宽限期

Amazon A EC2 uto Scaling 根据您的 Auto Scaling 组使用的运行状况检查的状态来确定实例是否运行正常。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

为确保这些运行状况检查尽快开始，请勿将组的运行状况检查宽限期设置得过高，而应设置得足够高，以便 Elastic Load Balancing 运行状况检查确定目标是否可用于处理请求。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

扩展您的自动扩缩组

实例维护策略仅适用于实例维护事件，并不阻止手动或自动扩缩组。

当您的自动扩缩组中附加了扩展策略或计划操作时，它们可以在实例维护事件发生时并行运行。在这种情况下，他们可以增加或减少组的所需容量，但只能在您定义的扩展限制范围内。有关这些限制的更多信息，请参阅为自动扩缩组设置扩缩限制。

应用场景示例

在典型情况下，您的实例维护策略和所需容量可能如下所示：

- 最低运行正常百分比 = 90%

- 最高运行正常百分比 = 120%
- 所需容量 = 100

在任何实例维护事件中，您的自动扩缩组可能少则有 90 个实例，而多则有 120 个实例。事件发生后，该组恢复到拥有 100 个实例的状态。

当您对具有暖池的自动扩缩组使用实例维护策略时，最低和最高运行正常百分比将分别应用于自动扩缩组和暖池。

例如，假设这是您的配置：

- 最低运行正常百分比 = 90%
- 最高运行正常百分比 = 120%
- 所需容量 = 100
- 暖池大小 = 10

如果您启动实例刷新以回收该组的实例，Amazon A EC2 uto Scaling 会先替换 Auto Scaling 组中的实例，然后替换温池中的实例。虽然 Amazon A EC2 uto Scaling 仍在努力替换 Auto Scaling 组中的实例，但该组可能只有 90 个实例，多达 120 个。完成群组操作后，Amazon A EC2 uto Scaling 可以开始替换温池中的实例。发生这种情况时，温暖池可能少则有 9 个实例，而多则有 12 个实例。

为自动扩缩组设置实例维护策略

您可以在创建自动扩缩组时创建实例维护策略。也可以为现有的组创建此功能。

通过为自动扩缩组设置实例维护策略，您不必再为实例刷新功能指定最低和最高运行正常百分比，除非您想覆盖实例维护策略。

在控制台中，Amazon A EC2 uto Scaling 提供了帮助您入门的选项。

内容

设置实例维护政策

要在自动扩缩组上设置实例维护策略，请使用以下方法之一：

Console

为新的组设置实例维护策略（控制台）

1. 按照 使用启动模板创建 Auto Scaling 组 中的说明完成过程中的每个步骤，直到步骤 11。
2. 在配置组大小和扩展策略中，对于所需容量，输入要启动的初始实例数。
3. 在扩展部分的扩展限制下，如果所需容量的新值大于所需的最小容量和最大所需容量，则所需的最大容量将自动增加到新的所需容量值。您可以按需更改这些限制。
4. 对于自动扩缩，请选择是否要创建目标跟踪扩展策略。您也可以在创建自动扩缩组后再创建此策略。

   如果您选择目标跟踪扩展策略，请按照 创建目标跟踪扩缩策略 中的说明创建策略。
5. 在实例维护策略部分，选择下列可用选项之一：
   - 终止前启动：必须先配置新实例，然后才能终止现有实例。对于偏向于可用性而不是成本节约的应用程序来说，这是一个不错的选择。
   - 终止并启动：在终止现有实例的同时配置新实例。对于偏向于节省成本而不是可用性的应用程序来说，这是一个不错的选择。对于启动容量不应超过当前可用容量的应用程序来说，它也是一个不错的选择。
   - 自定义策略：此选项允许您在替换实例时使用自定义的最小和最大容量范围来设置策略。这可以帮助您在成本和可用性之间取得适当的平衡。
6. 对于设置运行正常百分比，为以下一个或两个字段输入值。根据您在上一步中选择的选项，启用的字段会有所不同。
   - 最小：设置继续替换实例所需的最低运行正常百分比。
   - 最大：设置替换实例时可能的最高运行正常百分比。
7. 展开根据所需容量在更换期间查看容量部分，以确认最小值和最大值的值如何适用于您的组。使用的确切值取决于所需的容量值，如果组发生扩缩，该值将发生变化。
8. 继续完成使用启动模板创建 Auto Scaling 组中的步骤。

Amazon CLI

为新组设置实例维护策略 (Amazon CLI)

在create-auto-scaling-group命令中添加`--instance-maintenance-policy`选项。以下示例对名为的新自动扩缩组设置实例维护策略*my-asg*。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --default-instance-warmup 20 \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }' \
  --vpc-zone-identifier "subnet-5e6example,subnet-613example,subnet-c93example"
```

Console

为现有的组设置实例维护策略（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在详细信息选项卡上，选择实例维护策略，编辑。
5. 要为组设置实例维护策略，请选择下列可用选项之一：

   - 终止前启动：必须先配置新实例，然后才能终止现有实例。对于偏向于可用性而不是成本节约的应用程序来说，这是一个不错的选择。
   - 终止并启动：在终止现有实例的同时配置新实例。对于偏向于节省成本而不是可用性的应用程序来说，这是一个不错的选择。对于启动容量不应超过当前可用容量的应用程序来说，它也是一个不错的选择。
   - 自定义策略：此选项允许您在替换实例时使用自定义的最小和最大容量范围来设置策略。这可以帮助您在成本和可用性之间取得适当的平衡。
6. 对于设置运行正常百分比，为以下一个或两个字段输入值。根据您在上一步中选择的选项，启用的字段会有所不同。

   - 最小：设置继续替换实例所需的最低运行正常百分比。

- 最大：设置替换实例时可能的最高运行正常百分比。

7. 展开根据所需容量在更换期间查看容量部分，以确认最小值和最大值的值如何适用于您的组。使用的确切值取决于所需的容量值，如果组发生扩缩，该值将发生变化。
8. 选择更新。

Amazon CLI

为现有组设置实例维护策略 (Amazon CLI)

在update-auto-scaling-group命令中添加--instance-maintenance-policy选项。以下示例为指定的自动扩缩组设置实例维护策略。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }'
```

删除实例维护策略

如果您想停止在自动扩缩组使用实例维护策略，则可以将其删除。

Console

删除实例维护策略（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在详细信息选项卡上，选择实例维护策略，编辑。
5. 选择无实例维护策略。
6. 选择更新。

Amazon CLI

删除实例维护策略 (Amazon CLI)

在update-auto-scaling-group命令中添加--instance-maintenance-policy选项。以下示例从指定的自动扩缩组删除实例维护策略。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": -1,
      "MaxHealthyPercentage": -1
    }'
```

Amazon A EC2 uto Scaling 生命周期挂钩

Amazon A EC2 uto Scaling 允许向你的 Auto Scaling 群组添加生命周期挂钩。这些钩子使 Auto Scaling 组可让您创建解决方案，这些解决方案了解 Auto Scaling 实例生命周期中的事件，然后在发生相应的生命周期事件时对实例执行自定义操作。生命周期钩子提供了指定的时间（预设情况下为 1 小时），以在实例转换到下一个状态之前等待操作完成。

作为将生命周期钩子与 Auto Scaling 实例一起使用的示例：

- 在发生向外扩展事件时，您新启动的实例将完成其启动序列并转换到等待状态。该实例处于等待状态时，它将运行脚本以下载和安装您的应用程序所需的软件包，确保您的实例在开始接收流量前已完全准备好。脚本安装完软件后，它会发送 complete-lifecycle-action 命令以继续。
- 发生缩减事件时，生命周期挂钩会在实例终止之前将其暂停，并使用 Amazon 向您发送通知。EventBridge当实例处于等待状态时，您可以在实例完全终止之前调用 Amazon Lambda 函数或连接到该实例来下载日志或其他数据。

生命周期钩子的一个常见用途是控制何时在 Elastic Load Balancing 中注册实例。通过向您的 Auto Scaling 组添加启动生命周期钩子，您可以确保引导启动脚本已成功完成，并且实例上的应用程序在生命周期钩子结束时准备好接受流量。

内容

- 生命周期挂钩在自动扩缩组中如何工作
- 做好准备向 Auto Scaling 组添加生命周期钩子
- 通过实例元数据检索目标生命周期状态
- 向自动扩缩组添加生命周期挂钩
- 在自动扩缩组中完成生命周期操作
- 教程：使用数据脚本和实例元数据检索生命周期状态
- 教程：配置调用 Lambda 函数的生命周期钩子

生命周期钩子可用性

下表列出了可用于各种方案的生命周期钩子。

事件	实例启动或终止[1]	最大实例生命周期：替换实例	实例刷新：替换实例	容量再平衡：替换实例	温水池：进入和离开温水池的实例
实例启动	✓	✓	✓	✓	✓
实例终止	✓	✓	✓	✓	✓

[1] 适用于所有启动和终止，无论是自动启动还是手动启动，例如当您调用 `SetDesiredCapacity` 或 `TerminateInstanceInAutoScalingGroup` 操作时。当您附加或分离实例、将实例移入或移出备用模式或使用强制删除选项删除组时，不适用。

生命周期钩子的注意事项和限制

操作生命周期挂钩时，请记住以下注意事项和限制：

- Amazon A EC2 uto Scaling 提供了自己的生命周期来帮助管理 Auto Scaling 群组。此生命周期与其他 EC2 实例的生命周期不同。有关更多信息，请参阅 Amazon A EC2 uto Scaling 实例生命周期。暖池中的实例也有自己的生命周期，如 暖池中实例的生命周期状态转换 中所述。
- 您可以将生命周期钩子与竞价型实例一起使用，但生命周期钩子并不禁止在容量不再可用的情况下终止实例，这种情况会随时发生，并显示两分钟中断通知。有关更多信息，请参阅 Amazon EC2 用户指南中的竞价型实例中断。但是，您可以启用容量再平衡以主动替换已从 Amazon Spot 服务收到再

平衡建议的 EC2 竞价型实例，该信号是在竞价型实例中断风险较高时发送的。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

- 实例可以在有限的时间里保持等待状态。生命周期钩子的默认超时时间为一小时（检测信号超时时间）。此外，还有一个全局超时时间，它指定您可以将实例保持在等待状态的最长时间。全局超时时间为 48 小时或检测信号超时时间的 100 倍，以较小者为准。
- 生命周期挂钩的结果可以是放弃或继续。如果实例正在启动，则继续表示您的操作已成功，并且 Amazon A EC2 uto Scaling 可以将该实例投入使用。否则，“放弃”指示您的自定义操作未成功，并且可终止并替代实例。如果实例正在终止，放弃和继续都允许终止实例。不过，放弃将停止任何剩余操作（例如，其他生命周期钩子），而继续将允许完成任何其他生命周期钩子。
- 如果生命周期挂钩持续失败，Amazon A EC2 uto Scaling 会限制其允许实例启动的速率，因此请务必测试并修复生命周期操作中的任何永久性错误。
- 使用 Amazon CLI Amazon CloudFormation、或 SDK 创建和更新生命周期挂钩提供了从中创建生命周期挂钩时不可用的选项 Amazon Web Services Management Console。例如，用于指定 SNS 主题或 SQS 队列的 ARN 的字段不会出现在控制台中，因为 Amazon A EC2 uto Scaling 已经向亚马逊发送了事件。 EventBridge可以根据需要筛选这些事件并将其重定向到 Lambda、Amazon SNS 和亚马逊 SQS 等 Amazon 服务。
- 您可以在创建 Auto Scaling 群组时向该组添加多个生命周期挂钩，方法是使用 Amazon CLI Amazon CloudFormation、或 SDK 调用 CreateAutoScalingGroupAPI。但是，如果指定，每个钩子必须具有相同的通知目标和 IAM 角色。要创建具有不同通知目标和不同角色的生命周期挂钩，请在对 PutLifecycleHookAPI 的单独调用中逐个创建生命周期挂钩。
- 如果您为实例启动添加了生命周期挂钩，那么运行状况检查宽限期将在实例达到 `InService` 状态时立即开始。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

扩展注意事项

- 动态扩展策略会根据跨多个实例聚合的 CloudWatch 指标数据（例如 CPU 和网络 I/O）向内和向外扩展。扩展时，Amazon A EC2 uto Scaling 不会立即将新实例计入 Auto Scaling 组的聚合实例指标。它会一直等到实例达到 `InService` 状态并且实例预热完成时再计入。有关默认实例预热主题的更多信息，请参阅 扩缩性能注意事项。
- 在横向缩减时，聚合实例指标可能无法立即反映出终止实例的移除情况。在 Amazon A EC2 uto Scaling 终止工作流程开始后不久，终止的实例将停止计入该组的聚合实例指标。
- 如果调用生命周期挂钩，由简单扩缩策略引发的扩缩活动将暂停，直至生命周期操作完成并且冷却时间过期。为冷却时间设置较长的时间间隔意味着，恢复扩展将会需要更长的时间。有关更多信息，请参阅冷却主题中的生命周期挂钩可能会导致额外的延迟。通常，如果您可以改用步进扩缩策略或目标跟踪扩缩策略，我们建议不要使用简单扩缩策略。

相关资源

有关介绍视频，请参阅 re Amazon : Invent 2018 : 开启 Amazon A EC2 uto Scaling 让容量管理变得简单。YouTube

我们提供了几个 JSON 和 YAML 模板片段，您可以使用它们来了解如何在 Amazon CloudFormation 堆栈模板中声明生命周期挂钩。有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的AWS::AutoScaling::LifecycleHook参考资料。

您也可以访问我们的GitHub存储库，下载生命周期挂钩的示例模板和用户数据脚本。

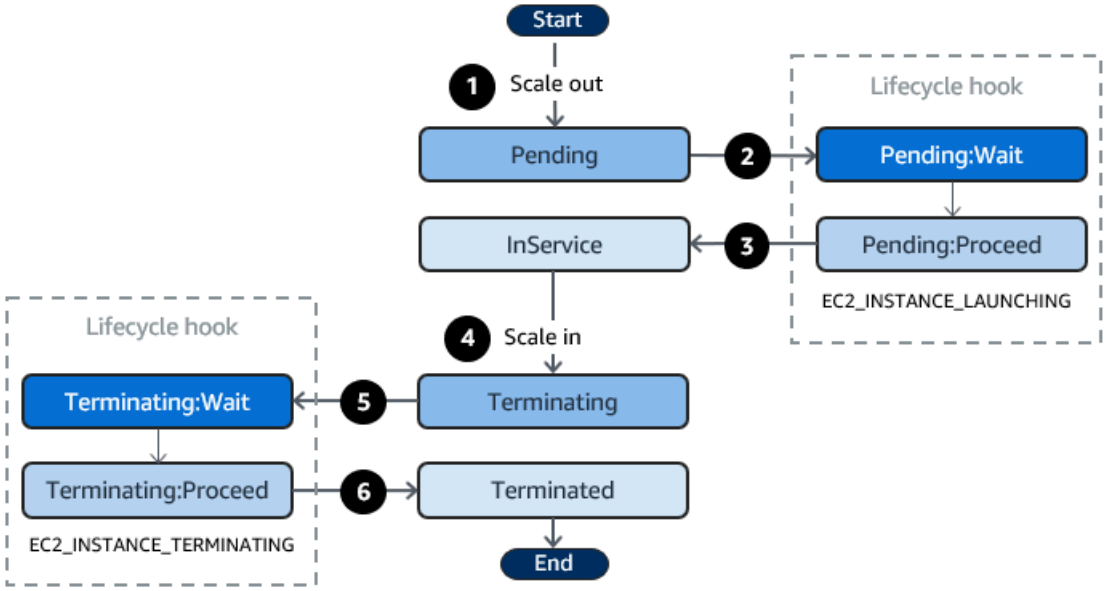
有关生命周期挂钩的使用示例，请参阅以下博客帖子。

- 使用 Lambda 和 Ama EC2 zon 运行命令为扩展实例构建备份系统
- 在终止 EC2 Auto Scaling 实例之前运行代码。

生命周期挂钩在自动扩缩组中如何工作

从启动到终止，Amazon EC2 实例会经历不同的状态。您可以为您的自动扩缩组创建自定义操作，以便在实例因生命周期挂钩而转换到等待状态时执行操作。

下图显示使用生命周期挂钩进行横向扩展和横向缩减时，Auto Scaling 实例状态之间的转换。



如上图中所示：

1. Auto Scaling 组响应向外扩展事件并开始启动实例。

2. 生命周期钩子将实例置于等待状态 (Pending:Wait)，然后执行自定义操作。

   实例将保持等待状态，直到您完成生命周期操作，或者直到超时时段结束。预设情况下，实例将保持等待状态 1 小时，然后 Auto Scaling 组继续启动过程 (Pending:Proceed)。如果您需要更长时间，可通过记录检测信号来重新开始超时时段。如果您在自定义操作已完成且超时时段尚未到期时完成生命周期操作，则该时间段结束，Auto Scaling 组将继续启动过程。

3. 实例进入 InService 状态并开始运行状况检查宽限期。但是，在实例达到 InService 状态之前，如果 Auto Scaling 组与 Elastic Load Balancing 负载均衡器关联，则实例将在负载均衡器中注册，然后负载均衡器开始检查其运行状况。运行状况检查宽限期结束后，Amazon A EC2 uto Scaling 开始检查实例的运行状况。

4. Auto Scaling 组响应缩减事件并开始终止实例。如果 Auto Scaling 组与 Elastic Load Balancing 一起使用，则终止实例要首先从负载均衡器中取消注册。如果为负载均衡器启用了 Connection Draining，则实例将停止接受新连接并等待现有连接耗尽，然后再完成取消注册过程。

5. 生命周期钩子将实例置于等待状态 (Terminating:Wait)，然后执行自定义操作。

   该实例将保持等待状态，直到您完成生命周期操作，或者直至超时期结束（预设情况下为 1 小时）。完成生命周期钩子或超时时段过期后，实例将转换到下一个状态 (Terminating:Proceed)。

6. 实例已终止。

> **Important**
>
> 暖池中的实例也有自己的生命周期以及相对应的等待状态，如 暖池中实例的生命周期状态转换 中所述。

做好准备向 Auto Scaling 组添加生命周期钩子

在向 Auto Scaling 组添加生命周期钩子之前，请确保正确用户数据脚本或通知目标。

- 要在实例启动时运行用户数据脚本以对实例执行自定义操作，无需配置通知目标。但是，您必须已创建指定用户数据脚本并将其与 Auto Scaling 组关联的启动模板或启动配置。有关用户数据脚本的更多信息，请参阅 Amazon EC2 用户指南中的 Linux 实例启动时运行命令。
- 要在生命周期操作完成时向 Amazon A EC2 uto Scaling 发送信号，您必须在脚本中添加 CompleteLifecycleActionAPI 调用，并且必须手动创建一个 IAM 角色，其策略允许 Auto Scaling 实例调用此 API。您的启动模板或启动配置必须使用启动时附加到您的 Amazon 实例的 IAM EC2 实

例配置文件来指定此角色。有关更多信息，请参阅在自动扩缩组中完成生命周期操作 和适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。

- 要使用诸如 Lambda 之类的服务执行自定义操作，您必须已创建 EventBridge 规则并指定一个 Lambda 函数作为其目标。有关更多信息，请参阅 为生命周期通知配置通知目标。
- 要允许 Lambda 在生命周期操作完成时向 Amazon A EC2 uto Scaling 发出信号，您必须在函数代码中添加 CompleteLifecycleActionAPI 调用。您还必须将 IAM policy 附加到函数的执行角色，以授予 Lambda 完成生命周期操作的权限。有关更多信息，请参阅 教程：配置调用 Lambda 函数的生命周期钩子。
- 要使用 Amazon SNS 或 Amazon SQS 之类的服务执行自定义操作，您必须已创建 SNS 主题或 SQS 队列并准备好其 Amazon Resource Name (ARN)。您还必须已经创建了 IAM 角色，该角色允许 Amazon A EC2 uto Scaling 访问您的 SNS 主题或 SQS 目标，并已准备好其 ARN。有关更多信息，请参阅 为生命周期通知配置通知目标。

> **Note**
>
> 默认情况下，当您在控制台中添加生命周期挂钩时，Amazon A EC2 uto Scaling 会向亚马逊发送生命周期事件通知 EventBridge。建议使用 EventBridge 或用户数据脚本。要创建直接向 Amazon SNS 或 Amazon SQS 发送通知的生命周期挂钩，请使用 Amazon CLI Amazon CloudFormation、或软件开发工具包添加生命周期挂钩。

为生命周期通知配置通知目标

您可以向 Auto Scaling 组添加生命周期钩子，以便在实例进入等待状态时执行自定义操作。您可以选择目标服务，以根据您的首选开发方法执行这些操作。

第一种方法使用 Amazon EventBridge 来调用 Lambda 函数来执行您想要的操作。第二种方法是创建发布通知的 Amazon Simple Notification Service (Amazon SNS) 主题。客户端可以订阅 SNS 主题并使用支持的协议接收已发布的消息。最后一种方法涉及到使用 Amazon Simple Queue Service (Amazon SQS)，它是分布式应用程序用于通过轮询模型交换消息的消息收发系统。

作为最佳实践，我们建议您使用 EventBridge。发送到亚马逊 SNS 和 Amazon SQS 的通知包含的信息与 Amazon A EC2 uto Scaling 发送到的通知相同。 EventBridge以前 EventBridge，标准做法是向 SNS 或 SQS 发送通知，然后将其他服务与 SNS 或 SQS 集成以执行编程操作。如今，它 EventBridge 为您提供了更多可以定位的服务选项，并使使用无服务器架构更轻松地处理事件。

以下过程介绍了如何设置通知目标。

请记住，如果您的启动模板或启动配置中有在实例启动时配置实例的用户数据脚本，则无需接收通知即可对实例执行自定义操作。

内容

> **Important**
>
> 与生命周期挂钩一起使用的 EventBridge 规则、Lambda 函数、Amazon SNS 主题和 Amazon SQS 队列必须始终位于您创建 Auto Scaling 组的同一区域。

使用将通知发送到 Lambda EventBridge

您可以配置 EventBridge 规则，以便在实例进入等待状态时调用 Lambda 函数。Amazon A EC2 uto Scaling 会向其发送 EventBridge 有关正在启动或终止的实例的生命周期事件通知，以及一个可用于控制生命周期操作的令牌。有关这些事件的示例，请参阅 Amazon A EC2 uto Scaling 事件参考。

> **Note**
>
> 当您使用创建事件规则时，控制台会自动添加授予 EventBridge 调用 Lambda 函数的权限所必需的 IAM 权限。 Amazon Web Services Management Console 如果您使用 Amazon CLI创建事件规则，则需要明确授予此权限。
> 有关如何在 EventBridge 控制台中创建事件规则的信息，请参阅《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的 Amazon EventBridge 规则。
> - 或者 -
> 有关面向控制台用户的入门教程，请参阅教程：配置调用 Lambda 函数的生命周期钩子。本教程向您展示如何创建一个简单的 Lambda 函数，该函数用于监听启动事件并将其写入日志中。CloudWatch

创建调用 Lambda 函数的 EventBridge 规则

1. 通过使用 Lambda 控制台创建 Lambda 函数，并记下其 Amazon Resource Name (ARN)。例如，arn:aws:lambda:*region*:*123456789012*:function:*my-function*。您需要使用 ARN 来创建目标。 EventBridge 有关更多信息，请参阅 Amazon Lambda 开发人员指南中的 Lambda 入门。
2. 要创建匹配实例创建事件的规则，请使用以下 put-rule 命令。

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state
 ENABLED
```

以下示例显示了实例启动生命周期操作的 pattern.json。将中的*italics*文本替换为您的 Auto Scaling 组的名称。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ]
   }
}
```

如果命令成功运行，则使用 EventBridge 规则的 ARN 进行响应。记下此 ARN。您需要在第 4 步中输入此信息。

要创建与其他事件匹配的规则，请修改事件模式。有关更多信息，请参阅 用于处理 EventBridge Auto Scaling 事件。

3. 要指定要用作规则目标的 Lambda 函数，请使用以下 put-targets 命令。

```
aws events put-targets --rule my-rule --targets
 Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

在前面的命令中，*my-rule*是您在步骤 2 中为规则指定的名称，Arn参数的值是您在步骤 1 中创建的函数的 ARN。

4. 要添加允许规则调用您的 Lambda 函数的权限，请使用以下 Lambda add-permission 命令。此命令将 EventBridge 服务主体 (events.amazonaws.com) 和范围权限信任到指定规则。

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
```

```
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
 arn:aws:events:region:123456789012:rule/my-rule
```

在上述命令中：

- *my-function*是您希望规则用作目标的 Lambda 函数的名称。
- *my-unique-id*是您定义的唯一标识符，用于描述 Lambda 函数策略中的语句。
- source-arn是规则的 ARN。 EventBridge

如果命令成功运行，则您将收到类似于以下内容的输出：

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
         \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

Statement 值是已添加到 Lambda 函数策略的语句的 JSON 字符串版本。

5. 在遵循以上说明操作后，继续向自动扩缩组添加生命周期挂钩 作为下一步。

使用 Amazon SNS 接收通知

您可以使用 Amazon SNS 设置通知目标（SNS 主题），以便在生命周期操作发生时接收通知。然后，Amazon SNS 将通知发送给订阅的收件人。确认订阅前，向主题发布的通知不会发送至收件人。

使用 Amazon SNS 设置通知

1. 通过使用 Amazon SNS 控制台或以下 create-topic 命令创建 Amazon SNS 主题。确保该主题与您使用的 Auto Scaling 组位于同一区域。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的 Amazon SNS 入门。

```
aws sns create-topic --name my-sns-topic
```

2. 记录主题 Amazon Resource Name (ARN)，例如 `arn:aws:sns:region:123456789012:my-sns-topic`。您需要它来创建生命周期钩子。
3. 创建 IAM 服务角色以授予 Amazon A EC2 uto Scaling 访问您的亚马逊 SNS 通知目标的权限。

   让 Amazon A EC2 uto Scaling 访问你的 SNS 话题

   a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在左侧的导航窗格中，选择角色。
   c. 选择 Create role（创建角色）。
   d. 对于选择可信实体，选择 Amazon 服务。
   e. 对于您的用例，在其他 Amazon 服务的用例下，选择 Auto Scaling，然后选择 EC2 Aut EC2 o Scaling 通知访问权限。
   f. 选择 Next（下一步）两次，以前往 Name, review, and create（命名、检查和创建）页面。
   g. 对于 Role name（角色名称），输入角色的名称（例如 **my-notification-role**），然后选择 Create role（创建角色）。
   h. 在 Roles (角色) 页面中，选择刚刚创建的角色以打开 Summary (摘要) 页面。记下角色的 ARN。例如，`arn:aws:iam::123456789012:role/my-notification-role`。您需要它来创建生命周期钩子。
4. 在遵循以上说明操作后，继续添加生命周期钩子 (Amazon CLI) 作为下一步。

使用 Amazon SQS 接收通知

您可以使用 Amazon SQS 设置通知目标以便在生命周期操作开始时接收消息。然后，队列使用者必须轮询 SQS 队列，以便对这些通知执行操作。

> **Important**
>
> FIFO 队列与生命周期挂钩不兼容。

使用 Amazon SQS 设置通知

1. 使用 Amazon SQS 控制台创建 Amazon SQS 队列。确保队列与您正在使用的 Auto Scaling 组位于同一个区域。有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的 Amazon SQS 入门。

2. 记录队列 ARN，例如 `arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`。您需要它来创建生命周期钩子。
3. 创建 IAM 服务角色以授予亚马逊 A EC2 uto Scaling 访问您的亚马逊 SQS 通知目标的权限。

   让 Amazon A EC2 uto Scaling 访问你的 SQS 队列

   a. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在左侧的导航窗格中，选择角色。
   c. 选择 Create role（创建角色）。
   d. 对于选择可信实体，选择 Amazon 服务。
   e. 对于您的用例，在其他 Amazon 服务的用例下，选择 Auto Scaling，然后选择 EC2 Aut EC2 o Scaling 通知访问权限。
   f. 选择 Next（下一步）两次，以前往 Name, review, and create（命名、检查和创建）页面。
   g. 对于 Role name（角色名称），输入角色的名称（例如 **my-notification-role**），然后选择 Create role（创建角色）。
   h. 在 Roles (角色) 页面中，选择刚刚创建的角色以打开 Summary (摘要) 页面。记下角色的 ARN。例如，`arn:aws:iam::123456789012:role/my-notification-role`。您需要它来创建生命周期钩子。
4. 在遵循以上说明操作后，继续添加生命周期钩子 (Amazon CLI) 作为下一步。

Amazon SNS 和 Amazon SQS 的通知消息示例

当实例处于等待状态时，将向 Amazon SNS 或 Amazon SQS 通知目标发布消息。消息包含以下信息：

- `LifecycleActionToken` — 生命周期操作令牌。
- `AccountId`— Amazon Web Services 账户 身份证。
- `AutoScalingGroupName` — Auto Scaling 组的名称。
- `LifecycleHookName` — 生命周期钩子的名称。
- `EC2InstanceId`— EC2 实例的 ID。
- `LifecycleTransition` — 生命周期钩子类型。
- `NotificationMetadata` — 通知元数据。

以下是通知消息示例。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

测试通知消息示例

首次添加生命周期钩子时，将向通知目标发布测试通知消息。以下是测试通知消息示例。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

> **Note**
>
> 有关从 Amazon A EC2 uto Scaling 发送到的事件的示例 EventBridge，请参阅Amazon A EC2 uto Scaling 事件参考。

通过实例元数据检索目标生命周期状态

您启动的每个 Auto Scaling 实例都经历多个生命周期状态。如需从实例内调用作用于特定生命周期状态转换的自定义操作，则您必须通过实例元数据检索目标生命周期状态。

例如，您可能需要一种从实例内部检测实例终止的机制，以便在实例终止之前在实例上运行一些代码。为此，您可以编写代码，直接从实例中轮询实例的生命周期状态。然后，您可以向自动扩缩组添加生命周期挂钩，以保持实例运行，直到您的代码发送 complete-lifecycle-action 命令以继续。

Auto Scaling 实例生命周期有两个主要的稳定状态：InService 和 Terminated，还有两个附加的稳定状态：Detached 和 Standby。如果使用暖池，生命周期还有四个额外的稳定状态：Warmed:Hibernated、Warmed:Running、Warmed:Stopped 和 Warmed:Terminated。

当实例准备过渡到上述稳定状态之一时，Amazon A EC2 uto Scaling 会更新实例元数据项目的值autoscaling/target-lifecycle-state。要从实例内获取目标生命周期状态，必须使用实例元数据服务从实例元数据中检索它。

> **Note**
>
> 实例元数据是有关 Amazon EC2 实例的数据，应用程序可以使用这些数据来查询实例信息。实例元数据服务是实例上的组件，本地代码用它来访问实例元数据。本地代码可以包括实例上运行的用户数据脚本或应用程序。

本地代码可以使用以下两种方法之一从正在运行的实例访问实例元数据：实例元数据服务版本 1 (IMDSv1) 或实例元数据服务版本 2 (IMDSv2)。 IMDSv2使用面向会话的请求并缓解多种类型的漏洞，这些漏洞可用于尝试访问实例元数据。有关这两种方法的详细信息，请参阅《Amazon EC2 用户指南》 IMDSv2中的 “使用”。

IMDSv2

```
[ec2-user ~]$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

IMDSv1

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

下面是示例输出。

```
InService
```

目标生命周期状态是实例转换到的状态。当前生命周期状态是实例所处的状态。在生命周期操作完成并且实例完成向目标生命周期状态的转换之后，这两个状态可能是相同的。您无法从实例元数据中检索实例的当前生命周期状态。

Amazon A EC2 uto Scaling 于 2022 年 3 月 10 日开始生成目标生命周期状态。如果您的实例在该日期之后转换为其中一个目标生命周期状态，则您的实例元数据中将显示该目标生命周期状态项目。否则，它不存在，并且您会收到 HTTP 404 错误。

有关检索实例元数据的更多信息，请参阅 Amazon EC2 用户指南中的检索实例元数据。

有关向您展示如何在使用目标生命周期状态的用户数据脚本中使用自定义操作创建生命周期钩子的教程，请参阅 教程：使用数据脚本和实例元数据检索生命周期状态。

> **Important**
>
> 为确保您可以尽快调用自定义操作，您的本地代码应经常轮询 IMDS，并在出现错误时重试。

向自动扩缩组添加生命周期挂钩

要将 Auto Scaling 实例置于等待状态并对它们执行自定义操作，您可以向 Auto Scaling 组添加生命周期钩子。自定义操作将在实例启动时或其终止之前执行。实例将保持等待状态，直到您完成生命周期操作，或者直到超时时段结束。

从创建 Auto Scaling 组后 Amazon Web Services Management Console，您可以向该组添加一个或多个生命周期挂钩，总共不超过 50 个生命周期挂钩。您还可以在创建 Auto Scaling 组时使用 Amazon CLI Amazon CloudFormation、或 SDK 向 Auto Scaling 组添加生命周期挂钩。

默认情况下，当您在控制台中添加生命周期挂钩时，Amazon A EC2 uto Scaling 会向亚马逊发送生命周期事件通知 EventBridge。建议使用 EventBridge 或用户数据脚本。要创建直接向 Amazon SNS 或 Amazon SQS 发送通知的生命周期挂钩，您可以使用命令，如put-lifecycle-hook本主题中的示例所示。

内容

添加生命周期钩子（控制台）

请按照以下步骤向您的自动扩缩组添加生命周期挂钩。要添加用于横向扩展（实例启动）和横向缩减（实例终止或返回至暖池）的生命周期挂钩，您必须创建两个单独的钩子。

如 做好准备向 Auto Scaling 组添加生命周期钩子 中所述，请在开始之前确认已设置自定义操作。

为横向扩展添加生命周期挂钩

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。这时将在页面底部打开一个拆分窗格。
3. 在 Instance management (实例管理) 选项卡的 Lifecycle hooks (生命周期挂钩) 中，选择 Create lifecycle hook (创建生命周期挂钩)。
4. 要为横向扩展（实例启动）定义生命周期挂钩，请执行以下操作：

   a. 对于 Lifecycle hook name (生命周期挂钩名称)，请指定生命周期挂钩的名称。

   b. 对于生命周期转换，请选择实例启动。

   c. 对于检测信号超时时间，请在钩子超时之前，指定实例在横向扩展时保持等待状态的时长（以秒为单位）。范围从 30 到 7200 秒。设置的超时时段越长，完成您的自定义操作的时间越长。然后，如果您在超时期限结束之前完成，请发送complete-lifecycle-action命令以允许实例进入下一个状态。

   d. 对于默认结果，指定在生命周期钩子超时结束或发生意外故障时要执行的操作。您可以选择继续或放弃。

      - 如果您选择继续，自动扩缩组可以继续执行任何其他生命周期挂钩，然后将实例投入使用。
      - 如果选择 放弃，自动扩缩组停止任何剩余操作并立即终止实例。

   e. （可选）对于通知元数据，请指定在 Amazon A EC2 uto Scaling 向通知目标发送消息时要包含的其他信息。

5. 选择创建。

为横向缩减添加生命周期挂钩

1. 选择创建生命周期挂钩，以从为横向扩展创建生命周期挂钩而中断的地方继续操作。
2. 要为横向缩减定义生命周期挂钩（实例终止或返回到暖池），请执行以下操作：

   a. 对于 Lifecycle hook name (生命周期挂钩名称)，请指定生命周期挂钩的名称。

   b. 对于生命周期转换，请选择实例终止。

   c. 对于检测信号超时时间，请在钩子超时之前，指定实例在横向扩展时保持等待状态的时长（以秒为单位）。我们建议将超时时间缩短30为120几秒，具体取决于执行任何最终任务（例如从中提取 EC2 日志）所需的时间 CloudWatch。

d. 对于 Default result（默认结果），请指定超时结束或发生意外故障时 Auto Scaling 组执行的操作。ABANDON（放弃）和 CONTINUE（继续）都允许终止实例。

- 如果您选择 CONTINUE（继续），Auto Scaling 组可以在终止之前继续执行任何剩余操作（如其他生命周期钩子）。
- 如果选择 放弃，自动扩缩组将立即终止实例。

e. （可选）对于通知元数据，请指定在 Amazon A EC2 uto Scaling 向通知目标发送消息时要包含的其他信息。

3. 选择创建。

添加生命周期钩子 (Amazon CLI)

使用 put-lifecycle-hook 命令创建和更新生命周期挂钩。

要执行扩展操作，请使用以下命令。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

要执行缩减操作，请使用以下命令。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

要使用 Amazon SNS 或 Amazon SQS 接收通知，请添加 `--notification-target-arn` 和 `--role-arn` 选项。

以下示例中创建了一个生命周期钩子，用于指定名为的 *my-sns-topic* 的 SNS 主题作为通知目标。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \
  --notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \
  --role-arn arn:aws:iam::123456789012:role/my-notification-role
```

该主题将使用以下键/值对接收测试通知。

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

默认情况下，该put-lifecycle-hook命令会创建心跳超时为3600秒（一小时）的生命周期挂钩。

要更改现有生命周期钩子的检测信号超时时间，请添加 `--heartbeat-timeout` 选项，如以下示例所示。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```

如果实例已经处于等待状态，则可以使用 record-lifecycle-action-heartbeatCLI 命令记录心跳来防止生命周期挂钩超时。这会将超时时间增加到您创建生命周期挂钩时指定的超时值。如果您在超时期限结束之前完成，则可以发送 complete-lifecycle-actionCLI 命令以允许实例进入下一个状态。有关更多信息以及示例，请参阅 在自动扩缩组中完成生命周期操作。

在自动扩缩组中完成生命周期操作

当 Auto Scaling 组响应生命周期事件时，它会将实例置于等待状态并发送事件通知。当实例处于等待状态时，您可以执行自定义操作。

如果您在超时周期过期之前完成生命周期操作，那么以 CONTINUE 的结果完成生命周期操作会很有帮助。如果您未完成生命周期操作，则生命周期挂钩将在超时周期结束后进入您为默认结果指定的状态。

内容

- 完成生命周期操作（手动）
- 完成生命周期操作（自动）

完成生命周期操作（手动）

以下过程适用于命令行界面，在控制台中不受支持。必须替换的信息（如实例 ID 或 Auto Scaling 组的名称）以斜体显示。

完成生命周期操作 (Amazon CLI)

1. 如果需要更多时间以完成自定义操作，请使用 record-lifecycle-action-heartbeat 命令重新开始超时时段，并将实例保持为等待状态。例如，如果超时时段为一小时，而您在 30 分钟后调用该命令，则实例将继续保持等待状态一小时（总共为 90 分钟）。

您可以指定随通知一起接收的生命周期操作令牌，如以下命令所示。

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
  --auto-scaling-group-name my-asg --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

或者，您可以指定随通知一起接收的实例的 ID，如以下命令所示。

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
  --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
```

2. 如果您在超时期限结束之前完成了自定义操作，请使用complete-lifecycle-action命令以便 Auto Scaling 组可以继续启动或终止实例。您可以指定生命周期操作令牌，如以下命令所示。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
  --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

或者，您可以指定实例的 ID，如以下命令所示。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
  --instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \
  --auto-scaling-group-name my-asg
```

完成生命周期操作（自动）

如果您拥有在实例启动后配置这些实例的用户数据脚本，则无需手动完成生命周期操作。您可以将complete-lifecycle-action命令添加到脚本中。该脚本可以从实例元数据中检索实例 ID，并在引导脚本成功完成时向 Amazon A EC2 uto Scaling 发出信号。

如果您尚未这样做，请更新脚本，从实例元数据中检索实例的实例 ID。有关更多信息，请参阅 Amazon EC2 用户指南中的检索实例元数据。

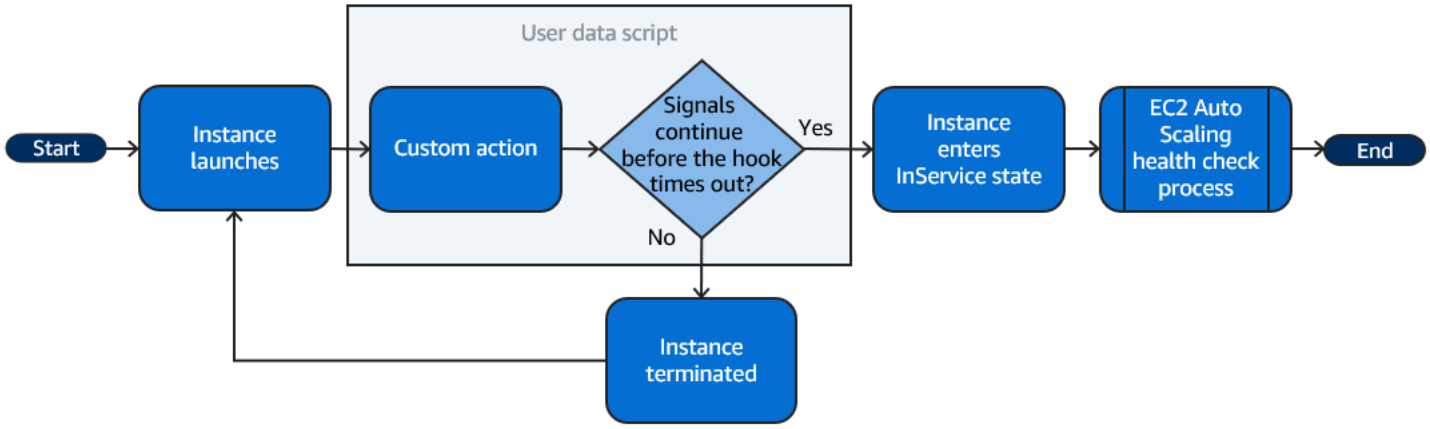
如果您使用 Lambda，则还可以在函数的代码中设置回调，以便在自定义操作成功时能让实例的生命周期继续。有关更多信息，请参阅 教程：配置调用 Lambda 函数的生命周期钩子。

教程：使用数据脚本和实例元数据检索生命周期状态

为生命周期挂钩创建自定义操作的一种常见方法是使用 Amazon A EC2 uto Scaling 发送到其他服务（例如亚马逊）的通知 EventBridge。但是，您可以避免创建额外的基础设施，方法是使用用户数据脚本将配置实例并完成生命周期操作的代码移动到实例本身中。

以下教程介绍如何开始使用用户数据脚本和实例元数据。您可以使用读取您组中实例的目标周期状态并在实例生命周期的特定阶段执行回调操作以继续启动过程的用户数据脚本来创建基本 Auto Scaling 组配置。

下图总结了使用用户数据脚本执行自定义操作时横向扩展事件的流程。实例启动后，实例的生命周期将暂停，直到生命周期挂钩完成，要么是超时，要么是 Amazon A EC2 uto Scaling 收到继续运行的信号。



内容

步骤 1：创建具有完成生命周期操作权限的 IAM 角色

当您使用 Amazon CLI 或 Amazon 软件开发工具包发送回调以完成生命周期操作时，必须使用具有完成生命周期操作权限的 IAM 角色。

创建策略

1. 打开 IAM 控制台的策略页面，然后选择创建策略。
2. 请选择 JSON 选项卡。
3. 在策略文档框中，将以下策略文档复制并粘贴到框中。将*sample text*替换为您的账号和您要创建的 Auto Scaling 群组的名称 (**TestAutoScalingEvent-group**)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/
TestAutoScalingEvent-group"
    }
  ]
}
```

4. 选择下一步。
5. 对于 Policy name，输入 **TestAutoScalingEvent-policy**。选择创建策略。

完成创建策略之后，您可以创建一个使用该策略的角色。

创建角色

1. 在左侧的导航窗格中，选择角色。
2. 选择 Create role（创建角色）。
3. 对于选择可信实体，选择 Amazon 服务。
4. 对于您的用例，请选择，EC2然后选择 “下一步”。
5. 在 “添加权限” 下，选择您创建的策略（TestAutoScalingEvent-policy）。然后选择下一步。
6. 在 Name, review, and create（命名、检查并创建）页面上，对于 Role name（角色名称），输入 **TestAutoScalingEvent-role**，然后选择 Create role（创建角色）。

步骤 2：创建启动模板并包含 IAM 角色和用户数据脚本

创建用于 Auto Scaling 组的启动模板。包含您创建的 IAM 角色和提供的示例用户数据脚本。

创建启动模板

1. 打开 Amazon EC2 控制台的启动模板页面。
2. 选择Create launch template（创建启动模板）。
3. 对于 Launch template name（启动模板名称），输入 **TestAutoScalingEvent-template**。
4. 在 Auto Scaling guidance（Auto Scaling 指导）下，选中复选框。
5. 对于 Application and OS Images (Amazon Machine Image) [应用程序和操作系统镜像（Amazon Machine Image）]，请从 Quick Start（快速启动）列表中选择 Amazon Linux 2 (HVM)、SSD Volume Type（SSD 卷类型）、64-bit (x86) [64 位（x86）]。
6. 对于实例类型，选择亚马逊 EC2 实例的类型（例如，“t2.micro”）。
7. 对于高级详细信息，展开该节以查看字段。
8. 对于 IAM 实例配置文件，请选择您的 IAM 角色（TestAutoScalingEvent- role）的 IAM 实例配置文件名称。实例配置文件是 IAM 角色的容器，它允许 Amazon EC2 在实例启动时将 IAM 角色传递给该实例。

   使用 IAM 控制台创建 IAM 角色时，控制台自动创建实例配置文件，按相应的角色为文件命名。
9. 对于 User data（用户数据），将下面的实例用户数据脚本粘贴到字段。将的示例文本替换为您`group_name`要创建的 Auto Scaling 组的`region`名称以及 Amazon Web Services 区域 您想让 Auto Scaling 组使用的名称。

```
#!/bin/bash

function get_target_state {
    echo $(curl -s http://169.254.169.254/latest/meta-data/autoscaling/target-
lifecycle-state)
}

function get_instance_id {
    echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
```

```
    region='us-west-2'

    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
      --lifecycle-hook-name TestAutoScalingEvent-hook \
      --auto-scaling-group-name $group_name \
      --lifecycle-action-result CONTINUE \
      --instance-id $instance_id \
      --region $region)
}

function main {
    while true
    do
        target_state=$(get_target_state)
        if [ \"$target_state\" = \"InService\" ]; then
            # Change hostname
            export new_hostname="${group_name}-$instance_id"
            hostname $new_hostname
            # Send callback
            complete_lifecycle_action
            break
        fi
        echo $target_state
        sleep 5
    done
}

main
```

此简单用户数据脚本将执行以下操作：

- 调用实例元数据以从实例元数据中检索目标生命周期状态和实例 ID
- 重复检索目标生命周期状态，直到它变为 `InService`
- 如果目标生命周期状态为 `InService`，则将实例的主机名更改为实例 ID 前面加上 Auto Scaling 组名称
- 通过调用 complete-lifecycle-action CLI 命令向 Amazon A EC2 uto Scaling 发送 EC2 启动过程`CONTINUE`的信号，发送回调

10. 选择Create launch template（创建启动模板）。
11. 在确认页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

> **Note**
>
> 有关可用作开发用户数据脚本的参考的其他示例，请参阅 Amazon A EC2 uto Scaling 的GitHub 存储库。

步骤 3：创建 Auto Scaling 组

创建启动模板后，创建 Auto Scaling 组。

创建 自动扩缩组

1. 在 Choose launch template or configuration（选择启动模板或配置）页面上，对于 Auto Scaling group name（Auto Scaling 组名称），输入 Auto Scaling 组的名称 (**`TestAutoScalingEvent-group`**)。
2. 选择 Next（下一步）转至 Choose instance launch options（选择实例启动选项）页面。
3. 对于 Network（网络），选择 VPC。
4. 对于 Availability Zones and subnets（可用区和子网），请从一个或多个可用区中选择一个或多个子网。
5. 在 Instance type requirements（实例类型要求）部分中，使用默认设置简化此步骤。（请勿覆盖启动模板。）在本教程中，您将仅使用启动模板中指定的实例类型启动一个按需型实例。
6. 在屏幕的底部选择 Skip to review（跳至审核）。
7. 在 Review（审核）页面中，检查 Auto Scaling 组的详细信息，然后选择 Create Auto Scaling group（创建 Auto Scaling 组）。

步骤 4：添加生命周期钩子

添加生命周期钩子以将实例保持在等待状态，直到您的生命周期操作完成。

添加生命周期挂钩

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。这时将在页面底部打开一个拆分窗格。
3. 在下方窗格中，在实例管理选项卡的生命周期钩子中，选择创建生命周期钩子。
4. 要为横向扩展（实例启动）定义生命周期挂钩，请执行以下操作：

a. 对于生命周期钩子名称，请输入 **TestAutoScalingEvent-hook**。

b. 对于生命周期转换，请选择实例启动。

c. 对于 Heartbeat timeout（检测信号超时），输入 **300** 以获取等待用户数据脚本回调的秒数。

d. 对于默认结果，请选择放弃。如果钩子超时而未收到来自您用户数据脚本的回调，Auto Scaling 组将终止新实例。

e. （可选）将 Notification metadata（通知元数据）留空。

5. 选择创建。

步骤 5：测试和验证功能

要测试功能，请通过将 Auto Scaling 组的所需容量增加 1 来更新 Auto Scaling 组。用户数据脚本运行并在实例启动后立即开始检查实例的目标生命周期状态。当目标生命周期状态为 `InService` 时，脚本会更改主机名并发送回调操作。完成此操作通常仅需要几秒钟时间。

增加您的 Auto Scaling 组的大小

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。在下方窗格中查看详细信息，同时仍然可以看到上方窗格的顶部几行。
3. 在下方窗格中的详细信息选项卡上，选择组详细信息、编辑。
4. 对于 Desired capacity (所需容量)，将当前值增加 1。
5. 选择更新。启动实例时，上方窗格中的状态列会显示正在更新容量状态。

在增加所需容量后，您可以通过扩缩活动的描述来验证您的实例是否已成功启动且未终止。

查看扩展活动

1. 返回到 Auto Scaling 组页面并选择您的组。
2. 在活动选项卡上的活动历史记录下，状态列显示您的 Auto Scaling 组是否已成功启动实例。
3. 如果用户数据脚本失败，则在超时时段过后，您会看到状态为 `Canceled` 且状态消息为 `Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result` 的扩缩活动。

步骤 6：清除

如果您已完成为本教程创建的资源，请使用以下步骤将其删除。

删除生命周期钩子

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。
3. 在实例管理选项卡的生命周期钩子中，选择生命周期钩子 (TestAutoScalingEvent-hook)。
4. 依次选择操作、删除。
5. 再次选择删除以确认。

要删除启动模板

1. 打开 Amazon EC2 控制台的启动模板页面。
2. 选择启动模板 (TestAutoScalingEvent-template)，然后依次选择 Actions（操作）、Delete template（删除模板）。
3. 当系统提示进行确认时，键入 **Delete** 以确认删除指定启动模板，然后选择 Delete（删除）。

如果您已完成使用此示例 Auto Scaling 组，请将其删除。您还可以删除您创建的 IAM 角色和权限策略。

要删除 Auto Scaling 组

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中 Auto Scaling 组 (TestAutoScalingEvent-group) 旁边的复选框并选择 Delete（删除）。
3. 当系统提示进行确认时，键入 **delete** 以确认删除指定自动扩缩组，然后选择 Delete（删除）。

   Name（名称）列中的加载图标指示 Auto Scaling 组正在被删除。终止实例并删除组需要几分钟时间。

要删除 IAM 角色

1. 打开 IAM 控制台的 Roles page（角色页面）。
2. 选择函数的角色 (TestAutoScalingEvent-role)。

3. 选择删除。
4. 在系统提示进行确认时，键入角色的名称，然后选择 Delete（删除）。

删除 IAM policy

1. 打开 IAM 控制台的 Policies（策略）页面。
2. 选择您创建的策略 (`TestAutoScalingEvent-policy`)。
3. 依次选择操作、删除。
4. 在系统提示进行确认时，键入策略的名称，然后选择 Delete（删除）。

相关资源

以下相关主题可能有助于您开发基于实例元数据中可用数据而调用实例操作的代码。

- 通过实例元数据检索目标生命周期状态. 本节介绍其他使用案例的生命周期状态，例如实例终止。
- 添加生命周期钩子（控制台）. 此过程向您演示如何为横向扩展（实例启动）和横向缩减（实例终止或返回到暖池）添加生命周期挂钩。
- Amazon EC2 用户指南中的@@ 实例元数据类别。本主题列出了可用于调用实例操作的所有类别的EC2实例元数据。

有关向您展示如何使用 Amazon EventBridge 创建规则，根据您的 Auto Scaling 组中实例发生的事件调用 Lambda 函数的教程，请参阅。教程：配置调用 Lambda 函数的生命周期钩子

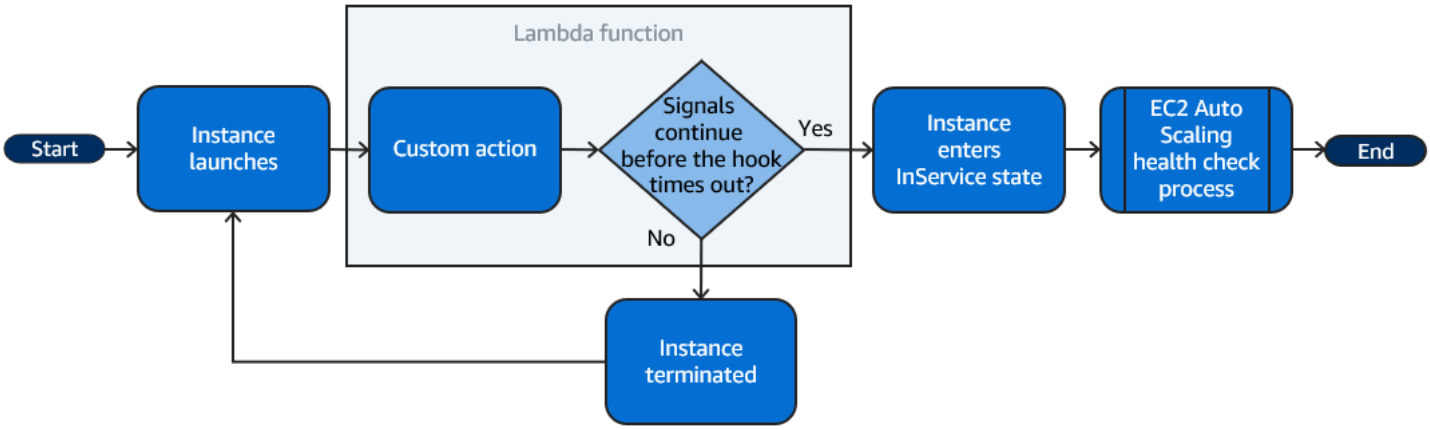
教程：配置调用 Lambda 函数的生命周期钩子

在本练习中，您将创建一条包含筛选模式的 Amazon EventBridge 规则，匹配该模式后，将调用一个 Amazon Lambda 函数作为规则目标。我们提供了要使用的筛选条件模式和示例函数代码。

如果一切配置正确，则在本教程结束时，Lambda 函数将在实例启动时执行自定义操作。自定义操作只是在与 Lambda 函数关联的 CloudWatch 日志日志流中记录事件。

Lambda 函数还执行回调，以便在此操作成功时继续执行实例的生命周期，但允许实例放弃启动并在操作失败时终止。

下图总结了使用 Lambda 函数执行自定义操作时横向扩展事件的流程。实例启动后，实例的生命周期将暂停，直到生命周期挂钩完成，要么是超时，要么是 Amazon A EC2 uto Scaling 收到继续运行的信号。



内容

前提条件

在开始本教程前，请创建 Auto Scaling 组（如果您还没有 Auto Scaling 组）。要创建 Auto Scaling 群组，请打开亚马逊 EC2控制台的 A uto Scaling 群组页面，然后选择创建 Auto Scaling 群组。

步骤 1：创建具有完成生命周期操作权限的 IAM 角色

在创建 Lambda 函数之前，必须首先创建执行角色和权限策略，以允许 Lambda 完成生命周期钩子。

创建策略

1. 打开 IAM 控制台的策略页面，然后选择创建策略。
2. 请选择 JSON 选项卡。
3. 在 “政策文档” 框中，将以下策略文档粘贴到框中，将中的*italics*文本替换为您的账号和 Auto Scaling 组的名称。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-
asg"
    }
  ]
}
```

4. 选择下一步。
5. 对于 Policy name，输入 **LogAutoScalingEvent-policy**。选择创建策略。

完成创建策略之后，您可以创建一个使用该策略的角色。

创建角色

1. 在左侧的导航窗格中，选择角色。
2. 选择 Create role（创建角色）。
3. 对于选择可信实体，选择 Amazon 服务。
4. 对于您的使用案例，选择 Lambda，然后选择 Next（下一步）。
5. 在 “添加权限” 下，选择您创建的策略（LogAutoScalingEvent-policy）和名为AWSLambdaBasicExecutionRole的策略。然后选择下一步。

> **Note**
>
> 该AWSLambdaBasicExecutionRole策略具有该函数将日志写入 CloudWatch 日志所需的权限。

6. 在 Name, review, and create（命名、检查并创建）页面上，对于 Role name（角色名称），输入 **LogAutoScalingEvent-role**，然后选择 Create role（创建角色）。

第 2 步：创建 Lambda 函数

创建 Lambda 函数以用作事件的目标。用 Node.js 编写的示例 Lambda 函数是在亚马逊 Auto EC2 Scaling 发出匹配事件 EventBridge 时调用的。

创建 Lambda 函数

1. 打开 Lambda 控制台的 Functions（函数）页面。
2. 选择 Create function（创建函数），然后选择 Author from scratch（从头开始创作）。
3. 在基本信息 下，对于函数名称，输入 **LogAutoScalingEvent**。
4. 对于运行时系统，选择 Node.js 18.x。
5. 向下滚动并选择更改默认执行角色，然后对于执行角色，选择使用现有角色。
6. 对于现有角色，选择 LogAutoScalingEvent-role。
7. 保留其他默认值。
8. 选择 Create function (创建函数)。您将返回到函数的代码和配置。
9. 当 LogAutoScalingEvent 函数在控制台中保持打开状态时，在编辑器中的 代码资源下，将以下示例代码复制到名为 index.mjs 的文件中。

```
import { AutoScalingClient, CompleteLifecycleActionCommand } from "@aws-sdk/client-
auto-scaling";
export const handler = async(event) => {
  console.log('LogAutoScalingEvent');
  console.log('Received event:', JSON.stringify(event, null, 2));
  var autoscaling = new AutoScalingClient({ region: event.region });
  var eventDetail = event.detail;
  var params = {
    AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
    LifecycleActionResult: 'CONTINUE', /* required */
    LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
    InstanceId: eventDetail['EC2InstanceId'],
    LifecycleActionToken: eventDetail['LifecycleActionToken']
  };
  var response;
  const command = new CompleteLifecycleActionCommand(params);
  try {
    var data = await autoscaling.send(command);
    console.log(data); // successful response
    response = {
      statusCode: 200,
```

```
      body: JSON.stringify('SUCCESS'),
    };
  } catch (err) {
    console.log(err, err.stack); // an error occurred
    response = {
      statusCode: 500,
      body: JSON.stringify('ERROR'),
    };
  }
  return response;
};
```

此代码仅记录事件，以便在本教程结束时，您可以看到与此 Lambda 函数关联的事件出现在 CloudWatch 日志日志流中。

10. 选择部署。

步骤 3：创建 EventBridge 规则

创建一条 EventBridge 规则来运行您的 Lambda 函数。有关使用 EventBridge 的更多信息，请参阅用于处理 EventBridge Auto Scaling 事件。

使用控制台创建规则

1. 打开 EventBridge 管理控制台。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 对于定义规则详细信息，请执行以下操作：
   a. 对于名称，请输入 **LogAutoScalingEvent-rule**。
   b. 对于事件总线，选择默认。当您的账户 Amazon Web Services 服务 中的某项生成事件时，它始终会转到您账户的默认事件总线。
   c. 对于规则类型，选择具有事件模式的规则。
   d. 选择下一步。
5. 对于 Build event pattern（构建事件模式），执行以下操作：
   a. 对于事件来源，选择Amazon 事件或 EventBridge 合作伙伴事件。
   b. 向下滚动到 事件模式，然后执行以下操作：

c. i. 对于事件源，选择 Amazon Web Services 服务。

ii. 对于 Amazon Web Services 服务，选择 Auto Scaling。

iii. 对于 Event Type (事件类型)，选择 Instance Launch and Terminate (实例启动和终止)。

iv. 默认情况下，该规则会与任何横向缩减或横向扩展事件匹配。要创建一条规则，以便在出现扩展事件并且实例因生命周期挂钩而进入等待状态时通知您，请选择特定实例事件，然后选择实例EC2启动生命周期操作。

v. 默认情况下，该规则与区域中任何 Auto Scaling 组匹配。若要使该规则与特定自动扩缩组匹配，请选择 特定组名称并选择组。

vi. 选择下一步。

6. 对于 Select target(s)（选择目标），请执行以下操作：

a. 对于 Target types（目标类型），选择 Amazon Web Services 服务。

b. 对于 Select a target（选择目标），选择 Lambda function（Lambda 函数）。

c. 对于 “函数”，选择LogAutoScalingEvent。

d. 选择 Next（下一步）两次。

7. 请在 审核和创建页面上，选择 创建。

步骤 4：添加生命周期钩子

在本节中，您将添加一个生命周期钩子，以便 Lambda 在启动时在实例上运行函数。

添加生命周期挂钩

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。这时将在页面底部打开一个拆分窗格。
3. 在下方窗格中，在实例管理选项卡的生命周期钩子中，选择创建生命周期钩子。
4. 要为横向扩展（实例启动）定义生命周期挂钩，请执行以下操作：

a. 对于生命周期钩子名称，请输入 **LogAutoScalingEvent-hook**。

b. 对于生命周期转换，请选择实例启动。

c. 对于检测信号超时，输入 **300** 以获取等待 Lambda 函数回调的秒数。

d. 对于默认结果，请选择放弃。这意味着，如果钩子超时而未收到 Lambda 函数的回调，Auto Scaling 组将终止新实例。

e. （可选）将通知元数据留空。我们传递给的事件数据 EventBridge 包含调用 Lambda 函数所需的所有必要信息。

5. 选择创建。

步骤 5：测试和验证事件

要测试事件，请通过将 Auto Scaling 组的所需容量增加 1 来更新 Auto Scaling 组。您的 Lambda 函数在增加所需容量后几秒钟内调用。

增加您的 Auto Scaling 组的大小

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中 Auto Scaling 组旁边的复选框可在下方窗格中查看详细信息，并且仍然可以看到上方窗格的顶行。
3. 在下方窗格中的详细信息选项卡上，选择组详细信息、编辑。
4. 对于 Desired capacity (所需容量)，将当前值增加 1。
5. 选择更新。启动实例时，上方窗格中的状态列会显示正在更新容量状态。

增加所需的容量后，可验证是否已调用 Lambda 函数。

查看 Lambda 函数的输出

1. 打开 CloudWatch 控制台的日志组页面。
2. 选择您的 Lambda 函数 (`/aws/lambda/LogAutoScalingEvent`) 的日志组的名称。
3. 选择日志流的名称，以查看由生命周期操作的函数提供的数据。

接下来，您可以通过扩展活动的描述来验证您的实例是否已成功启动。

查看扩展活动

1. 返回到 Auto Scaling 组页面并选择您的组。
2. 在活动选项卡上的活动历史记录下，状态列显示您的 Auto Scaling 组是否已成功启动实例。
   - 如果操作成功，扩展活动的状态将为“成功”。

- 如果失败，在等待几分钟后，您将看到状态为“已取消”的扩展活动，并显示状态消息“实例无法完成用户的生命周期操作：使用令牌 e85eb647-4fe0-4909-b341-a6c42EXAMPLE 的生命周期操作已放弃：使用“放弃”结果完成的生命周期操作”。

减少您的 Auto Scaling 组的大小

如果您不需要您为此测试启动的其他实例，则可以打开详细信息选项卡并将所需容量减少 1。

步骤 6：清除

如果您已完成仅为本教程创建的资源，请使用以下步骤将其删除。

删除生命周期钩子

1. 打开亚马逊 EC2 控制台的 A uto Scaling 群组页面。
2. 选中您的自动扩缩组旁边的复选框。
3. 在实例管理选项卡的生命周期钩子中，选择生命周期钩子 (LogAutoScalingEvent-hook)。
4. 依次选择操作、删除。
5. 再次选择删除以确认。

删除 Amazon EventBridge 规则

1. 在 Amazon EventBridge 控制台中打开 “规则” 页面。
2. 在事件总线下，选择与规则 (Default) 关联的事件总线。
3. 然后选中您的规则 (LogAutoScalingEvent-rule) 旁边的复选框。
4. 选择删除。
5. 在系统提示进行确认时，键入规则的名称，然后选择 Delete（删除）。

如果您已完成使用此示例函数，请将其删除。您还可以删除存储函数日志的日志组以及您创建的执行角色和权限策略。

要删除 Lambda 函数

1. 打开 Lambda 控制台的 Functions（函数）页面。
2. 选择函数 (LogAutoScalingEvent)。
3. 依次选择操作、删除。

4. 当系统提示进行确认时，键入 **delete** 以确认删除指定函数，然后选择 Delete（删除）。

删除日志组

1. 打开 CloudWatch 控制台的日志组页面。
2. 选择函数的日志组 (/aws/lambda/LogAutoScalingEvent)。
3. 依次选择 Actions（操作）和 Delete log group(s)（删除日志组）。
4. 在 Delete log group(s)（删除日志组）对话框中，选择 Delete（删除）。

删除执行角色

1. 打开 IAM 控制台的 Roles page（角色页面）。
2. 选择函数的角色 (LogAutoScalingEvent-role)。
3. 选择删除。
4. 在系统提示进行确认时，键入角色的名称，然后选择 Delete（删除）。

删除 IAM policy

1. 打开 IAM 控制台的 Policies（策略）页面。
2. 选择您创建的策略 (LogAutoScalingEvent-policy)。
3. 依次选择操作、删除。
4. 在系统提示进行确认时，键入策略的名称，然后选择 Delete（删除）。

相关资源

当您根据发生在 Auto Scaling 组中的实例的事件创建 EventBridge 规则时，以下相关主题可能会有所帮助。

- 用于处理 EventBridge Auto Scaling 事件. 本节向您演示其他使用案例的事件示例，包括针对横向缩减的事件。
- 添加生命周期钩子（控制台）. 此过程向您演示如何为横向扩展（实例启动）和横向缩减（实例终止或返回到暖池）添加生命周期挂钩。

有关向您演示如何使用实例元数据服务 (IMDS) 从实例本身内部调用操作的教程，请参阅 教程：使用数据脚本和实例元数据检索生命周期状态。

使用暖池减少启动时间较长的应用程序的延迟

暖池使您能够减少引导时间非常长的应用程序的延迟，例如，因为实例需要将大量数据写入磁盘。使用暖池，您不再需要过度配置 Auto Scaling 组来管理延迟以提高应用程序性能。有关更多信息，请参阅以下博客文章使用 A EC2 uto Scaling Warm Pools 更快地扩展应用程序。

> **Important**
>
> 在不需要的情况下创建暖池可能会导致不必要的成本。如果您的首次启动时间不会给您的应用程序带来明显的延迟问题，则您可能不需要使用暖池。

主题

核心概念

在您开始之前，请熟悉以下核心概念：

暖池

温池是预先初始化的 EC2 实例池，位于 Auto Scaling 组旁边。无论何时您的应用程序需要向外扩展，自动扩缩组都可以在暖池上绘制，以满足其新的所需容量。它可帮助您确保实例准备好快速开

启为应用程序提供流量，从而加快对横向扩展事件的响应。当实例离开暖池时，它们将计入该组的所需容量。这称为热启动。

当实例处于预热池中时，仅当处于 InService 状态的实例的指标值大于扩缩策略的告警阈值上限（与目标跟踪扩缩策略的目标利用率相同）时，扩缩策略才会横向扩展。

暖池大小

默认情况下，暖池的大小以 Auto Scaling 组的最大容量与其所需容量之差来计算。例如，如果 Auto Scaling 组的所需容量为 6 且最大容量为 10，则当您首次设置暖池并且该池正在初始化时，暖池的大小将为 4。

要单独指定暖池的最大容量，请使用自定义规格（MaxGroupPreparedCapacity）选项，并为暖池设置一个大于该组当前容量的自定义值。如果您提供自定义值，则暖池的大小将计算为自定义值与组当前所需容量之间的差值。例如，如果自动扩缩组的所需容量为 6，最大容量为 20 且自定义值为 8，则当您首次设置暖池并且该池初始化时，暖池的大小将为 2。

仅当使用大自动扩缩组时，才可能需要使用自定义规格（MaxGroupPreparedCapacity）选项来管理拥有暖池的成本效益。例如，与在暖池中持续预留 500 个实例以备未来使用的方案相比，拥有 1,000 个实例的自动扩缩组、最大容量为 1,500（用于为紧急流量峰值提供额外容量）和拥有 100 个实例的暖池方案可能有利于更好地实现目标。

最小暖池大小

考虑使用最小大小设置 (MinSize) 来静态设置要在温池中维护的最小实例数。默认情况下，没有设置最小大小。当您指定MaxGroupPreparedCapacity要确保即使 Auto Scaling 组的所需容量高于 Auto Scaling 组的所需容量时，也要确保在温池中保留最少数量的实例，则该MinSize设置非常有用MaxGroupPreparedCapacity。

暖池实例状态

您可以将暖池中的实例保持在以下三种状态之一：Stopped、Running 或 Hibernated。将实例保持在 Stopped 状态是一个有效的方法，以最大限度地降低成本。对于已停止的实例，您只需为您使用的卷和已附加到实例的弹性 IP 地址付费。

或者，您也可以将实例保持在 Hibernated 状态，以停止实例而不删除其内存内容 (RAM)。当某个实例处于休眠状态时，这表示操作系统会将 RAM 的内容保存到 Amazon EBS 根卷中。当再次开启此实例时，会将此根卷还原到其之前的状态，还将重新加载 RAM 内容。当实例处于休眠状态时，您只需为 EBS 卷（包括 RAM 内容的存储空间，以及已附加到实例的弹性 IP 地址）付费。

此外还可以将暖池中的实例保持在 Running 状态，但我们强烈建议不要这样操作，以避免产生不必要的费用。当实例停止或休眠时，您可以节省实例本身的成本。您仅在实例正在运行时为它们付费。

生命周期钩子

您使用生命周期挂钩来将实例置于等待状态，以便您可以对实例执行自定义操作。自定义操作将在实例启动时或其终止之前执行。

在暖池配置中，生命周期挂钩可以延迟实例停止或休眠以及延迟在横向扩展事件期间投入使用，直到它们完成初始化。如果您在没有生命周期钩子的情况下向 Auto Scaling 组添加暖池，则需要很长时间才能完成初始化的实例可能会停止或休眠，然后在准备就绪之前在横向扩展事件期间投入使用。

实例再使用策略

默认情况下，Amazon A EC2 uto Scaling 会在您的 Auto Scaling 组缩容时终止您的实例。然后，它会将新实例启动到暖池中，以替换已终止的实例。

而如果您希望将实例返回到暖池，则可以指定实例再使用策略。这使您可以再使用这些已配置为给应用程序提供流量的实例。为了确保您的温池不会过度配置，Amazon A EC2 uto Scaling 可以根据其设置终止温池中的实例以缩小其大小。当终止暖池中的实例时，它会使用默认终止策略来选择首先终止哪些实例。

> **Important**
>
> 如果您希望在横向缩减时休眠实例，并且 Auto Scaling 组中有现有实例，则它们必须满足实例休眠的要求。如果不这样做，当实例返回到暖池时，它们将回退到停止状态，而不是休眠状态。

> **Note**
>
> 目前，您只能通过使用 Amazon CLI 或 SDK 来指定实例再使用策略。此功能在控制台中不可用。

先决条件

在为自动扩缩组创建暖池之前，请决定如何使用生命周期挂钩将新实例初始化为适当的初始状态。

要在实例因生命周期挂钩而处于等待状态时对其执行自定义操作，您有两种选择：

- 对于您希望在启动时对实例运行命令的简单场景，可以在为 Auto Scaling 组创建启动模板或启动配置时包含用户数据脚本。用户数据脚本只是普通的 shell 脚本或 cloud-init 由运行的指令 cloud-init 您

的实例何时启动。该脚本还可通过使用实例（该脚本运行所在的实例）的 ID 来控制您的实例何时转换到下一个状态。如果您尚未这样做，请更新脚本，从实例元数据中检索实例的实例 ID。有关更多信息，请参阅 Amazon EC2 用户指南中的访问实例元数据。

> **Tip**
>
> 要在实例重新开启时运行用户数据脚本，用户数据必须采用 MIME 多段格式，并在用户数据的 `#cloud-config` 部分中指定以下内容：
>
> ```
> #cloud-config
> cloud_final_modules:
>  - [scripts-user, always]
> ```

- 对于需要服务的高级场景，例如 Amazon Lambda 在实例进入或离开温池时执行某项操作，您可以为 Auto Scaling 组创建生命周期挂钩，并将目标服务配置为根据生命周期通知执行自定义操作。有关更多信息，请参阅 支持的通知目标。

使实例为休眠做好准备

要让 Auto Scaling 实例做好使用*Hibernated*池状态的准备，请按照 Amazon 用户指南中的休眠先决条件主题中所述，创建一个已正确设置为支持实例休眠的新启动模板或启动配置。 EC2 然后，将新启动模板或启动配置与 Auto Scaling 组关联起来，并开启实例刷新，以替换与之前的启动模板或启动配置关联的实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。

更新暖池中的实例

如需更新暖池中的实例，您可以创建新的启动模板或启动配置，并将其与自动扩缩组关联起来。所有新实例都将使用新 AMI 和在启动模板或启动配置中指定的其他更新启动，但现有实例不受影响。

要强制启动使用新启动模板或启动配置的替换暖池实例，您可以启动实例刷新，对您的组进行滚动更新。实例刷新首先将替换 `InService` 实例。然后，它将替换暖池中的实例。有关更多信息，请参阅使用实例刷新更新自动扩缩组中的实例。

相关资源

您可以访问我们的GitHub存储库，查看温池生命周期挂钩的示例。

限制

- 您无法将暖池添加到具有混合实例策略的自动扩缩组。您也无法向具有请求竞价型实例的启动模板或启动配置的 Auto Scaling 组或具有指定 Systems Manager 参数的启动模板的 Auto Scaling 组添加温池。
- 只有当实例将 Amazon EBS 卷作为其根设备时，Amazon A EC2 uto Scaling 才能将其置于`Stopped`或`Hibernated`状态。无法停止或休眠对根设备使用实例存储的实例。
- 只有满足亚马逊 EC2 用户指南中休眠先决条件主题中列出的所有要求时，Ama zon A EC2 uto Scaling 才能将实例置于`Hibernated`状态。
- 如果您的暖池在发生向外扩展事件时耗尽，则实例将直接启动到 Auto Scaling 组（冷启动)。如果可用区容量不足，您也可能会遇到冷启动。
- 如果暖池中的实例在启动过程中遇到问题，导致其无法进入 `InService` 状态，则该实例将被视为启动失败并已终止。无论根本原因是什么（例如容量不足错误或任何其他因素），此情况都适用。
- 如果您尝试将暖池与 Amazon Elastic Kubernetes Service (Amazon EKS) 托管节点组一起使用，则仍在初始化的实例可能会注册到您的 Amazon EKS 集群。因此，在准备停止或休眠时，集群可能会在实例上安排作业。
- 同样，如果您尝试将暖池与 Amazon ECS 集群一起使用，则实例可能会在完成初始化之前注册到集群。要解决此问题，您必须配置启动模板或启动配置，使其包含用户数据中的特殊代理配置变量。有关更多信息，请参阅 《Amazon Elastic Container Service 开发人员指南》中的为自动扩缩组使用暖池。

在自动扩缩组中将生命周期挂钩与暖池一起使用

暖池中的实例将保持自己独立的生命周期，以帮助您为每次转换创建适当的自定义操作。此生命周期旨在帮助您在实例仍在初始化之时以及将其投入使用之前，调用目标服务（例如，Lambda 函数）中的操作。

> **Note**
>
> 不会更改用于添加和管理生命周期钩子以及完成生命周期操作的 API 操作。只会更改实例生命周期。

有关添加生命周期钩子的更多信息，请参阅 向自动扩缩组添加生命周期挂钩。有关完成生命周期操作的更多信息，请参阅 在自动扩缩组中完成生命周期操作。

对于进入暖池的实例，出于以下原因之一，您可能需要生命周期钩子：

- 您想从需要很长时间才能完成初始化的 AMI 启动 EC2 实例。
- 你想运行用户数据脚本来引导 EC2 实例。

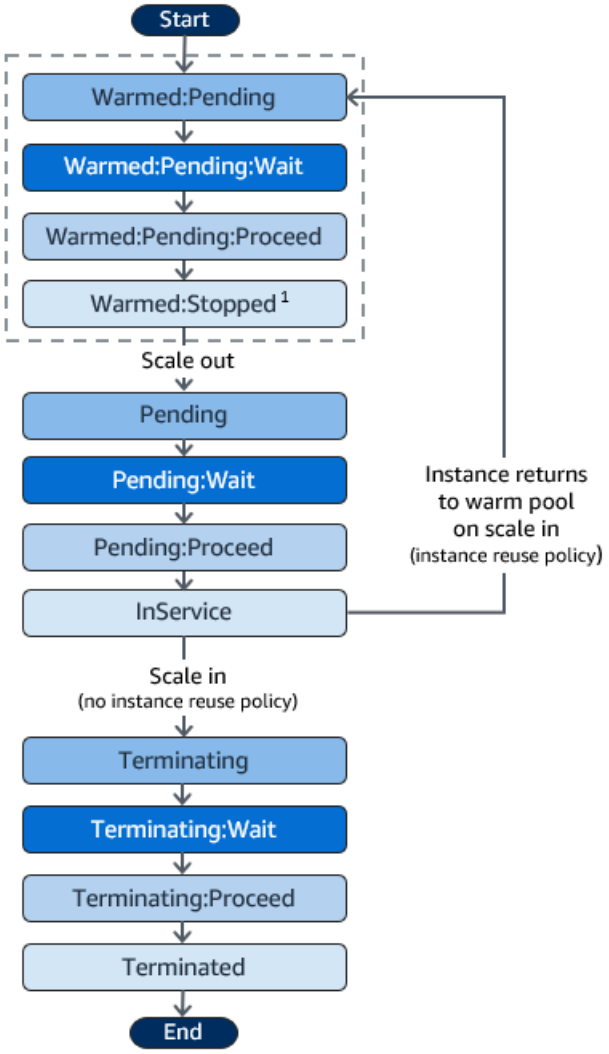
对于离开暖池的实例，出于以下原因之一，您可能需要生命周期钩子：

- 您可以额外花一些时间来准备要使用的 EC2 实例。例如，您可能有这样的服务，它们必须在实例重新开启时开启，然后您的应用程序才能正常工作。
- 您希望预填充缓存数据，以使新服务器不会使用空缓存启动。
- 您希望使用配置管理服务将新实例注册为托管实例。

暖池中实例的生命周期状态转换

作为其生命周期的组成部分，Auto Scaling 实例可在多种状态之间转换。

下图显示了在您使用暖池时 Auto Scaling 状态之间的转换：



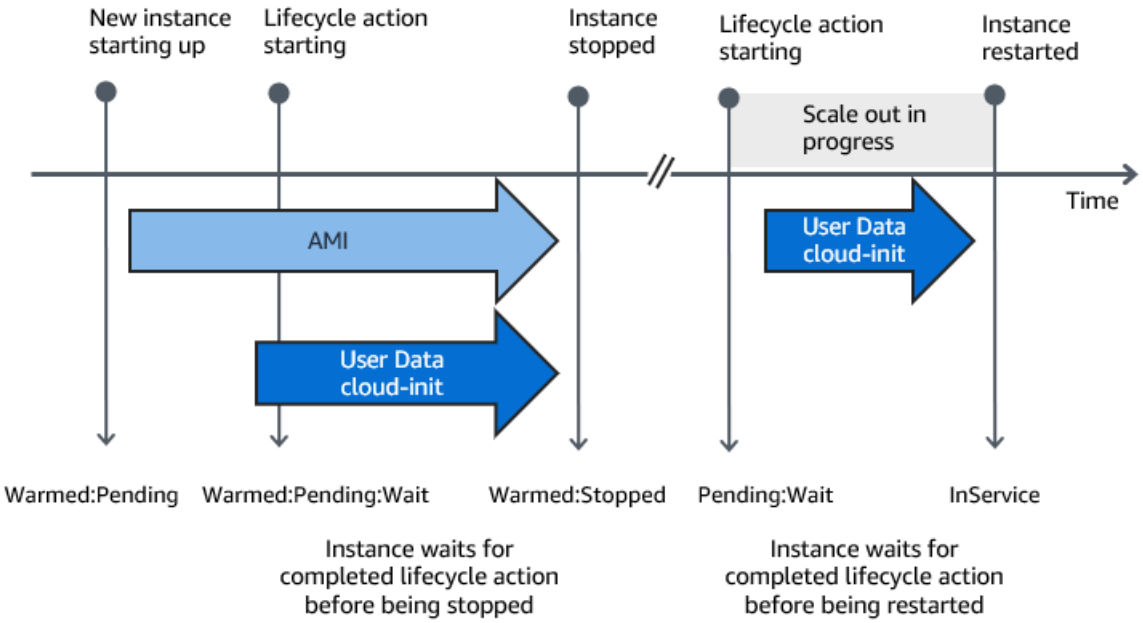
[1] 此状态因暖池的池状态设置而异。如果将池状态设置为 Running，则此状态为 Warmed:Running。如果将池状态设置为 Hibernated，则此状态为 Warmed:Hibernated。

添加生命周期钩子时，请考虑以下事项：

- 为 autoscaling:EC2_INSTANCE_LAUNCHING 生命周期操作配置生命周期挂钩时，新启动的实例在到达 Warmed:Pending:Wait 状态时会先暂停以执行自定义操作，然后在实例重启并到达 Pending:Wait 状态时再次重复上述操作。
- 为 EC2_INSTANCE_TERMINATING 生命周期操作配置生命周期挂钩，终止的实例在到达 Terminating:Wait 状态时会暂停以执行自定义操作。但是，如果您指定了实例重用策略来将实例大规模返回到暖池，而不是终止它们，则对于 EC2_INSTANCE_TERMINATING 生命周期操作，返回暖池的实例将在 Warmed:Pending:Wait 状态暂停以执行自定义操作。

- 如果对应用程序的需求耗尽了温池，Amazon A EC2 uto Scaling 可以直接在 Auto Scaling 组中启动实例，前提是该组尚未达到最大容量。如果实例直接启动到该组中，则这些实例仅在 `Pending:Wait` 状态中暂停以执行自定义奥做。
- 要控制实例在转换到下一个状态之前保持等待状态的时间，请将您的自定义操作配置为使用 complete-lifecycle-action 命令。使用生命周期挂钩，实例会一直处于等待状态，直到您通知 Amazon A EC2 uto Scaling 指定的生命周期操作已完成，或者直到超时时间结束（默认为一小时）。

以下总结了横向扩展事件的流程。



当实例达到等待状态时，Amazon A EC2 uto Scaling 会发送通知。本指南的 EventBridge 部分提供了这些通知的示例。有关更多信息，请参阅 暖池示例事件类型和模式。

支持的通知目标

Amazon A EC2 uto Scaling 支持将以下任何一项定义为生命周期通知的通知目标：

- EventBridge 规则
- Amazon SNS 主题
- Amazon SQS 队列

> **Important**
>
> 请记住，如果您的启动模板或启动配置中有在实例启动时配置实例的用户数据 (cloud-init) 脚本，则无需接收通知即可对正在启动或重启的实例执行自定义操作。

以下各节包含多个链接，它们指向描述如何配置通知目标的文档：

EventBridge 规则：要在 Amazon A EC2 uto Scaling 将实例置于等待状态时运行代码，您可以创建 EventBridge 规则并指定一个 Lambda 函数作为其目标。要根据不同的生命周期通知调用不同的 Lambda 函数，您可以创建多条规则，并将每条规则与特定的事件模式和 Lambda 函数关联起来。有关更多信息，请参阅 为温水池活动创建 EventBridge 规则。

Amazon SNS 主题：要在实例处于等待状态时接收通知，您可以创建 Amazon SNS 主题，然后设置 Amazon SNS 消息筛选，以根据消息属性以不同的方式传送生命周期通知。有关更多信息，请参阅 使用 Amazon SNS 接收通知。

Amazon SQS 队列：要为生命周期通知设置传送点，使相关使用者可以接收并处理它们，您可以创建 Amazon SQS 队列和处理来自 SQS 队列的消息的队列使用者。如果您希望队列使用者根据消息属性以不同方式处理生命周期通知，则还必须设置队列使用者以解析消息，然后在特定属性与所需值匹配时对消息执行操作。有关更多信息，请参阅 使用 Amazon SQS 接收通知。

为自动扩缩组创建一个暖池

本主题介绍如何为自动扩缩组创建暖池。

> **Important**
>
> 在您继续操作之前，请先完成创建暖池的先决条件，并确认您已为自动扩缩组创建了生命周期挂钩。

创建暖池

使用以下步骤为您的自动扩缩组创建暖池。

创建暖池（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。

2. 选中现有组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。

3. 选择实例管理选项卡。
4. 在暖池下，选择创建暖池。
5. 要配置暖池，请执行以下操作：

   a. 对于暖池实例状态，选择要在实例进入暖池时将其转换为哪个状态。默认为 Stopped。

   b. 对于最小暖池大小，输入要在暖池中保留的最少实例数。

   c. 对于实例重用，请选中在横向缩减上重复使用复选框，以允许自动扩缩组中的实例在横向缩减时可以退回暖池。

   d. 对于暖池大小，请选择可用选项之一：

      - 默认规格：暖池的大小由自动扩缩组的最大容量与所需容量之差来确定。此选项简化了暖池管理。创建暖池后，只需调整组的最大容量即可轻松更新其大小。
      - 自定义规格：暖池的大小由自定义值与自动扩缩组所需容量之差来确定。此选项使您可以灵活地独立于组的最大容量来管理暖池的大小。

6. 查看基于当前设置估计暖池大小部分，以确认默认或自定义规格如何应用于暖池大小。请记住，暖池的大小取决于自动扩缩组的所需容量；如果该组发生扩缩，则该容量将发生变化。
7. 选择创建。

删除暖池

当您不再需要暖池时，您可以使用以下步骤将其删除。

要删除暖池（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。

3. 选择实例管理选项卡。
4. 对于 Warm pool（暖池），选择 Actions（操作）、Delete（删除）。
5. 当系统提示进行确认时，选择 Delete（删除）。

查看运行状况检查状态以及运行状况检查失败的原因

运行状况检查允许 Amazon A EC2 uto Scaling 确定实例何时运行状况不佳并应终止。对于保持在 `Stopped` 状态的暖池实例，它采用了 Amazon EBS 拥有的 `Stopped` 实例的可用性知识来识别运行状况不佳的实例。它通过调用 `DescribeVolumeStatus` API 以确定附加到实例的 EBS 卷的状态来执行此操作。对于保持状态的温池实例，它依靠 EC2 状态检查来确定实例的运行状况。`Running`虽然温池实例没有运行状况检查宽限期，但是在生命周期挂钩完成之前，Amazon A EC2 uto Scaling 不会开始检查实例的运行状况。

当发现某个实例运行状况不佳时，Amazon A EC2 uto Scaling 会自动删除运行状况不佳的实例，并创建一个新的实例来替换它。实例通常在运行状况检查失败后几分钟内终止。有关更多信息，请参阅 查看运行状况检查失败原因。

还支持自定义运行状况检查。如果您拥有自己的运行状况检查系统，可以检测实例的运行状况并将此信息发送到 Amazon A EC2 uto Scaling，这可能会很有帮助。有关更多信息，请参阅 为您的自动扩缩组设置自定义运行状况检查。

在 Amazon A EC2 uto Scaling 控制台上，您可以查看您的温池实例的状态（健康或不健康）。您也可以使用 Amazon CLI 或其中一个来查看他们的健康状态 SDKs。

查看暖池实例的状态（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在实例管理选项卡的暖池实例下，生命周期列显示实例的状态。

   运行状况列显示了 Amazon A EC2 uto Scaling 对实例运行状况所做的评估。

   > **Note**
   >
   > 新实例开始运行状况良好。在生命周期钩子完成之前，不会检查实例的运行状况。

查看运行状况检查失败原因（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。

2. 选中 Auto Scaling 组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。

3. 在活动选项卡的活动历史记录下，状态列显示您的 Auto Scaling 组是否已成功启动或终止实例。

   如果它终止了任何运行状况不佳的实例，原因列显示终止的日期和时间以及运行状况检查失败的原因。例如，“在 2021-04-01T21:48:35Z 时，实例因 EBS 卷运行状况检查失败而停止服务”。

查看暖池实例的状态 (Amazon CLI)

使用以下describe-warm-pool命令查看 Auto Scaling 组的温池。

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

输出示例。

```
{
    "WarmPoolConfiguration": {
        "MinSize": 0,
        "PoolState": "Stopped"
    },
    "Instances": [
        {
            "InstanceId": "i-0b5e5e7521cfaa46c",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        },
        {
            "InstanceId": "i-0e21af9dcfb7aa6bf",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
```

```
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        }
    ]
}
```

查看运行状况检查失败原因 (Amazon CLI)

使用以下 describe-scaling-activities 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下为示例响应，其中 Description 表示您的 Auto Scaling 组已终止实例，Cause 指示运行状况检查失败的原因。

扩展活动按开始时间排序。首先描述仍在进行的活动。

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to EBS volume health check failure.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

使用创建和管理温池的示例 Amazon CLI

您可以使用 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或创建和管理温池 SDKs。

以下示例向您演示如何使用 Amazon CLI创建和管理暖池。

内容

示例 1：将实例保持在 **Stopped** 状态

以下put-warm-pool示例创建了一个使实例保持Stopped状态的温池。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped
```

示例 2：将实例保持在 **Running** 状态

以下put-warm-pool示例创建了一个使实例保持Running状态而不是Stopped状态的温池。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Running
```

示例 3：将实例保持在 **Hibernated** 状态

以下put-warm-pool示例创建了一个使实例保持Hibernated状态而不是Stopped状态的温池。这使您可以停止实例，而无需删除其内存内容 (RAM)。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
```

```
  --pool-state Hibernated
```

示例 4：在横向缩减时将实例返回到暖池

以下put-warm-pool示例创建了一个使实例保持Stopped状态的温池，并包含--instance-reuse-policy选项。实例重用策略值'{"ReuseOnScaleIn": true}'告诉 Amazon A EC2 uto Scaling 在您的 Auto Scaling 组缩容时将实例返回到温池。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

示例 5：指定暖池中的最小实例数

以下put-warm-pool示例创建了一个至少维护 4 个实例的温池，因此至少有 4 个实例可用于处理流量高峰。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 4
```

示例 6：使用自定义规格定义暖池大小

默认情况下，Amazon A EC2 uto Scaling 按照 Auto Scaling 组的最大容量和所需容量之差来管理您的温池的大小。但是，您可以使用 --max-group-prepared-capacity 选项管理暖池的大小，使其独立于组的最大容量。

以下put-warm-pool示例创建了一个温池，并设置了在温池和 Auto Scaling 组中可以同时存在的最大实例数。如果组的所需容量为 800，则运行此命令后初始化时，暖池的初始大小为 100。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900
```

要在暖池中保留最少数量的实例，请使用命令包含 --min-size 选项，如下所示。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

示例 7：定义绝对暖池大小

如果您为 --max-group-prepared-capacity 和 --min-size 选项设置了相同的值，则暖池将具有绝对大小。以下put-warm-pool示例创建了一个保持 10 个实例的恒定温池大小的温池。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

示例 8：删除暖池

使用以下delete-warm-pool命令删除温池。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

如果温池中有实例，或者扩展活动正在进行中，请使用带--force-delete选项的delete-warm-pool命令。此选项还会终止 Amazon EC2 实例和任何未完成的生命周期操作。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Auto Scaling 组的区域偏移

区域转移是 Amazon 应用程序恢复控制器 (ARC) 中的一项功能。通过区域切换，您只需一个操作即可快速从可用区中的应用程序损坏中恢复。当您为 Auto Scaling 组启用区域偏移时，该组将在 ARC 区域偏移服务中注册。然后，您可以使用、或 API 开始区域移动 Amazon Web Services Management Console Amazon CLI，Auto Scaling 组会将处于活动区域偏移状态的可用区视为受损。

Auto Scaling 分组区域偏移概念

在继续操作之前，请确保您熟悉以下与与 ARC 区域偏移集成相关的核心概念。

ARC 区域偏移

启用 Auto Scaling 功能后，Auto Scaling 可以将 Auto Scaling 组注册为 ARC 区域偏移。注册后，您可以使用 ARC `ListManagedResources` API 查看您的资源。有关更多信息，请参阅《Amazon 应用程序恢复控制器 (ARC) 开发人员指南》中的 ARC 中的区域偏移。

可用区再平衡

Auto Scaling 会尝试保持每个可用区域的容量平衡。当可用区之间出现不平衡时，Auto Scaling 会自动尝试修复不平衡问题。有关更多信息，请参阅 实例分配。

动态扩展

动态扩展会根据您在扩展策略中选择的指标来扩展 Auto Scaling 组的所需容量。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。

运行状况检查

Auto Scaling 会定期检查 Auto Scaling 组内所有实例的运行状况，以确保它们正在运行且状态良好。当检测到运行状况不佳的实例时，Auto Scaling 会将其标记为替换。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

实例刷新

您可以使用实例刷新以更新自动扩缩组中的实例。实例刷新开始后，Auto Scaling 会尝试替换 Auto Scaling 组中的所有实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。

已预先缩放

您可以容忍单个可用区的损失，因为剩余的可用区中有足够的容量供应用程序使用。

横向扩展

当您增加 Auto Scaling 组的所需容量时，Auto Scaling 会尝试启动其他实例以满足新的所需容量。默认情况下，Auto Scaling 以平衡的方式启动实例，以在 Auto Scaling 组中每个已启用的可用区域中保持相等的容量。

Auto Scaling 群组的区域偏移是如何运作的

假设您有一个包含以下可用区域的 Auto Scaling 组：

- `us-east-1a`
- `us-east-1b`
- `us-east-1c`

您已在所有可用区域中启用区域移动，并注意到其中出现故障`us-east-1a`，因此可以触发区域偏移。当触发区域偏移时，会发生以下行为。`us-east-1a`

- 向@@ 外扩展 — Auto Scaling 将在运行状况良好的可用区（`us-east-1b`和`us-east-1c`）中启动所有新的容量请求。
- 动态扩展 — Auto Scaling 将阻止扩展策略减少所有可用区域中的所需容量。Auto Scaling 不会阻止扩展策略增加所有可用区域中的所需容量。
- 实例刷新 — Auto Scaling 将延长在区域偏移处于活动状态时延迟的任何实例刷新过程的超时时间。

下表描述了触发区域偏移时每个选项的运行状况检查行为。`us-east-1a`

可用区运行状况检查行为选择受损	Health Check 行为			
替换不健康的	所有可用区（us-east-1a 、us-east-1b 和us-east-1c ）中显示运行状况不佳的实例将被替换。			
忽略不健康	显示为运行状况不佳的实例将在us-east-1b 和us-east-1c 中替换。可用区中的实例不会被有效的区域移动 (us-east-1a) 所取代。			

使用区域偏移的最佳实践

为了在使用区域转移时保持应用程序的高可用性，我们建议采用以下最佳实践：

- 监控 EventBridge 通知以确定何时出现持续的可用区损坏事件。有关更多信息，请参阅 用于处理 EventBridge Auto Scaling 事件。
- 使用具有适当阈值的扩展策略，确保您有足够的容量来容忍可用区的损失。
- 将实例维护策略设置为最低健康百分比为 100。使用此设置，Auto Scaling 会等待新实例准备就绪，然后再终止运行状况不佳的实例。

对于预先缩放的客户，我们还建议采取以下措施：

- 选择 I gnore un healthy 作为受损可用区的运行状况检查行为，因为在受损事件期间，您无需更换运行状况不佳的实例。

- 在 ARC 中为你的 Auto Scaling 组使用分区自动移位。ARC 中的区域自动切换功能允许在 Amazon 检测 Amazon 到可用区存在障碍时将资源的流量从可用区转移出去。有关更多信息，请参阅《Amazon 应用程序恢复控制器 (ARC) 开发人员指南》中的 ARC 中的区域自动切换。

对于禁用跨区域负载均衡器的客户，我们还建议采取以下措施：

- 仅在可用区分配中使用平衡。
- 如果您在 Auto Scaling 组和负载均衡器上都使用区域偏移，请先取消您的 Auto Scaling 组的区域偏移。然后，等待所有可用区域的容量均衡，然后再取消负载均衡器上的区域切换。
- 由于启用区域转移并使用禁用跨区域的负载均衡器时可能会出现容量不平衡，因此 Auto Scaling 包含一个额外的验证步骤。如果您遵循最佳实践，则可以通过选中 Amazon Web Services Management Console 复选框或使用`CreateAutoScalingGroup`、`UpdateAutoScalingGroup`或中的`skip-zonal-shift-validation`标志来确认这种可能性`AttachTrafficSources`。

有关在 Auto Scaling 群组中使用区域偏移的更多信息，请参阅 Amazon 计算博客在 Amazon Aut EC2 o Scaling 中使用区域偏移。

使用 Amazon Web Services Management Console 或启用区域偏移 Amazon CLI

要启用区域偏移，请使用以下方法之一。

Console

在新群组上启用区域切换（控制台）

1. 按照中的使用启动模板创建 Auto Scaling 组说明完成过程中的每个步骤，直到步骤 10。
2. 在与其他服务集成页面上，对于应用程序恢复控制器 (ARC) 区域切换，选中复选框以启用区域移动。
3. 对于运行状况检查行为，请选择 “忽略不健康状况” 或 “替换不健康”。有关更多信息，请参阅 Auto Scaling 群组的区域偏移是如何运作的。
4. 继续完成使用启动模板创建 Auto Scaling 组中的步骤。

Amazon CLI

要在新群组上启用区域偏移 ()Amazon CLI

向 create-auto-scaling-group 命令添加 `--availability-zone-impairment-policy` 参数。

该`--availability-zone-impairment-policy`参数有两个选项：

- ZonalShiftEnabled— 如果设置为`true`，Auto Scaling 将使用 ARC 区域偏移注册 Auto Scaling 组，您可以在 ARC 控制台上启动、更新或取消区域偏移。如果设置为`false`，则 Auto Scaling 会从 ARC 区域偏移中取消注册 Auto Scaling 组。必须已启用区域偏移才能将其设置为。`false`
- ImpairedZoneHealthCheckBehavior— 如果设置为`replace-unhealthy`，则可用区中运行状况不佳的实例将替换为有效的区域切换。如果设置为`ignore-unhealthy`，则可用区中运行状况不佳的实例不会被活跃的区域转移所取代。有关更多信息，请参阅 Auto Scaling 群组的区域偏移是如何运作的。

以下示例对名*my-asg*为的新 Auto Scaling 组启用区域偏移。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --availability-zones us-east-1a us-east-1b us-east-1c \
  --availability-zone-impairment-policy '{
      "ZonalShiftEnabled": true,
      "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
    }'
```

Console

在现有群组上启用区域切换（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在集成选项卡的应用程序恢复控制器 (ARC) 区域偏移下，选择编辑。

5. 选中该复选框以启用区域移动。
6. 对于运行状况检查行为，请选择 “忽略不健康状况” 或 “替换不健康”。有关更多信息，请参阅 Auto Scaling 群组的区域偏移是如何运作的。
7. 选择更新。

Amazon CLI

要在现有群组上启用区域偏移 ()Amazon CLI

向 update-auto-scaling-group 命令添加 `--availability-zone-impairment-policy` 参数。

该`--availability-zone-impairment-policy`参数有两个选项：

- ZonalShiftEnabled— 如果设置为`true`，Auto Scaling 将使用 ARC 区域偏移注册 Auto Scaling 组，您可以在 ARC 控制台上启动、更新或取消区域偏移。如果设置为`false`，则 Auto Scaling 会从 ARC 区域偏移中取消注册 Auto Scaling 组。必须已启用区域偏移才能将其设置为。`false`
- ImpairedZoneHealthCheckBehavior— 如果设置为`replace-unhealthy`，则可用区中运行状况不佳的实例将替换为有效的区域切换。如果设置为`ignore-unhealthy`，则可用区中运行状况不佳的实例不会被活跃的区域转移所取代。有关更多信息，请参阅 Auto Scaling 群组的区域偏移是如何运作的。

以下示例在指定的 Auto Scaling 组上启用区域偏移。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --availability-zone-impairment-policy '{
      "ZonalShiftEnabled": true,
      "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
    }'
```

Auto Scaling 组可用区域分布

以下信息描述了 Auto Scaling 组可用区域策略。

平衡尽力而为

Auto Scaling 在已启用的可用区域中保持相同数量的实例。如果某个可用区的启动尝试失败，Auto Scaling 会尝试在另一个运行状况良好的可用区中启动实例。对于需要可用区冗余且不受组不平衡影响的应用程序来说，此策略非常重要。

仅限平衡

Auto Scaling 在已启用的可用区域中保持相同数量的实例。如果在可用区启动尝试失败，Auto Scaling 将继续尝试在可用区启动实例。此策略对于满足某些要求（例如基于法定人数的工作负载）非常重要，或者您的 Auto Scaling 组可以容忍可用区的损失，因为您在剩余的可用区中有足够的容量。

可用区分发策略选择位于的 “网络” 部分中， Amazon Web Services Management Console 或者您可以使用create-auto-scaling-group或update-auto-scaling-group命令。

有关更多信息，请参阅 使用启动模板创建自动扩缩组。

从自动扩缩组中分离或附加实例

您可以从自动扩缩组中分离实例。分离实例后，该实例将变为独立，可以自行管理，也可以附加到与其所属原始组分开的其他自动扩缩组。例如，当您想要使用已经运行应用程序的现有实例执行测试时，这可能很有用。

本主题提供有关如何分离和附加实例的说明。附加实例时，您也可以使用现有实例而不是已分离的实例。

建议使用备用程序将实例暂时从组中删除，而不是将实例分离然后重新附加到同一个组。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

内容

分离实例的注意事项

分离实例时，请记住以下几点：

- 仅当实例处于 `InService` 状态时，您才能将其分离。
- 分离实例后，该实例会继续运行并产生费用。为避免不必要的费用，请确保在不再需要时重新附加或终止已分离的实例。

- 您可以选择按照要分离的实例数量递减所需容量。如果您选择不减少容量，Amazon A EC2 uto Scaling 会启动新实例来替换已分离的实例以保持所需的容量。
- 如果要分离的实例数将使自动扩缩组容量降到低于其最小容量，则必须递减最小容量。
- 如果在未递减所需容量的情况下从同一可用区分离多个实例，除非暂停 AZRebalance 进程，否则组将自行重新平衡。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 如果您将实例从已附加负载均衡器目标组或经典负载均衡器的 Auto Scaling 组分离，则将从该负载均衡器取消注册实例。如果您的负载均衡器启用了连接耗尽（取消注册延迟），Amazon A EC2 uto Scaling 会等待正在进行的请求完成。

> **Note**
>
> 如果您要分离的实例位于 Standby 状态，请谨慎行事。在将实例置于 Standby 状态后尝试分离实例可能会导致其他实例意外终止。

附加实例的注意事项

附加实例时应注意以下几点：

- Amazon A EC2 uto Scaling 对待附加实例的处理方式与群组本身启动的实例相同。这意味着，如果选择了附加的实例，则可以在横向缩减事件期间将其终止。授予的权限 AWSServiceRoleForAutoScaling 服务相关角色允许 Amazon A EC2 uto Scaling 执行此操作。
- 当您附加实例时，该组的所需容量将增加要附加的实例数。如果添加新实例后的所需容量超出了组的最大大小，则附加更多实例的请求会失败。
- 如果您向组中添加实例导致可用区分布不均衡，Amazon A EC2 uto Scaling 会重新平衡该组以重新建立均衡分配，除非您暂停该AZRebalance流程。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 如果您将实例附加到已附加负载均衡器目标组或经典负载均衡器的 Auto Scaling 组，则会将实例注册到该负载均衡器。

对于要附加的实例，必须满足以下条件：

- 该实例的running状态为 Amazon EC2。
- 用于启动实例的 AMI 必须仍然存在。
- 实例不是其他 Auto Scaling 组的成员。

- 实例会启动到自动扩缩组中定义的可用区之一。
- 如果 Auto Scaling 组具有附加的负载均衡器目标组或经典负载均衡器，则实例和负载均衡器必须都位于同一 VPC 中。

使用分离和附加将实例移至其他组

使用以下程序之一将实例与自动扩缩组分离，然后将其附加到其他自动扩缩组。

要从分离的实例创建新的自动扩缩组，请参阅使用现有实例创建 Auto Scaling 组 Amazon CLI（不推荐，创建启动配置）。

Console

将实例与自动扩缩组分离

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Instance management（实例管理）选项卡上的 Instances（实例）中，选择一个实例，然后选择 Actions（操作）、Detach（分离）。
4. 在分离实例对话框中，保持替换实例复选框处于选中状态以启动替换实例。清除该复选框可减少所需容量。
5. 当系统提示进行确认时，键入 **detach** 以确认从自动扩缩组中删除指定的实例，然后选择 Detach instance。

现在，您可以将该实例附加到其他自动扩缩组。

将实例附加到自动扩缩组

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. （可选）在导航窗格上的 Auto Scaling 下，选择 Auto Scaling Groups（Auto Scaling 组）。选择 Auto Scaling 组并验证 Auto Scaling 组的最大大小足以再添加一个实例。否则，在 Details（详细信息）选项卡上，增加最大容量。
3. 在导航窗格上的 Instances（实例）下，选择 Instances（实例），然后选择一个实例。
4. 依次选择操作、实例设置和附加到 Auto Scaling 组。

5. 在附加到 Auto Scaling 组页面上，为 Auto Scaling 组，输入组名称，然后选择附加。
6. 如果实例不符合条件，则会显示一条错误消息并提供详细信息。例如，实例可能没有位于与 Auto Scaling 组相同的可用区中。选择关闭并使用符合条件的自动扩缩组重试。

Amazon CLI

要分离和附加实例，请使用以下示例命令。将每个 *user input placeholder* 替换为您自己的信息。

将实例与自动扩缩组分离

1. 要描述当前实例，请使用以下describe-auto-scaling-instances命令。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

下面的示例显示运行此命令时产生的输出。

记下您打算从该组中移除的实例的 ID。在下一步骤中，您需要用到此 ID。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
```

```
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0c20ac468fa3049e8",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0787762faf1c28619",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0f280a4c58d319a8a",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        }
    ]
}
```

2. 要分离实例而不递减所需容量，请使用以下 detach-instances 命令。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
```

```
  --auto-scaling-group-name my-asg
```

要分离实例并递减所需容量，请包含 --should-decrement-desired-capacity 选项。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

现在，您可以将该实例附加到其他自动扩缩组。

将实例附加到自动扩缩组

1. 要将实例附加到其他自动扩缩组，请使用以下 attach-instances 命令。

```
aws autoscaling attach-instances --instance-ids i-05b4f7d5be44822a6 --auto-
scaling-group-name my-asg-for-testing
```

2. 要在连接实例后验证 Auto Scaling 组的大小，请使用以下describe-auto-scaling-groups命令。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg-
for-testing
```

以下示例响应显示该组有两个正在运行的实例，其中一个是您附加的实例。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg-for-testing",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "2",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            ...
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
```

```
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "2",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-00dcdfffdf5175890",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            ...
        }
    ]
}
```

临时从 Auto Scaling 组中移除实例

您可以将处于 `InService` 状态的实例置于 `Standby` 状态，更新实例或排查实例问题，然后将实例恢复运行状态。处于备用状态的实例仍是 Auto Scaling 组的一部分，但它们不会主动处理负载均衡器流量。

此功能可帮助您停止和启动实例或重启实例，而不必担心 Amazon A EC2 uto Scaling 会在运行状况检查或缩容事件期间终止实例。

例如，您可以随时更改启动模板或启动配置，以更改 Auto Scaling 组的 Amazon Machine Image (AMI)。Auto Scaling 组启动的任何后续实例都使用此 AMI。不过，Auto Scaling 组不会更新当前正在

运行的实例。您可以终止这些实例并让 Amazon A EC2 uto Scaling 替换它们，也可以使用实例刷新功能终止和替换这些实例。或者，您可以将实例置于备用状态，更新软件，然后将实例恢复运行。

从 Auto Scaling 组中分离实例类似于将实例置于备用状态。如果您想将实例附加到其他组或像独立实例一样管理实例并可能将其终止，则分离 EC2 实例可能会很有用。有关更多信息，请参阅 从自动扩缩组中分离或附加实例。

内容

备用状态的工作方式

备用状态按如下方式工作以帮助您临时从 Auto Scaling 组中删除实例：

1. 将实例置于备用状态。实例保持此状态，直至您退出备用状态。
2. 如果已有负载均衡器目标组或经典负载均衡器附上 Auto Scaling 组，则会将实例取消注册到该负载均衡器。如果为负载均衡器启用了 Connection Draining，预设情况下，Elastic Load Balancing 会等待 300 秒，然后再完成注销过程，这有助于正在进行的请求的完成。
3. 您可以更新实例或排查实例的问题。
4. 可通过退出备用状态，将实例恢复运行状态。
5. 如果已有负载均衡器目标组或经典负载均衡器附上 Auto Scaling 组，则会将实例注册到该负载均衡器。

有关自动扩缩组中实例的生命周期的更多信息，请参阅 Amazon A EC2 uto Scaling 实例生命周期。

注意事项

以下是将实例移入和移出备用状态时的注意事项：

- 当您将实例置于备用状态时，可以通过此操作减少所需容量，也可以将其保持不变。
  - 如果您选择不减少 Auto Scaling 组的所需容量，Amazon A EC2 uto Scaling 会启动一个实例来替换处于待机状态的实例。这样做是为了帮助您在一个或多个实例处于备用状态时保持应用程序的容量。

  - 如果您选择减少自动扩缩组的所需容量，这样会阻止启动实例来替换处于备用状态的实例。
- 将实例重新投入使用后，所需容量会增加，以反映自动扩缩组中有多少个实例。
- 要进行增加（和减少）操作，新的所需容量必须介于最小和最大组大小之间。否则，该操作将失败。
- 如果在将实例置于备用状态或通过退出备用状态使实例恢复服务后，发现您的 Auto Scaling 组在可用区域之间不平衡，Amazon A EC2 uto Scaling 会通过重新平衡可用区来进行补偿，除非您暂停该过程。AZRebalance有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 您需为处于备用状态的实例付费。

处于备用状态的实例的运行状况

Amazon A EC2 uto Scaling 不会对处于待机状态的实例执行运行状况检查。当实例处于备用状态时，其运行状况将反映您将实例置于备用状态之前，实例具有的状态。在您将实例重新投入使用之前，Amazon A EC2 uto Scaling 不会对该实例执行运行状况检查。

例如，如果您将运行正常的实例置于备用状态，然后将其终止，Amazon A EC2 uto Scaling 会继续将该实例报告为运行正常。如果您尝试将处于待机状态的已终止实例重新投入使用，Amazon A EC2 uto Scaling 会对该实例执行运行状况检查，确定该实例已终止且运行状况不佳，然后启动替代实例。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

通过将实例设置为备用来暂时移除实例

使用以下程序之一，通过将实例置于备用状态来暂时停止使用。

Console

临时删除实例

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Instance management（实例管理）选项卡的 Instances（实例）中，选择实例。
4. 选择 Actions（操作）和 Set to Standby（设置为备用）。
5. 在设置为备用对话框上，保持替代实例 复选框为选中状态，以启动替代实例。清除该复选框可减少所需容量。

6. 当系统提示您进行确认时，键入 **standby** 以确认将指定实例置于 Standby 状态，然后选择设置为备用。
7. 您可以根据需要更新实例或排查实例的问题。当您完成后，请继续下一步以将实例恢复运行。
8. 选择实例，选择操作，设置为 InService。在 “设置为 InService” 对话框中，选择 “设置为” InService。

Amazon CLI

要从自动扩缩组中暂时移除实例，请使用以下示例命令。将每个 *user input placeholder* 替换为您自己的信息。

临时删除实例

1. 使用以下 describe-auto-scaling-instances 命令确定要更新的实例。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

下面的示例显示运行此命令时产生的输出。

记下您打算从该组中移除的实例的 ID。在下一步骤中，您需要用到此 ID。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceId": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        ...
    ]
```

```
}
```

2. 将实例移动到 Standby 状态使用以下 enter-standby 命令。--should-decrement-desired-capacity 选项将减少所需容量以使 Auto Scaling 组不再启动替代实例。

```
aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

以下为响应示例。

```
{
    "Activities": [
        {
            "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was
 moved to standby
              in response to a user request, shrinking the capacity from 4 to
 3.",
            "StartTime": "2023-12-15T21:31:26.150Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

3. （可选）使用以下 describe-auto-scaling-instances 命令验证实例是否处于 Standby。

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

以下为响应示例。注意，实例状态此时为 Standby。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
```

```
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby"
        },
        ...
    ]
}
```

4. 您可以根据需要更新实例或排查实例的问题。当您完成后，请继续下一步以将实例恢复运行。
5. 使用以下 exit-standby 命令使实例恢复运行：

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-
group-name my-asg
```

以下为响应示例。

```
{
    "Activities": [
        {
            "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was
 moved out of standby in
               response to a user request, increasing the capacity from 3 to
 4.",
            "StartTime": "2023-12-15T21:46:14.678Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

6. （可选）使用以下 describe-auto-scaling-instances 命令验证实例是否已恢复运行。

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

以下为响应示例。请注意，实例状态为 InService。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        ...
    ]
}
```

删除 Auto Scaling 基础设施

要完全删除您的扩展基础设施，请完成以下任务。

任务

- 删除 Auto Scaling 组
- （可选）删除启动配置
- （可选）删除启动模板
- （可选）删除负载均衡器和目标组
- （可选）删除 CloudWatch警报

删除 Auto Scaling 组

当您删除 Auto Scaling 组时，其所需值、最小值和最大值设置为 0。因此，将会终止实例。删除实例还会删除任何关联的日志或数据，以及该实例上的任何卷。如果不想终止一个或多个实例，您可在删除 Auto Scaling 组之前分离它们。如果组具有扩展策略，则在删除组时，将会删除策略、基础警报操作以及不再具有关联操作的任何警报。

删除 Auto Scaling 组（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中自动扩缩组旁边的复选框并选择操作，删除。
3. 当系统提示进行确认时，键入 **`delete`** 以确认删除指定自动扩缩组，然后选择 Delete（删除）。

   Name（名称）列中的加载图标指示 Auto Scaling 组正在被删除。Desired（所需）、Min（最小）和 Max（最大）列显示 Auto Scaling 组具有 0 个实例。终止实例并删除组需要几分钟时间。刷新列表以查看当前状态。

要删除 Auto Scaling 组 (Amazon CLI)

使用以下delete-auto-scaling-group命令删除 Auto Scaling 组。如果该组有任何 EC2 实例，则此操作不起作用；它仅适用于实例为零的组。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

如果该组正在进行实例或扩展活动，请使用带`--force-delete`选项的delete-auto-scaling-group命令。这同时将终止 EC2 实例。当您从 Amazon Auto Scaling 控制台中删除 A EC2 uto Scaling 组时，控制台会使用此操作终止所有 EC2 实例，同时删除该组。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-
delete
```

（可选）删除启动配置

要保留启动配置以备将来使用，可跳过此步骤。

删除启动配置（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。

2. 在左侧导航窗格的自动扩缩下方，选择自动扩缩组。
3. 在页面顶部附近，选择启动配置。当提示您确认时，选择查看启动配置以确认您要查看启动配置页面。
4. 选择启动配置，选择 操作，然后单击 删除启动配置。
5. 当系统提示进行确认时，选择 Delete（删除）。

删除启动配置 (Amazon CLI)

使用以下 delete-launch-configuration 命令。

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-config
```

（可选）删除启动模板

您可以删除启动模板或仅删除启动模板的某个版本。在删除启动模板时，将删除其所有版本。

您可以跳过此步骤来保留启动模板以供将来使用。

删除启动模板（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的实例下，选择启动模板。
3. 选择启动模板，然后执行下列操作之一：

   - 选择 Actions（操作），然后选择 Delete template（删除模板）。当系统提示进行确认时，键入 **Delete** 以确认删除指定启动模板，然后选择 Delete（删除）。
   - 选择 Actions（操作），然后选择 Delete template version（删除模板版本）。选择要删除的版本，然后选择 Delete (删除)。

删除启动模板 (Amazon CLI)

使用以下 delete-launch-template 命令可删除您的模板及其所有版本。

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

或者，您也可以使用 delete-launch-template-versions 命令删除启动模板的特定版本。

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --
versions 1
```

（可选）删除负载均衡器和目标组

如果未在 Elastic Load Balancing 负载均衡器中关联 Auto Scaling 组，或者要保留负载均衡器以供将来使用，请跳过该步骤。

删除您的负载均衡器（控制台）

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格上的负载均衡下，选择负载均衡器。
3. 选择负载均衡器，然后依次选择 Actions（操作）和 Delete（删除）。
4. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

删除目标组（控制台）

1. 在导航窗格上的负载均衡下，选择目标组。
2. 选择目标组，然后依次选择 Actions（操作）、Delete（删除）。
3. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

删除与 Auto Scaling 组关联的负载均衡器 (Amazon CLI)

对于应用程序负载均衡器和网络负载均衡器，请使用以下delete-load-balancer和命令。delete-target-group

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

对于经典负载均衡器，请使用以下delete-load-balancer命令。

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

（可选）删除 CloudWatch警报

要删除与您的 Auto Scaling 组关联的 CloudWatch 警报，请完成以下步骤。例如，您可能有与步进扩缩策略或简单扩缩策略相关的警报。

> **Note**
>
> 删除 Auto Scaling 组会自动删除 Amazon A EC2 uto Scaling 为目标跟踪扩展策略管理的 CloudWatch 警报。

如果您的 Auto Scaling 组未与任何 CloudWatch 警报关联，或者您想保留警报以备将来使用，则可以跳过此步骤。

删除 CloudWatch 警报（控制台）

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格上，选择 Alarms（警报）。
3. 选择警报，然后选择 Action（操作）、Delete（删除）。
4. 当系统提示进行确认时，选择 Delete（删除）。

删除 CloudWatch 警报 (Amazon CLI)

使用 delete-alarms 命令。您可以一次删除一个或多个警报。例如，使用以下命令可删除 `Step-Scaling-AlarmHigh-AddCapacity` 和 `Step-Scaling-AlarmLow-RemoveCapacity` 警报：

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

替换自动扩缩组中的实例

Amazon A EC2 uto Scaling 提供的功能允许您在更新后替换 Auto Scaling 组中的亚马逊 EC2 实例，例如使用新的亚马逊系统映像 (AMI) 添加新的启动模板或添加新的实例类型。它还可以让您选择将更新包含在替换实例的同一操作中，从而帮助您简化更新。

本部分包含有助于您实现以下功能的信息：

- 启动实例刷新以替换 Auto Scaling 组中的实例。
- 声明描述所需配置的特定更新，并将 Auto Scaling 组更新为所需配置。
- 跳过替换已更新的实例。
- 使用检查点分阶段更新实例并在特定时间点对您的实例执行验证。
- 使用烘焙时间在实例刷新结束时暂停以验证实例的运行状况。
- 达到检查点时通过电子邮件接收通知。
- 使用回滚将自动扩缩组恢复到其以前使用的配置。
- 如果实例刷新由于某种原因失败，或者您指定的任何 Amazon CloudWatch 警报进入ALARM状态，则自动回滚。
- 限制实例的生命周期，以便在整个自动扩缩组中提供一致的软件版本和实例配置。

内容

使用实例刷新更新自动扩缩组中的实例

您可以使用实例刷新以更新自动扩缩组中的实例。配置更改要求您替换实例时，尤其是当您的自动扩缩组中包含大量实例时，此功能非常有用。

实例刷新可以提供帮助的情况包括：

- 跨自动扩缩组部署新的亚马逊机器映像（AMI）或用户数据脚本。您可以创建包含更改的新启动模板，然后使用实例刷新立即推出更新。
- 将您的实例迁移到新实例类型，以利用最新的改进和优化。

- 将自动扩缩组从使用启动配置切换为使用启动模板。您可以将启动配置复制到启动模板，然后使用实例刷新将实例更新为新模板。有关迁移到启动模板的更多信息，请参阅将自动扩缩组迁移到启动模板。

内容

自动扩缩组中实例刷新的工作原理

本主题描述了实例刷新的工作原理，并介绍有效使用该功能需要了解的主要概念。

内容

工作方式

要刷新自动扩缩组中的实例，您可以定义一个新配置，其中包含应用程序的最新版本以及您要进行的任何其他更新。然后，启动实例刷新，以根据该配置将现有实例替换为新实例。

执行实例刷新：

1. 创建新的启动模板或使用所需配置更改（例如，新的亚马逊机器映像（AMI））更新现有模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

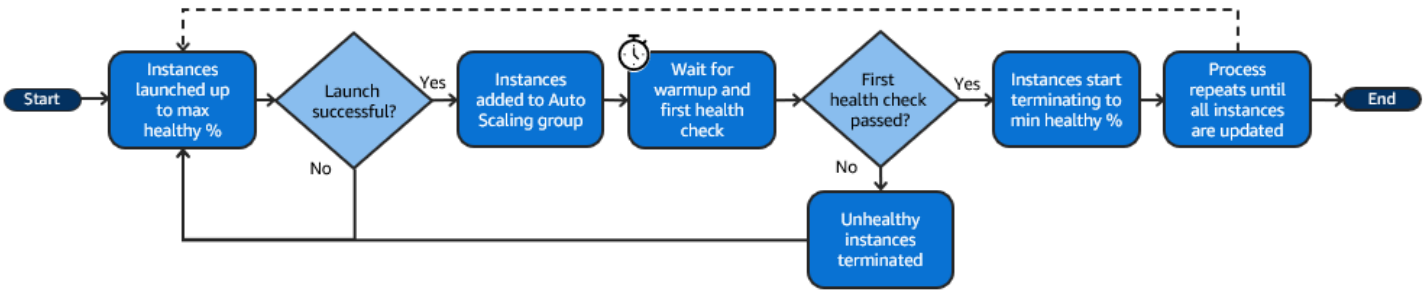
2. 使用 Amazon A EC2 uto Scaling 控制台或软件开发工具包开始刷新实例： Amazon CLI

   - 指定新启动模板或您创建的启动模板版本。这将用于启动新实例。
   - 设置首选的最小和最大运行正常百分比。这可以控制同时替换多少个实例，以及是否在终止旧实例之前启动新实例。
   - 配置任何可选设置，例如：
     - 检查点：在完成一定比例的替换后暂停实例刷新以验证进度。
     - 烘焙时间-在实例刷新结束时暂停以验证实例运行状况，然后才认为实例刷新已完成。
     - 跳过匹配：将旧实例与新配置进行比较，仅替换不匹配的实例。当您从控制台开始实例刷新时，默认情况下会开启跳过匹配。
     - 多个实例类型：作为所需配置的一部分，应用新的或更新的混合实例策略。

实例刷新开始后，Amazon A EC2 uto Scaling 将：

- 根据最小和最大运行正常百分比分批替换实例。
- 如果最小运行正常百分比设置为 100%，则应先启动新实例，然后再终止旧实例。这可确保始终保持所需容量。
- 检查实例的运行状况，让它们有时间进行预热，然后再替换更多实例。
- 终止并替换被发现运行状况不佳的实例。
- 实例刷新成功后，使用新的配置更改自动更新自动扩缩组设置。
- 先替换 `InService` 实例，然后再替换暖池中的实例。

以下流程图说明了将最小运行正常百分比设置为 100% 时终止前启动的行为。



> **Note**
>
> 仅当您尚未设置实例维护策略或需要覆盖实例维护策略时，才需要指定实例刷新的最小和最大运行正常百分比。有关更多信息，请参阅 实例维护策略。

同样，仅当您尚未启用默认预热或需要覆盖默认预热时，才需要为实例刷新指定实例预热期。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

核心概念

在您开始之前，请熟悉以下实例刷新核心概念：

最低运行正常百分比

最小运行正常百分比是实例刷新期间保持可用、运行正常且随时可用的所需容量的百分比。例如，如果最低运行正常百分比为 90%，最高运行正常百分比为 100%，那么一次将替换的容量百分比为 10%。如果新实例未通过运行状况检查，Amazon A EC2 uto Scaling 将终止并替换它们。如果实例刷新无法启动任何运行正常的实例，则它最终将失败，使组中其他的 90% 保持不变。如果新实例保持健康状态并完成预热期，Amazon A EC2 uto Scaling 可以继续替换其他实例。

实例刷新可以一次替换一个实例，一次替换多个实例，也可以一次替换全部实例。要一次替换一个实例，请将最低和最高运行正常百分比均设置为 100%。这会将实例刷新的行为更改为在终止之前启动，从而防止组的容量降至其所需容量的 100% 以下。要一次替换全部实例，将最低运行正常百分比设置为 0%。

最高运行正常百分比

最高运行正常百分比是替换实例时您的自动扩缩组可以增加到的所需容量的百分比。最小值和最大值之间的差值不能大于 100。范围越大，可以同时替换的实例的数量就会增加。

实例预热

实例预热是从新实例的状态变为 `InService` 之时起到其被视为已完成初始化之时止需要经过的时间段。在实例刷新期间，如果实例通过了运行状况检查，Amazon A EC2 uto Scaling 在确定新启动的实例运行状况良好后，不会立即开始替换下一个实例。它会等待预热期结束，然后继续替换下一个实例。如果应用程序在响应请求之前仍需要一些初始化时间，此功能可能很有用。

实例预热与默认实例预热的工作方式相同。因此，同样的扩缩注意事项也适用。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

所需配置

所需的配置是您希望 Amazon A EC2 uto Scaling 在您的 Auto Scaling 组中部署的新配置。例如，您可以为实例指定新的启动模板和新的实例类型。在实例刷新期间，Amazon A EC2 uto Scaling 会将 Auto Scaling 组更新为所需的配置。如果在实例刷新期间发生扩展事件，Amazon A EC2 uto

Scaling 会启动具有所需配置而不是组当前设置的新实例。实例刷新成功后，Amazon A EC2 uto Scaling 会更新 Auto Scaling 组设置，以反映您在实例刷新过程中指定的新的所需配置。

跳过匹配

跳过匹配会让 Amazon A EC2 uto Scaling 忽略已经有最新更新的实例。这样，您就无需替换超过所需数量的实例。当您希望确保自动扩缩组使用特定版本的启动模板，并且只替换那些使用其他版本的实例时，这是很有用的。

检查点

检查点是实例刷新在指定时间内暂停的时间点。实例刷新可以包含多个检查点。Amazon A EC2 uto Scaling 会为每个检查点发出事件。因此，您可以添加一条 EventBridge 规则，将事件发送到目标（例如 Amazon SNS），以便在到达检查点时收到通知。达到某个检查点后，您将有机会验证您的部署。如果发现任何问题，您可以取消实例刷新，或回滚实例。分阶段部署更新的能力是检查点的一个关键优势。如果不使用检查点，则会持续执行滚动替换。

要详细了解在开始实例刷新时可以配置的所有默认设置，请参阅 了解实例刷新的默认值。

运行状况检查宽限期

Amazon A EC2 uto Scaling 根据您的 Auto Scaling 组使用的运行状况检查的状态来确定实例是否运行正常。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

为确保这些运行状况检查尽快开始，请勿将组的运行状况检查宽限期设置得过高，而应设置得足够高，以便 Elastic Load Balancing 运行状况检查确定目标是否可用于处理请求。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

实例类型兼容性

在更改实例类型之前，最好验证它是否适用于您的启动模板。这确认了与您指定的 AMI 的兼容性。例如，如果您从半虚拟化（PV）AMI 启动了原始实例，但希望更改为仅受硬件虚拟机（HVM）AMI 支持的最新一代实例类型。在这种情况下，您必须在启动模板中使用 HVM AMI。

要在不启动实例的情况下确认实例类型的兼容性，请使用带 `--dry-run` 选项的 run-instances 命令，如以下示例所示。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --
dry-run
```

有关如何确定兼容性的信息，请参阅 Amazon EC2 用户指南中更改实例类型的兼容性。

限制

- 总持续时间：实例刷新可继续主动替换实例的最长时间为 14 天。
- 加权组特有的行为差异：如果混合实例组配置的实例权重大于或等于该组的所需容量，则 Amazon A EC2 uto Scaling 可能会同时替换所有`InService`实例。为避免这种情况，请遵循 配置自动扩缩组以使用实例权重 主题中的建议。指定所需容量，该容量大于在自动扩缩组中使用权重时的最大权重。
- 一小时超时：当实例刷新由于等待替换处于待机状态的实例或受保护而无法继续进行替换，或者新实例未通过运行状况检查时，Amazon A EC2 uto Scaling 会继续重试一个小时。它还将提供状态消息以帮助您解决问题。如果问题在一小时后仍然存在，则操作失败。这样做的目的是在出现临时问题时给它时间恢复。
- 通过用户数据部署代码：跳过匹配不会检查通过用户数据脚本部署的代码更改。如果您使用用户数据提取新代码并在新实例上安装这些更新，则建议您关闭跳过匹配功能，以确保所有实例都能收到您的最新代码，即使没有启动模板版本更新也是如此。
- 更新限制：如果您在具有所需配置的实例刷新处于活动状态时，尝试更新自动扩缩组的启动模板、启动配置或混合实例策略，则请求将失败，并显示以下验证错误：`An active instance refresh with a desired configuration exists. All configuration options derived from the desired configuration are not available for update while the instance refresh is active.`

了解实例刷新的默认值

在开始刷新实例之前，您可以自定义影响实例刷新的各种首选项。某些首选项默认值会有所不同，具体取决于您使用的是控制台还是命令行（Amazon CLI 或 Amazon SDK）。

下表列出了实例刷新设置的默认值。

设置	Amazon CLI 或者 Amazon SDK	亚马逊 A EC2 uto Scaling 控制台
CloudWatch 警报	已禁用（空白）	已禁用
自动回滚	已禁用 (`false`)	已禁用
烘烤时间	零	零

设置	Amazon CLI 或者 Amazon SDK	亚马逊 A EC2 uto Scaling 控制台
检查点	已禁用 (`false`)	已禁用
检查点延迟	1 小时（`3600` 秒）	1 小时
实例预热	默认实例预热（如果已定义），否则为运行状况检查宽限期。	默认实例预热（如果已定义），否则为运行状况检查宽限期。
最高运行正常百分比	根据实例维护策略而有所不同。如果没有实例维护策略，则默认为 100%（空白）。	根据实例维护策略而有所不同。如果没有实例维护策略，则默认为 100%（空白）。
最低运行正常百分比	根据实例维护策略而有所不同。如果没有实例维护策略，则默认为 90%。	根据实例维护策略而有所不同。如果没有实例维护策略，则默认为 90%。
横向缩减保护实例	Wait	忽略
跳过匹配	已禁用 (`false`)	已启用
备用实例	Wait	忽略

各设置的描述如下所示：

CloudWatch 警报 (**`AlarmSpecification`**)

CloudWatch 警报规格。 CloudWatch 警报可用于识别任何问题，并在警报进入ALARM状态时使操作失败。有关更多信息，请参阅 使用自动回滚启动实例刷新。

自动回滚 (**`AutoRollback`**)

控制在实例刷新失败时，Amazon A EC2 uto Scaling 是否将 Auto Scaling 组回滚到之前的配置。有关更多信息，请参阅 通过手动或自动回滚撤消更改。

烘烤时间 (**`BakeTime`**)

在实例刷新结束时等待的时间长度，然后才认为实例刷新已完成。

检查点 (**CheckpointPercentages**)

控制 Amazon A EC2 uto Scaling 是否分阶段替换实例。如果您需要在替换所有实例之前对实例进行验证，则此功能非常有用。有关更多信息，请参阅 将检查点添加到实例刷新。

检查点延迟 (**CheckpointDelay**)

在到达检查点之后与继续操作之前需要等待的时间量（以秒为单位）。有关更多信息，请参阅 将检查点添加到实例刷新。

实例预热 (**InstanceWarmup**)

以秒为单位的时间段，在此期间，Amazon A EC2 uto Scaling 会等待一个新实例被视为已完成初始化，然后再继续替换下一个实例。如果您已经正确定义了自动扩缩组的默认实例预热，则无需更改实例预热（除非您想覆盖默认）。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

最高运行正常百分比 (**MaxHealthyPercentage**)

替换实例时可增加到自动扩缩组所需容量的百分比。

最低运行正常百分比 (**MinHealthyPercentage**)

在可继续操作之前必须处于服务状态、运行正常且准备好使用的自动扩缩组所需容量的百分比。

横向缩减保护实例 (**ScaleInProtectedInstances**)

控制 Amazon A EC2 uto Scaling 在发现受保护免受缩容保护的实例时会做什么。有关这些实例的更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

Amazon A EC2 uto Scaling 提供以下选项：

- 替换（`Refresh`）：替换受横向缩减保护的实例。
- 忽略（`Ignore`）：忽略受横向缩减保护的实例，并继续替换未受保护的实例。
- 等待（`Wait`）：等待一小时，以删除横向缩减保护。如果您不这样做，实例刷新将失败。

跳过匹配 (**SkipMatching**)

控制 Amazon A EC2 uto Scaling 是否跳过替换与所需配置相匹配的实例。如果未指定所需配置，则会跳过替换相关实例，这些实例的启动模板和实例类型与自动扩缩组在实例刷新启动之前使用的启动模板和实例类型相同。有关更多信息，请参阅 使用实例刷新和跳过匹配。

备用实例 (**StandbyInstances**)

控制 Amazon A EC2 uto Scaling 在发现实例处于`Standby`状态时会执行的操作。有关这些实例的更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

Amazon A EC2 uto Scaling 提供以下选项：

- 终止（Terminate）：终止处于 Standby 状态的实例。
- 忽略（Ignore）：忽略处于 Standby 状态的实例，并继续替换处于 InService 状态的实例。
- 等待（Wait）：等待一小时让实例恢复服务。如果您不这样做，实例刷新将失败。

使用 Amazon Web Services Management Console 或开始刷新实例 Amazon CLI

> **Important**
>
> 您可以回滚正在进行的实例刷新，以撤消任何更改。要使此功能起作用，自动扩缩组在启动实例刷新之前必须满足使用回滚的先决条件。有关更多信息，请参阅 通过手动或自动回滚撤消更改。

以下过程可帮助您使用 Amazon Web Services Management Console 或启动实例刷新 Amazon CLI。

启动实例刷新（控制台）

如果这是您第一次开启实例刷新，则使用控制台将有助于您了解可用的功能和选项。

在控制台中开启实例刷新（基本步骤）

如果您以前没有为 Auto Scaling 组定义混合实例策略，请使用下面的步骤。如果您以前定义了混合实例策略，请参阅 在控制台中开启实例刷新（混合实例组） 以开启实例刷新。

启动实例刷新

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   将在 Auto Scaling group（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Instance refresh（实例刷新）选项卡上的 Active instance refresh（活跃实例刷新）中，选择 Start instance refresh（开启实例刷新）。
4. 对于可用性设置，请执行以下操作：

   a. 对于实例替换方法：

- 如果您尚未在自动扩缩组上设置实例维护策略，则实例替换方法的默认设置为终止并启动。这是实例刷新的旧版默认行为。
- 如果您在自动扩缩组上设置了实例维护策略，它将为实例替换方法提供默认值。要覆盖实例维护策略，请选择覆盖。覆盖仅适用于当前的实例刷新。下次启动实例刷新时，这些值将重置为实例维护策略的默认值。

以下步骤介绍如何更新实例替换方法。

i. 选择以下实例替换方法之一：

- 终止前启动：必须先配置新实例，然后才能终止现有实例。对于偏向于可用性而不是成本节约的应用程序来说，这是一个不错的选择。
- 终止并启动：在终止现有实例的同时配置新实例。对于偏向于节省成本而不是可用性的应用程序来说，这是一个不错的选择。对于启动容量不应超过当前可用容量的应用程序来说，它也是一个不错的选择。
- 自定义行为：此选项允许您为替换实例时所需的可用容量设置自定义的最小和最大范围。这可以帮助您在成本和可用性之间取得适当的平衡。

ii. 对于设置运行正常百分比，为以下一个或两个字段输入值。启用字段因所选择的实例替换方法的选项而异。

- 最小：设置继续进行实例刷新所需的最低运行正常百分比。
- 最大：设置实例刷新期间可能的最高运行正常百分比。

iii. 展开根据当前群组规模查看替换期间的估计临时容量部分，以确认最小值和最大值如何适用于您的组。使用的确切值取决于所需的容量值，如果组发生扩缩，该值将发生变化。

iv. 展开为无效的替换大小设置回退行为部分，然后选择是通过违反最高运行正常百分比来确定可用性的优先级，还是通过违反最低运行正常百分比。

对于非常小的组，不建议保留默认的违反最低运行正常百分比选项。如果自动扩缩组中只有一个实例时，启动实例刷新可能会导致中断。

如果您使用的自动扩缩组还没有实例维护策略，则此步骤将配置回退行为。此选项不可用，也不会在您的组有实例维护策略时出现。此选项也仅适用于终止并启动替换方法。为了优先考虑可用性，其他替换方法将违反最高运行正常百分比。

b. 对于实例预热，输入从新实例的状态变为 `InService` 之时起到其完成初始化时为止需要经过的秒数。Amazon A EC2 uto Scaling 在继续替换下一个实例之前会等待这段时间。

预热时，新启动的实例也不会计入自动扩缩组的聚合实例指标（例如 CPUUtilization、NetworkIn 和 NetworkOut）。如果已在 Auto Scaling 组中添加了扩展策略，扩展活动将并行运行。如果为实例刷新预热期设置了较长的时间间隔，则新启动的实例需要更多的时间才能显示在指标中。因此，充足的预热期会阻止 Amazon A EC2 uto Scaling 根据陈旧的指标数据进行扩展。

如果您已经为自动扩缩组正确定义了默认实例预热，则无需更改实例预热。但是，如果您想覆盖默认，则可以为此选项设置值。有关设置默认实例预热的更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

5. 对于刷新设置，请执行以下操作：

   a. （可选）对于 Checkpoints（检查点），选择 Enable checkpoints（启用检查点），以使用递增量或分阶段的实例刷新方法来替换实例。这将为在替换集之间进行验证提供额外时间。如果您选择不启用检查点，将以一次近乎连续的操作替换这些实例。

   如果您启用检查点，请参阅 启用检查点（控制台） 以了解其他步骤。

   b. （可选）对于烘焙时间，请指定在实例刷新结束后等待的时间，然后才认为实例刷新已完成。

   c. 启用或关闭 Skip matching（跳过匹配）：

   - 若要跳过替换已与您的启动模板匹配的实例，请将启用跳过匹配复选框保持为选中状态。
   - 如果通过清除此复选框关闭跳过匹配，则可替换所有实例。

   当您启用跳过匹配时，您可以设置新启动模板或启动模板的新版本，而不是使用现有启动模板。请在启动实例刷新页面的所需配置部分中执行此操作。

   > **Note**
   >
   > 要使用跳过匹配功能更新当前使用启动配置的 Auto Scaling 组，您必须在 Desired configuration（所需配置）中选择启动模板。不支持通过启动配置跳过匹配。

   d. 对于备用实例，选择忽略、终止或等待。这决定了当实例处于 Standby 状态时会发生什么。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

   如果您选择等待，则必须采取其他步骤才能使这些实例恢复服务。如果您不这样做，则实例刷新会替换所有 InService 实例并等待一小时。然后，如果还有任何 Standby 实例，则实例刷新将失败。为防止出现这种情况，请改为选择忽略或终止这些实例。

e. 对于横向缩减保护实例，选择忽略、替换或等待。这决定了如果找到横向缩减保护实例会发生什么。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

如果您选择等待，则必须采取其他措施来删除这些实例的横向缩减保护。如果您不这样做，则实例刷新会替换所有未受保护的实例并等待一小时。然后，如果还有任何横向缩减保护实例，则实例刷新将失败。为防止出现这种情况，请改为选择忽略或替换这些实例。

6. （可选）对于CloudWatch 警报，请选择启用 CloudWatch 警报，然后选择一个或多个警报。CloudWatch 警报可用于识别任何问题，并在警报进入ALARM状态时使操作失败。有关更多信息，请参阅 使用自动回滚启动实例刷新。

7. （可选）展开所需配置部分，以指定您要对自动扩缩组进行的任何更新。

对于此步骤，您可以选择使用 JSON 或 YAML 语法来编辑参数值，而不是在控制台界面中进行选择。为此，请选择 Use code editor（使用代码编辑器），而不是 Use console interface（使用控制台界面）。以下步骤介绍如何使用控制台界面进行选择。

a. 对于 Update launch template（更新启动模板）：

- 如果您尚未为自动扩缩组创建新启动模板或新启动模板版本，则不要选中此复选框。
- 如果您已创建新启动模板或新启动模板版本，请选中此复选框。当您选择此选项时，Amazon A EC2 uto Scaling 会向您显示当前的启动模板和当前的启动模板版本。它还列出了任何其他可用版本。选择启动模板，然后选择版本。

  在您选择版本后，可以看到版本信息。这是作为实例刷新的组成部分替换实例时将要使用的启动模板的版本。如果实例刷新成功，则每当新实例启动时（例如在组扩展时），也将使用此版本的启动模板。

b. 对于 Choose a set of instance types and purchase options to override the instance type in the launch template（选择一组实例类型和购买选项以覆盖启动模板中的实例类型）：

- 如果您想使用您在启动模板中指定的实例类型和购买选项，请不要选中此复选框。
- 如果您要覆盖启动模板中的实例类型或运行竞价型实例，请选中此复选框。您可以手动添加每种实例类型，也可以选择主实例类型和为您检索任何其他匹配实例类型的建议选项。如果您计划启动竞价型实例，我们建议添加几种不同的实例类型。这样，如果您选择的可用区域中的实例容量不足，Amazon A EC2 uto Scaling 就可以启动另一种实例类型。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

> **Warning**
>
> 不要将竞价型实例用于无法处理竞价型实例中断的应用程序。如果 Amazon EC2 Spot 服务需要回收容量，则可能会出现中断。

如果您选中此复选框，请确保启动模板尚未请求竞价型实例。您无法使用请求竞价型实例的启动模板来创建使用多种实例类型并启动竞价型和按需型实例的自动扩缩组。

> **Note**
>
> 要在当前使用启动配置的 Auto Scaling 组上配置这些选项，您必须在 Update launch template（更新启动模板）中选择启动模板。不支持覆盖您的启动配置中的实例类型。

8. （可选）对于回滚设置，选择启用自动回滚以在实例刷新失败时自动回滚实例刷新。

   只有当自动扩缩组满足使用回滚的先决条件时，才能启用此设置。

   有关更多信息，请参阅 通过手动或自动回滚撤消更改。

9. 审核您的所有选择，以确认所有选择的设置都正确。

   此时，最好验证当前更改与建议更改之间的差异不会以意外或不需要的方式影响您的应用程序。要确认您的实例类型与您的启动模板兼容，请参阅 实例类型兼容性。

10. 如果您对实例刷新选择感到满意，请选择启动实例刷新。

在控制台中开启实例刷新（混合实例组）

如果您已使用混合实例策略创建了 Auto Scaling 组，则请使用以下步骤。如果您尚未为您的组定义混合实例策略，请参阅 在控制台中开启实例刷新（基本步骤） 以开启实例刷新。

启动实例刷新

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   将在 Auto Scaling group（Auto Scaling 组）页面底部打开一个拆分窗格。

3. 在 Instance refresh（实例刷新）选项卡上的 Active instance refresh（活跃实例刷新）中，选择 Start instance refresh（开启实例刷新）。
4. 对于可用性设置，请执行以下操作：

   a. 对于实例替换方法：

      - 如果您尚未在自动扩缩组上设置实例维护策略，则实例替换方法的默认设置为终止并启动。这是实例刷新的旧版默认行为。
      - 如果您在自动扩缩组上设置了实例维护策略，它将为实例替换方法提供默认值。要覆盖实例维护策略，请选择覆盖。覆盖仅适用于当前的实例刷新。下次启动实例刷新时，这些值将重置为实例维护策略的默认值。

      以下步骤介绍如何更新实例替换方法。

      i. 选择以下实例替换方法之一：

         - 终止前启动：必须先配置新实例，然后才能终止现有实例。对于偏向于可用性而不是成本节约的应用程序来说，这是一个不错的选择。
         - 终止并启动：在终止现有实例的同时配置新实例。对于偏向于节省成本而不是可用性的应用程序来说，这是一个不错的选择。对于启动容量不应超过当前可用容量的应用程序来说，它也是一个不错的选择。
         - 自定义行为：此选项允许您为替换实例时所需的可用容量设置自定义的最小和最大范围。这可以帮助您在成本和可用性之间取得适当的平衡。

      ii. 对于设置运行正常百分比，为以下一个或两个字段输入值。启用字段因所选择的实例替换方法的选项而异。

         - 最小：设置继续进行实例刷新所需的最低运行正常百分比。
         - 最大：设置实例刷新期间可能的最高运行正常百分比。

      iii. 展开根据当前群组规模查看替换期间的估计临时容量部分，以确认最小值和最大值如何适用于您的组。使用的确切值取决于所需的容量值，如果组发生扩缩，该值将发生变化。

      iv. 展开为无效的替换大小设置回退行为部分，然后选择是通过违反最高运行正常百分比来确定可用性的优先级，还是通过违反最低运行正常百分比。

      对于非常小的组，不建议保留默认的违反最低运行正常百分比选项。如果自动扩缩组中只有一个实例时，启动实例刷新可能会导致中断。

如果您使用的自动扩缩组还没有实例维护策略，则此步骤将配置回退行为。此选项不可用，也不会在您的组有实例维护策略时出现。此选项也仅适用于终止并启动替换方法。为了优先考虑可用性，其他替换方法将违反最高运行正常百分比。

b. 对于实例预热，输入从新实例的状态变为 `InService` 之时起到其完成初始化时为止需要经过的秒数。Amazon A EC2 uto Scaling 在继续替换下一个实例之前会等待这段时间。

预热时，新启动的实例也不会计入自动扩缩组的聚合实例指标（例如 `CPUUtilization`、`NetworkIn` 和 `NetworkOut`）。如果已在 Auto Scaling 组中添加了扩展策略，扩展活动将并行运行。如果为实例刷新预热期设置了较长的时间间隔，则新启动的实例需要更多的时间才能显示在指标中。因此，充足的预热期会阻止 Amazon A EC2 uto Scaling 根据陈旧的指标数据进行扩展。

如果您已经为自动扩缩组正确定义了默认实例预热，则无需更改实例预热。但是，如果您想覆盖默认，则可以为此选项设置值。有关设置默认实例预热的更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

5. 对于刷新设置，请执行以下操作：

a. （可选）对于 Checkpoints（检查点），选择 Enable checkpoints（启用检查点），以使用递增量或分阶段的实例刷新方法来替换实例。这将为在替换集之间进行验证提供额外时间。如果您选择不启用检查点，将以一次近乎连续的操作替换这些实例。

如果您启用检查点，请参阅 启用检查点（控制台） 以了解其他步骤。

b. 启用或关闭 Skip matching（跳过匹配）：

- 若要跳过替换已与您的启动模板匹配的实例和任何实例类型覆盖，请将启用跳过匹配复选框保持为选中状态。
- 如果您选择通过清除此复选框关闭跳过匹配，则可替换所有实例。

当您启用跳过匹配时，您可以设置新启动模板或启动模板的新版本，而不是使用现有启动模板。请在启动实例刷新页面的所需配置部分中执行此操作。您还可以在 Desired configuration（所需配置）中更新实例类型覆盖。

c. 对于备用实例，选择忽略、终止或等待。这决定了当实例处于 `Standby` 状态时会发生什么。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

如果您选择等待，则必须采取其他步骤才能使这些实例恢复服务。如果您不这样做，则实例刷新会替换所有 InService 实例并等待一小时。然后，如果还有任何 Standby 实例，则实例刷新将失败。为防止出现这种情况，请改为选择忽略或终止这些实例。

d. 对于横向缩减保护实例，选择忽略、替换或等待。这决定了如果找到横向缩减保护实例会发生什么。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

如果您选择等待，则必须采取其他措施来删除这些实例的横向缩减保护。如果您不这样做，则实例刷新会替换所有未受保护的实例并等待一小时。然后，如果还有任何横向缩减保护实例，则实例刷新将失败。为防止出现这种情况，请改为选择忽略或替换这些实例。

6. （可选）对于CloudWatch 警报，请选择启用 CloudWatch 警报，然后选择一个或多个警报。CloudWatch 警报可用于识别任何问题，并在警报进入ALARM状态时使操作失败。有关更多信息，请参阅 使用自动回滚启动实例刷新。

7. 在 Desired configuration（所需配置）部分中，执行以下操作。

对于此步骤，您可以选择使用 JSON 或 YAML 语法来编辑参数值，而不是在控制台界面中进行选择。为此，请选择 Use code editor（使用代码编辑器），而不是 Use console interface（使用控制台界面）。以下步骤介绍如何使用控制台界面进行选择。

a. 对于 Update launch template（更新启动模板）：

- 如果您尚未为自动扩缩组创建新启动模板或新启动模板版本，则不要选中此复选框。
- 如果您已创建新启动模板或新启动模板版本，请选中此复选框。当您选择此选项时，Amazon A EC2 uto Scaling 会向您显示当前的启动模板和当前的启动模板版本。它还列出了任何其他可用版本。选择启动模板，然后选择版本。

  在您选择版本后，可以看到版本信息。这是作为实例刷新的组成部分替换实例时将要使用的启动模板的版本。如果实例刷新成功，则每当新实例启动时（例如在组扩展时），也将使用此版本的启动模板。

b. 对于 Use these settings to override the instance type and purchase option defined in the launch template（使用这些设置覆盖在启动模板中定义的实例类型和购买选项）：

默认情况下，此复选框处于选中状态。Amazon A EC2 uto Scaling 使用当前在 Auto Scaling 组的混合实例策略中设置的值填充每个参数。仅更新您要更改的参数的值。有关这些设置的指导，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

> **Warning**
>
> 我们建议您不要清除此复选框。仅当您希望停止使用混合实例策略时才清除它。实例刷新成功后，Amazon A EC2 uto Scaling 会更新您的群组以匹配所需的配置。如果不再包含混合实例策略，Amazon A EC2 uto Scaling 会逐渐终止当前正在运行的所有竞价型实例，并将其替换为按需实例。或者，如果您的启动模板请求竞价型实例，则 Amazon A EC2 uto Scaling 会逐渐终止当前正在运行的所有按需实例，并将其替换为竞价型实例。

8. （可选）对于回滚设置，选择启用自动回滚以在实例刷新因任何原因而失败时自动回滚。

   只有当自动扩缩组满足使用回滚的先决条件时，才能启用此设置。

   有关更多信息，请参阅 通过手动或自动回滚撤消更改。

9. 审核您的所有选择，以确认所有选择的设置都正确。

   此时，最好验证当前更改与建议更改之间的差异不会以意外或不需要的方式影响您的应用程序。要确认您的实例类型与您的启动模板兼容，请参阅 实例类型兼容性。

   如果您对实例刷新选择感到满意，请选择启动实例刷新。

启动实例刷新（Amazon CLI）

启动实例刷新

使用以下start-instance-refresh命令从启动实例刷新 Amazon CLI。您可以在 JSON 配置文件中指定要更改的任何首选项。引用配置文件时，请提供该文件的路径和名称，如以下示例所示。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 50,
      "AutoRollback": true,
      "ScaleInProtectedInstances": Ignore,
```

```
        "StandbyInstances": Terminate
    }
}
```

如果不提供首选项，则会使用默认值。有关更多信息，请参阅 了解实例刷新的默认值。

输出示例：

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

使用 Amazon Web Services Management Console 或监控实例刷新 Amazon CLI

您可以使用或监控正在进行的实例刷新，也可以查看过去六周内过去的实例刷新状态。 Amazon Web Services Management Console Amazon CLI

监控和检查实例刷新的状态

要监控和检查实例刷新的状态，请使用以下方法之一：

Console

> **Tip**
>
> 在此过程中，应已显示已命名的列。要显示隐藏的列或更改显示的行数，请选择该部分右上角的齿轮图标以打开首选项模式。根据需要更新设置，然后选择确认。

监控和检查实例刷新的状态（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Instance refresh（实例刷新）选项卡上的 Instance refresh history（实例刷新历史记录）下，您可以通过查看 Status（状态）列来确定您的请求的状态。操作在初始化时进入

Pending 状态。然后，状态应快速更改为 InProgress。所有实例更新后，状态将更改为 Successful。

4. 通过查看该组的扩缩活动，您可以进一步监控正在进行的活动是成功还是失败。在活动选项卡上的活动历史记录下，当实例刷新开始时，您会看到实例终止时的条目以及启动实例时的另一组条目。如果您有许多扩缩活动，则可以通过选择活动历史记录顶部的 > 图标来查看其中的更多活动。有关对可能导致活动失败的问题进行排查的信息，请参阅对 Amazon A EC2 uto Scaling 中的问题进行故障排除。
5. （可选）在实例管理选项卡的实例下，您可以根据需要查看特定实例的进度。

Amazon CLI

监控和检查实例刷新的状态（Amazon CLI）

使用以下 describe-instance-refreshes 命令。

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

下面是示例输出。

实例刷新按开始时间排序。首先描述仍在进行的实例刷新。

```
{
  "InstanceRefreshes":[
    {
      "InstanceRefreshId":"08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
      "AutoScalingGroupName":"my-asg",
      "Status":"InProgress",
      "StatusReason":"Waiting for instances to warm up before continuing. For
 example: i-0645704820a8e83ff is warming up.",
      "StartTime":"2023-11-24T16:46:52+00:00",
      "PercentageComplete":50,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
```

```
      }
    },
    {
      "InstanceRefreshId":"0e151305-1e57-4a32-a256-1fd14157c5ec",
      "AutoScalingGroupName":"my-asg",
      "Status":"Successful",
      "StartTime":"2023-11-22T13:53:37+00:00",
      "EndTime":"2023-11-22T13:59:45+00:00",
      "PercentageComplete":100,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
      }
    }
  ]
}
```


通过查看该组的扩缩活动，您可以进一步监控正在进行的活动是成功还是失败。扩缩活动还可以帮助您深入了解更多详细信息，以帮助您排查实例刷新问题。有关更多信息，请参阅 对 Amazon A EC2 uto Scaling 中的问题进行故障排除。

实例刷新状态

当您启动实例刷新时，它会进入待处理状态。它从 “待处理” 传递到，InProgress直到达到 “成功”、“失败”、“已取消” 或RollbackFailed。RollbackSuccessful

实例刷新可以具有以下状态：

状态	描述
待处理	请求已创建，但实例刷新尚未开始。
InProgress	实例刷新正在进行中。
成功	实例刷新已成功完成。

状态	描述
已失败	实例刷新未能完成。您可以使用状态原因和扩展活动进行故障排除。
正在取消	正在取消进行中的实例刷新。
已取消	实例刷新已取消。
RollbackInProgress	正在回滚实例刷新。
RollbackFailed	回滚未能完成。您可以使用状态原因和扩展活动进行故障排除。
RollbackSuccessful	回滚已成功完成。
烘焙	在实例刷新完成实例更新后，等待指定的烘焙时间。

使用 Amazon Web Services Management Console 或取消实例刷新 Amazon CLI

您可以取消仍在进行的实例刷新。在完成该操作后，您无法取消。

取消实例刷新不会回滚任何已替换的实例。要回滚对您的实例所做的更改，请改为执行回滚。有关更多信息，请参阅 通过手动或自动回滚撤消更改。

主题

- 取消实例刷新（控制台）
- 取消实例刷新（Amazon CLI）

取消实例刷新（控制台）

取消实例刷新

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。
3. 在实例刷新选项卡的活跃实例刷新中，选择操作、取消。
4. 当系统提示进行确认时，选择 Confirm。

实例刷新状态设置为正在取消。取消完成后，实例刷新的状态设置为已取消。

取消实例刷新（Amazon CLI）

取消实例刷新

使用中的cancel-instance-refresh命令 Amazon CLI 并提供 Auto Scaling 组名。

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

输出示例：

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

通过手动或自动回滚撤消更改

您可以回滚仍在进行的实例刷新。完成后，您无法将其回滚。但是，您可以通过开始新的实例刷新来再次更新您的自动扩缩组。

回滚时，Amazon A EC2 uto Scaling 会替换迄今为止已部署的实例。新实例与您在启动实例刷新之前保存在自动扩缩组中的配置相匹配。

Amazon A EC2 uto Scaling 提供了以下回滚方式：

- 手动回滚：手动开始回滚，以撤消部署到回滚点的内容。
- 自动回滚：如果实例刷新由于某种原因失败或您指定的任何 CloudWatch 警报进入状态，Amazon A EC2 uto Scaling 会自动撤消已部署的ALARM内容。

内容

注意事项

在使用回滚时，请注意以下几点：

- 只有在启动实例刷新时指定所需的配置时，回滚选项才可用。
- 如果启动模板是特定编号的版本，则只能回滚到该版本的先前版本。如果自动扩缩组配置为使用`$Latest`或`$Default`启动模板版本，则回滚选项不可用。
- 您也无法回滚到配置为使用 Param Amazon Systems Manager eter Store 中的 AMI 别名的启动模板。
- 您上次保存在自动扩缩组中的配置必须处于稳定状态。如果未处于稳定状态，回滚工作流程仍会发生，但最终会失败。在您解决问题之前，自动扩缩组可能处于失败状态，无法再成功启动实例。这可能会影响服务或应用程序的可用性。

手动开始回滚

Console

手动开始实例刷新回滚（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。
3. 在实例刷新选项卡的活跃实例刷新中，依次选择操作、开始回滚。
4. 当系统提示进行确认时，选择 Confirm。

Amazon CLI

手动开始实例刷新回滚 (Amazon CLI)

使用中的rollback-instance-refresh命令 Amazon CLI 并提供 Auto Scaling 组名。

```
aws autoscaling rollback-instance-refresh --auto-scaling-group-name my-asg
```

输出示例：

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

使用自动回滚启动实例刷新

使用自动回滚功能，您可以在实例刷新失败时（例如出现错误或指定的 Amazon CloudWatch 警报进入ALARM状态时）自动回滚实例刷新。

如果您启用了自动回滚，并且在替换实例时出现错误，则实例刷新会尝试在一小时内完成所有替换，然后才会失败并回滚。这些错误通常是由 EC2启动失败、运行状况检查配置错误或不忽略或不允许终止处于Standby状态或受保护无法扩展的实例等原因造成的。

指定 CloudWatch 警报是可选的。要指定警报，首先需要创建警报。您可以指定指标警报和复合警报。有关创建警报的信息，请参阅 Amazon CloudWatch 用户指南。以 Elastic Load Balancing 指标为例，如果您使用应用程序负载均衡器，则可以使用 HTTPCode_ELB_5XX_Count 和 HTTPCode_ELB_4XX_Count 指标。

注意事项

- 如果您指定了 CloudWatch 警报但未启用 auto rollback，并且警报状态变为ALARM，则实例刷新失败而不回滚。
- 启动实例刷新时，您最多可以选择 10 个警报。
- 选择 CloudWatch 警报时，警报必须处于兼容状态。如果警报状态为 INSUFFICIENT_DATA 或 ALARM，则在尝试启动实例刷新时会收到错误消息。
- 在创建警报供 Amazon A EC2 uto Scaling 使用时，警报应包括如何处理丢失的数据点。如果指标在设计上经常缺少数据点，则在这些期间警报的状态为 INSUFFICIENT_DATA。发生这种情况时，在找到新的数据点之前，Amazon A EC2 uto Scaling 无法替换实例。要强制警报保持之前的 ALARM 或 OK 状态，您可以改为选择忽略缺少的数据。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的配置警报如何处理丢失的数据。

Console

使用自动回滚启动实例刷新（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。

2. 选中 Auto Scaling 组旁边的复选框。
3. 在 Instance refresh（实例刷新）选项卡上的 Active instance refresh（活跃实例刷新）中，选择 Start instance refresh（开启实例刷新）。
4. 按照 启动实例刷新（控制台） 程序进行操作并根据需要配置实例刷新设置。
5. （可选）在 “刷新设置” 下，对于CloudWatch CloudWatch 警报，选择启用警报，然后选择一个或多个警报以识别任何问题，并在警报进入ALARM状态时操作失败。
6. 在 Rollback 设置下，选择启用自动回滚以将失败的实例刷新自动回滚到您在启动实例刷新之前保存在自动扩缩组中的配置。
7. 查看您的选择，然后选择启动实例刷新。

Amazon CLI

使用自动回滚启动实例刷新（Amazon CLI）

使用start-instance-refresh命令并在中`true`为`AutoRollback`选项指定`Preferences`。

以下示例说明如何启动实例刷新，如果出现故障，该刷新将自动回滚。将 *italicized* 参数值替换为您自己的值。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json` 的内容。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1"
       }
    },
    "Preferences": {
      "AutoRollback": true
    }
}
```

或者，要在实例刷新失败或指定 CloudWatch 警报处于ALARM状态时自动回滚，请在中指定`AlarmSpecification`选项`Preferences`并提供警报名称，如下例所示。将 *italicized* 参数值替换为您自己的值。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1"
       }
    },
    "Preferences": {
      "AutoRollback": true,
      "AlarmSpecification": { "Alarms": [ "my-alarm" ] }
    }
}
```

如果成功，该命令将返回类似于以下内容的输出。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

使用实例刷新和跳过匹配

跳过匹配会让 Amazon A EC2 uto Scaling 忽略已经有最新更新的实例。这样，您就无需替换超过所需数量的实例。当您希望确保自动扩缩组使用特定版本的启动模板，并且只替换那些使用其他版本的实例时，这是很有用的。

在跳过匹配时，请注意以下事项：

- 如果您在启动实例刷新时既跳过匹配又有所需的配置，Amazon A EC2 uto Scaling 会检查是否有任何实例与您的所需配置匹配。然后，它只会替换与所需配置不匹配的实例。实例刷新成功后，Amazon A EC2 uto Scaling 会更新群组以反映您所需的配置。
- 如果您使用跳过匹配开始实例刷新，但没有指定所需的配置，Amazon A EC2 uto Scaling 会检查是否有任何实例与您上次保存在 Auto Scaling 组中的配置相匹配。然后，它只会替换与您上次保存的配置不匹配的实例。

- 您可以将跳过匹配与新的启动模板、启动模板的新版本或一组实例类型配合使用。如果您启用了跳过匹配，但这些内容都没有更改，则实例刷新将立即成功，而无需替换任何实例。如果您对所需配置进行了任何其他更改（例如更改竞价分配策略），Amazon A EC2 uto Scaling 会等待实例刷新成功。然后，它会更新自动扩缩组设置以反映新的所需配置。
- 您不能将跳过匹配与新的启动配置配合使用。
- 当您开始实例刷新并提供所需的配置时，Amazon A EC2 uto Scaling 可确保所有实例都使用所需的配置。因此，当您指定 `$Default` 或 `$Latest` 作为启动模板的所需版本，然后在实例刷新过程中创建启动模板的新版本时，任何已替换的实例都将再次被替换。
- 跳过匹配不知道启动模板中的用户数据脚本是否会提取更新的代码并将其安装在新实例上。因此，跳过匹配可能会跳过替换已安装过时代码的实例。在这种情况下，您应该关闭跳过匹配功能，以确保即使没有启动模板版本更新，所有实例也都能收到最新代码。

本节包含 Amazon CLI 有关在启用跳过匹配的情况下开始实例刷新的说明。有关如何使用控制台的说明，请参阅启动实例刷新（控制台）。

跳过匹配（基本程序）

按照本节中的步骤使用 Amazon CLI 来执行以下操作：

- 创建您希望应用于实例的启动模板。
- 启动实例刷新，将启动模板应用到自动扩缩组。如果您不启用跳过匹配，则所有实例都将被替换。即使用于预置实例的启动模板与您为所需配置指定的启动模板相同，也是如此。

将跳过匹配与新的启动模板配合使用

1. 使用create-launch-template命令为您的 Auto Scaling 组创建新的启动模板。包括定义为您的自动扩缩组创建的实例详细信息的 `--launch-template-data` 选项和 JSON 输入。

   例如，使用以下命令创建包含 AMI ID *ami-0123456789abcdef0* 和 *t2.micro* 实例类型的基本启动模板。

   ```
   aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
    --version-description version1 \
     --launch-template-data
    '{"ImageId":"ami-0123456789abcdef0","InstanceType":"t2.micro"}'
   ```

   如果成功，该命令将返回类似于以下内容的输出。

```
{
   "LaunchTemplate": {
     "LaunchTemplateId": "lt-068f72b729example",
     "LaunchTemplateName": "my-template-for-auto-scaling",
     "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
     "CreateTime": "2023-01-30T18:16:06.000Z",
     "DefaultVersionNumber": 1,
     "LatestVersionNumber": 1
  }
}
```

有关更多信息，请参阅 使用创建和管理启动模板的示例 Amazon CLI。

2. 使用start-instance-refresh命令启动实例替换工作流程，并应用带有 ID 的新启动模板*lt-068f72b729example*。因为启动模板是新的，所以它只有一个版本。这意味着启动模板的版本 1 是本次实例刷新的目标。如果在实例刷新期间发生扩展事件，并且 Amazon A EC2 uto Scaling 使用此启动模板1的版本配置了新实例，则这些实例将不会被替换。成功完成操作后，新的启动模板即成功应用到自动扩缩组。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateId": "lt-068f72b729example",
          "Version": "$Default"
       }
    },
    "Preferences": {
      "SkipMatching": true
    }
}
```

如果成功，该命令将返回类似于以下内容的输出。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
```

```
}
```

跳过匹配（混合实例组）

如果您的 Auto Scaling 组采用混合实例策略，请按照本节中的步骤使用跳过匹配开始实例刷新。Amazon CLI 您有以下选项：

- 提供新的启动模板，以应用于在策略中指定的所有实例类型。
- 提供一组更新的实例类型，无论是否更改策略中的启动模板。例如，您可能想要从不需要的实例类型迁移。您可以按原样使用启动模板，无需更改 AMI、安全组或要替换实例的其他具体信息。

请按照以下部分之一的步骤操作，具体取决于哪个选项适合您的需求。

将跳过匹配与新的启动模板配合使用

1. 使用create-launch-template命令为您的 Auto Scaling 组创建新的启动模板。包括定义为您的自动扩缩组创建的实例详细信息的 `--launch-template-data` 选项和 JSON 输入。

   例如，使用以下命令创建包含 AMI ID *ami-0123456789abcdef0* 的启动模板。

   ```
   aws ec2 create-launch-template --launch-template-name my-new-template --version-
   description version1 \
     --launch-template-data '{"ImageId":"ami-0123456789abcdef0"}'
   ```

   如果成功，该命令将返回类似于以下内容的输出。

   ```
   {
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-04d5cc9b88example",
        "LaunchTemplateName": "my-new-template",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2023-01-31T15:56:02.000Z",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
     }
   }
   ```

   有关更多信息，请参阅 使用创建和管理启动模板的示例 Amazon CLI。

2. 要查看您的 Auto Scaling 组的现有混合实例策略，请运行describe-auto-scaling-groups命令。在下一步中，当您开始刷新实例时，您将需要这些信息。

   以下示例命令返回为名为 *my-asg* 的自动扩缩组配置的混合实例策略。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

   如果成功，该命令将返回类似于以下内容的输出。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg",
      "AutoScalingGroupARN":"arn",
      "MixedInstancesPolicy":{
        "LaunchTemplate":{
          "LaunchTemplateSpecification":{
            "LaunchTemplateId":"lt-073693ed27example",
            "LaunchTemplateName":"my-old-template",
            "Version":"$Default"
          },
          "Overrides":[
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
```

```
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

3. 使用start-instance-refresh命令启动实例替换工作流程，并应用带有 ID 的新启动模板*lt-04d5cc9b88example*。因为启动模板是新的，所以它只有一个版本。这意味着启动模板的版本 1 是本次实例刷新的目标。如果在实例刷新期间发生扩展事件，并且 Amazon A EC2 uto Scaling 使用此启动模板1的版本配置了新实例，则这些实例将不会被替换。成功完成操作后，更新的混合实例策略将成功应用到自动扩缩组。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容。

```
{
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-04d5cc9b88example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          },
          {
            "InstanceType":"m5a.large"
          }
        ]
```

```
      },
      "InstancesDistribution":{
        "OnDemandAllocationStrategy":"prioritized",
        "OnDemandBaseCapacity":1,
        "OnDemandPercentageAboveBaseCapacity":50,
        "SpotAllocationStrategy":"price-capacity-optimized"
        }
      }
    }
  },
  "Preferences":{
    "SkipMatching":true
  }
}
```

如果成功，该命令将返回类似于以下内容的输出。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

在下一个步骤中，您无需更改启动模板即可提供一组更新的实例类型。

将跳过匹配与一组更新的实例类型配合使用

1. 要查看您的 Auto Scaling 组的现有混合实例策略，请运行describe-auto-scaling-groups命令。在下一步中，当您开始刷新实例时，您将需要这些信息。

   以下示例命令返回为名为 *my-asg* 的自动扩缩组配置的混合实例策略。

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

   如果成功，该命令将返回类似于以下内容的输出。

   ```
   {
     "AutoScalingGroups":[
       {
         "AutoScalingGroupName":"my-asg",
         "AutoScalingGroupARN":"arn",
         "MixedInstancesPolicy":{
           "LaunchTemplate":{
   ```

```
          "LaunchTemplateSpecification":{
            "LaunchTemplateId":"lt-073693ed27example",
            "LaunchTemplateName":"my-template-for-auto-scaling",
            "Version":"$Default"
          },
          "Overrides":[
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

2. 使用start-instance-refresh命令启动实例替换工作流程并应用您的更新。如果您希望替换使用特定实例类型的实例，您所需的配置必须指定仅含所需实例类型的混合实例策略。您可以选择是否在这些实例类型上添加新的实例类型。

   以下示例命令在没有不需要的实例类型 *m5a.large* 的情况下启动实例刷新。当组中的实例类型与其余三个实例类型之一不匹配时，实例将被替换。（请注意，实例刷新不会选择要从中预置新实例的实例类型；而是由分配策略来选择。）成功完成操作后，更新的混合实例策略将成功应用到自动扩缩组。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容

```
{
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-073693ed27example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          }
        ]
      },
      "InstancesDistribution":{
        "OnDemandAllocationStrategy":"prioritized",
        "OnDemandBaseCapacity":1,
        "OnDemandPercentageAboveBaseCapacity":50,
        "SpotAllocationStrategy":"price-capacity-optimized"
        }
      }
    }
  },
  "Preferences":{
    "SkipMatching":true
  }
}
```

将检查点添加到实例刷新

在使用实例刷新时，您可以选择分阶段替换实例，以便您可以随时对实例执行验证。要执行分阶段替换，请添加检查点，这些检查点是实例刷新暂停时的时间点。使用检查点使您能够更好地控制选择更新 Auto Scaling 组的方式。它可以帮助您确认您的应用程序将以可靠、可预测的方式运行。

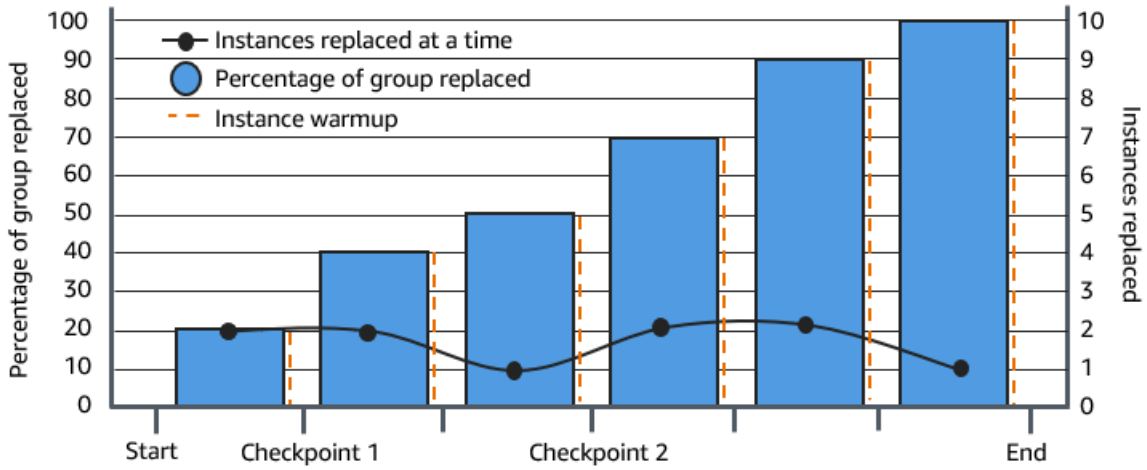
内容

工作方式

在启动实例刷新时，您可以将检查点指定为自动扩缩组中实例总数的百分比。这些检查点表示在认为已到达检查点之前，自动扩缩组中必须是新实例的实例数最小百分比。例如，如果您的检查点是 [20, 50, 100]，则在 20% 的实例为新实例时到达第一个检查点；50% 的实例为新实例时到达第二个检查点；当所有实例都是新实例时，则到达最后一个检查点。

Amazon A EC2 uto Scaling 会调整实例替换时间，以遵守指定的检查点百分比，同时保持组的最低健康百分比。为了达到检查点百分比，Amazon A EC2 uto Scaling 有时会替换更少但永远不会超过最低健康百分比允许的值。

以下自动扩缩组为例，该组有 10 个实例。检查点百分比为 [20,50,100]，最低运行正常百分比为 80%，最高运行正常百分比为 100%。为了保持最低运行正常百分比，一次仅可替换两个实例。下图总结了在到达检查点之前替换实例的过程。



在上面的示例中，每个启动的新实例都有一个实例预热期。您可能还拥有生命周期挂钩，该挂钩将实例置于等待状态，然后在实例启动或终止时执行自定义操作。

Amazon A EC2 uto Scaling 会为除百分之百完成的检查点之外的每个检查点发出事件。您可以添加一条 EventBridge 规则，将事件发送到目标，例如 Amazon SNS。这样，当您可以运行所需的验证时，系统就会通知您。有关更多信息，请参阅 为实例刷新事件创建 EventBridge 规则。

注意事项

在使用检查点时，请记住以下注意事项：

- 由于检查点基于百分比，因此要替换的实例数将随组的大小而变化。当发生横向扩展活动并且组的大小增加时，正在进行的操作可能会再次达到检查点。如果发生这种情况，Amazon A EC2 uto Scaling 会再次发送通知，并在检查点之间重复等待时间，然后再继续。
- 在某些情况下可以跳过检查点。例如，假设您的 Auto Scaling 组有两个实例，并且您的检查点百分比为 `[10,40,100]`。替换第一个实例后，Amazon A EC2 uto Scaling 计算出该组的 50% 已被替换。由于 50% 高于前两个检查点，因此它将跳过第一个检查点 (`10`) 并发送第二个检查点 (`40`) 的通知。
- 取消操作将停止进行任何进一步替换。如果您取消操作或在达到最后一个检查点之前操作失败，则任何已替换的实例都不会回滚到其以前的配置。
- 对于部分刷新，当您重新运行操作时，Amazon A EC2 uto Scaling 不会从最后一个检查点处重新启动，也不会在仅替换较早的实例时停止。但是，它将首先针对旧实例进行替换，然后再针对新实例。
- 当检查点的百分比相对于组中的实例数量而言过低时，实际完成百分比可能会高于该检查点的百分比。例如，假设检查点的百分比为 20%，并且该组有四个实例。如果 Amazon A EC2 uto Scaling 替换了四个实例中的一个，则实际替换的百分比（25%）将高于检查点的百分比（20%）。
- 达到检查点后，显示的总体完成百分比直到实例完成预热后才会更新。例如，您的检查点百分比为 `[20,50]`，其中具有 15 分钟的检查点延迟和 80 % 的最小运行正常百分比。您的自动扩缩组有 10 个实例，并进行以下替换：
  - 0:00：将两个旧实例替换为新实例。
  - 0:10：两个新实例完成预热。
  - 0:25：将两个旧实例替换为新实例。（为了保持最低运行正常百分比，仅替换两个实例。）
  - 0:35：两个新实例完成预热。
  - 0:35：将一个旧实例替换为新实例。
  - 0:45：一个新实例完成预热。

在 0:35 时，操作将停止启动新实例。完成百分比尚不能准确反映已完成替换的数量 (50%)，因为新实例未完成预热。在新实例于 0:45 完成其预热期后，完成百分比将显示 50%。

使用 Amazon Web Services Management Console 或启用检查点 Amazon CLI

您可以使用 Amazon Web Services Management Console 或 Amazon CLI 来启用检查点。

启用检查点（控制台）

您可以在开启实例刷新之前启用检查点，以使用增量或分阶段方法替换实例。这将为验证提供额外时间。

启动使用检查点的实例刷新

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   将在 Auto Scaling group（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Instance refresh（实例刷新）选项卡上的 Active instance refresh（活跃实例刷新）中，选择 Start instance refresh（开启实例刷新）。
4. 在 Start instance refresh（开启实例刷新）页面上，输入 Minimum healthy percentage（最低运行正常百分比）和 Instance warmup（实例预热）的值。
5. 选择 Enable checkpoints（启用检查点）复选框。

   此时将显示一个框，您可以在其中定义第一个检查点的百分比阈值。
6. 对于 Proceed until ____ % of the group is refreshed（继续直到刷新该组的 ____ %）中，输入一个数字 (1–100)。这将设置第一个检查点的百分比。
7. 要添加另一个检查点，请选择添加检查点，然后定义下一个检查点的百分比。
8. 要指定 Amazon A EC2 uto Scaling 在到达检查点后等待多长时间，请更新检查点**1hour**之间等待中的字段。时间单位可以是小时、分钟或秒。
9. 如果您已完成实例刷新选择，请选择启动实例刷新。

启用检查点 (Amazon CLI)

要使用启用检查点启动实例刷新 Amazon CLI，您需要一个定义以下参数的配置文件：

- CheckpointPercentages：指定要替换的实例百分比的阈值。这些阈值提供检查点。当替换和预热的实例百分比达到指定阈值之一时，操作将等待指定的时间段。您可指定在 CheckpointDelay 中等待的秒数。当指定的时间段过后，实例刷新将继续进行直到达到下一个检查点（如果适用）。
- CheckpointDelay：指定在到达检查点之后与继续操作之前需要等待的时间量（以秒为单位）。选择提供足够时间以执行验证的时间段。

CheckpointPercentages 数组中显示的最后一个值描述需要成功替换的 Auto Scaling 组的百分比。在成功替换此百分比，并且每个实例都被视为已完成初始化后，该操作将转换到 Successful。

创建多个检查点

要创建多个检查点，请使用以下示例start-instance-refresh命令。此示例配置了一个实例刷新，它最初刷新 Auto Scaling 组的 1%。在等待 10 分钟后，它会随后刷新接下来的 19%，然后再等待 10 分钟。最后，它会刷新该组的其余部分，然后结束操作。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20,100],
      "CheckpointDelay": 600
    }
}
```

创建单个检查点

要创建单个检查点，请使用以下示例start-instance-refresh命令。此示例配置了一个实例刷新，它最初刷新 Auto Scaling 组的 20%。在等待 10 分钟后，它会随后刷新该组的其余部分，然后结束操作。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
```

```
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [20,100],
      "CheckpointDelay": 600
    }
}
```

部分刷新 Auto Scaling 组

要仅替换 Auto Scaling 组的一部分然后完全停止，请使用以下示例start-instance-refresh命令。此示例配置了一个实例刷新，它最初刷新 Auto Scaling 组的 1%。在等待 10 分钟后，它会随后刷新接下来的 19%，然后结束操作。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20],
      "CheckpointDelay": 600
    }
}
```

基于最大实例生命周期替换 Auto Scaling 实例

最大实例生命周期指定实例在终止和替换之前可提供服务的最长时间（以秒为单位）。由于内部安全策略或外部合规性控制，常见使用案例可能需要按计划替换您的实例。

您必须指定至少 86,400 秒（一天）的值。要清除以前设置的值，请指定新值 0。此设置适用于 Auto Scaling 组中的所有当前和未来实例。

内容

注意事项

以下是使用此功能时的注意事项：

- 每当替换旧实例并启动新实例时，新实例都会使用当前与自动扩缩组关联的启动模板或启动配置。如果启动模板或启动配置指定了其他版本应用程序的亚马逊机器映像（AMI）ID，则将自动部署此版本的应用程序。
- 将最大实例生命周期设置过低可能会导致实例的替换速度超出预期。Amazon A EC2 uto Scaling 通常会一次替换一个实例，在替换之间会有暂停时间。但是，如果指定的最大实例寿命不能提供足够的时间来单独替换每个实例，那么 Amazon A EC2 uto Scaling 必须一次替换多个实例。可能会一次替换多个实例，最多可达 Auto Scaling 组当前容量的 10%。为避免一次替换过多实例，请设置更长的最大实例生命周期，或者使用实例横向缩减保护来暂时防止单个实例被终止。有关更多信息，请参阅使用实例横向缩减保护以控制实例终止。
- 默认情况下，Amazon A EC2 uto Scaling 会创建一个新的扩展活动来终止该实例，然后将其终止。在实例终止期间，另一个扩缩活动将会启动一个新实例。您可以使用实例维护策略将此行为更改为在终止前启动。有关更多信息，请参阅 实例维护策略。

设置最大实例生命周期

在控制台中创建 Auto Scaling 组时，您将无法设置最大实例生命周期。但在创建该组后，您就可以对其进行编辑，以设置最大实例生命周期。

为组设置最大实例生命周期（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格，其中显示有关您选择的组的信息。
3. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
4. 对于 Maximum instance lifetime (最大实例生命周期)，输入实例可使用的最长秒数。
5. 选择更新。

在 Activity（活动）选项卡上的 Activity history（活动历史记录）下，您可以通过其整个历史记录查看该组中实例的替换情况。

为组设置最大实例生命周期 (Amazon CLI)

您还可以使用 Amazon CLI 为新的或现有 Auto Scaling 组设置最长实例生命周期。

对于新的 Auto Scaling 组，请使用create-auto-scaling-group命令。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

以下示例 config.json 文件显示的最长实例生命周期为 2592000 秒（30 天）。

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
    },
    "MinSize": 1,
    "MaxSize": 5,
    "MaxInstanceLifetime": 2592000,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

对于现有的 Auto Scaling 组，请使用update-auto-scaling-group命令。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --
max-instance-lifetime 2592000
```

验证 Auto Scaling 组的最大实例生命周期

使用 describe-auto-scaling-groups 命令。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

限制

- 不能保证每个实例的最大生命周期都是准确的：不能保证仅在实例最大持续时间结束时替换实例。在某些情况下，Amazon A EC2 uto Scaling 可能需要在更新最长实例生命周期参数后立即开始替换实例。此行为的原因是避免同时替换所有实例。

- 支持@@ 实例缩容保护：Amaz on A EC2 uto Scaling 提供实例缩减保护，以帮助您控制可以终止哪些实例。在实例上启用此保护后，即使该实例已达到其最大实例生命周期，Amazon A EC2 uto Scaling 也不会终止该实例。
- 实例在启动前终止：当 Auto Scaling 组中只有一个实例时，最大实例生命周期功能可能会导致中断，因为 Amazon A EC2 uto Scaling 会终止一个实例，然后默认启动一个新实例。要将此行为更改为在终止前启动，请参阅 实例维护策略。

通过扩缩来增加或减少应用程序的计算容量

扩展是增加或减少应用程序的计算容量的能力。扩展从事件或扩展操作开始，它指示 Auto Scaling 组启动或终止 Amazon EC2 实例。

Amazon A EC2 uto Scaling 提供了多种调整扩展的方法，以最好地满足您的应用程序需求。因此，您需要很好地了解您的应用程序，这十分重要。请注意以下事项：

- Amazon A EC2 uto Scaling 应该在您的应用程序架构中扮演什么角色？ 通常可以将自动扩展视为一种增加和减少容量的主要方法，但自动扩展在保持稳定数量的服务器方面也是非常有用的。
- 哪些成本约束对您比较重要？ 由于 Amazon A EC2 uto Scaling 使用 EC2 实例，因此您只需为使用的资源付费。了解成本约束可以帮助您确定何时扩展应用程序以及扩展量。
- 哪些指标对您的应用程序比较重要？ 亚马逊 CloudWatch 支持多种不同的指标，您可以在 Auto Scaling 群组中使用这些指标。

内容

选择您的扩缩方法

Amazon A EC2 uto Scaling 为您提供了多种扩展 Auto Scaling 群组的方法。

保持固定数量的实例

自动扩缩组的默认设置是没有任何附加的扩缩策略或计划操作，这使其保持固定大小。创建自动扩缩组后，组先启动足够的实例以满足其最低容量。如果没有附加扩缩条件，该组将继续保持所需容量，即使实例运行状况不佳时也是如此。Amazon A EC2 uto Scaling 会监控您的 Auto Scaling 组中每个实例的

运行状况。如果发现实例运行状况不佳，则将其替换为新实例。您可以在 自动扩缩组中实例的运行状况检查 中阅读有关此过程的更深入的描述。

手动缩放

手动扩展是扩展您的自动扩缩组最基本的方法。您可以更新自动扩缩组的所需容量，或终止自动扩缩组中的实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的手动扩展。

按计划扩展

按计划扩缩意味着扩缩操作作为日期和时间的函数自动执行。这在您确切地知道何时增加或减少组中的实例数量时非常有用，因为该需求遵循可预测的计划。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。

根据需求动态扩缩

使用动态扩缩是一种更高级的资源扩缩方法，您可以定义扩缩策略，以动态调整 Auto Scaling 组的大小以满足需求的变化。例如，假设您有一个当前在两个实例上运行的 Web 应用程序，并希望在应用程序负载变化时将 Auto Scaling 组的 CPU 使用率保持在 50% 左右。当您不知道流量何时会发生变化时，此方法对于在流量发生变化时进行扩缩非常有用。您可以通过配置扩缩策略来进行响应。您可以使用多种策略类型（或其组合）以根据流量变化进行横向缩减。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。

主动扩缩

您还可以将预测性扩展和动态扩展（分别为主动和被动方法）相结合，以更快地扩展 EC2 容量。在流量出现每日和每周模式之前，使用预测性扩展来增加 Auto Scaling 组中的 EC2 实例数量。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的预测性扩展。

为自动扩缩组设置扩缩限制

扩缩限制表示所需自动扩缩组的最小组大小和最大组大小。您可以单独设置最小和最大大小的限制。

组的所需容量可以调整为位于最小和最大大小限制范围内的数值。所需容量必须大于等于组的最小大小，小于等于组的最大大小。

- 所需容量：表示自动扩缩组在创建时的初始容量。自动扩缩组会尝试保持所需容量。它首先按照为所需容量指定的数量启动实例，如果自动扩缩组没有附加任扩缩策略或计划操作，则将维持这一实例数量。
- 最小容量：表示最小组大小。设定扩缩策略后，它们无法将组所需的容量降至最小容量以下。
- 最大容量：表示最大组大小。设定扩缩策略后，它们无法将组所需的容量升至最大容量以上。

最小和最大大小限制也适用于以下场景：

- 当您通过更新所需容量来手动扩展自动扩缩组时。
- 会更新所需容量的计划操作运行时。如果计划操作运行时没有为组指定新的最小和最大大小限制，则会使用该组当前的最小和最大大小限制。

自动扩缩组始终会尝试保持维持所需容量。如果某个实例意外终止（例如，由于竞价型实例中断、运行状况检查失败或人工操作），该组会自动启动一个新实例以维持所需容量。

要在控制台中管理这些设置

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格的 Auto Scaling 下，选择 自动扩缩组。
3. 在 Auto Scaling groups（自动扩缩组）页面中，选择您的自动扩缩组旁的复选框。

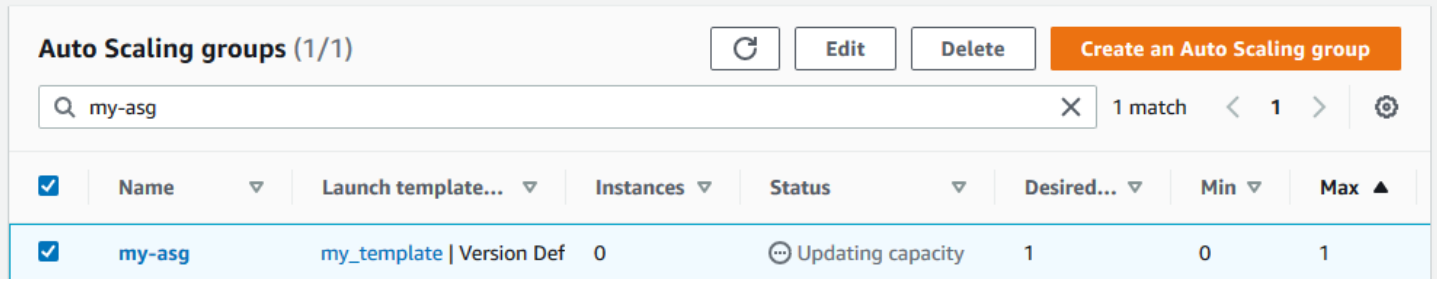
   这时将在页面底部打开一个拆分窗格。
4. 在详细信息选项卡的下窗格，查看或更改组所需的容量、最小容量和最大容量的当前设置。有关更多信息，请参阅 更改现有自动扩缩组的所需容量。

在 详细信息窗格上方，您可以找到自动扩缩组中的当前实例数、最小容量、最大容量和所需容量以及状态列等信息。如果自动扩缩组使用实例权重，则还可以找到贡献到所需容量的容量单位数信息。

要在列表中添加或删除列，请选择页面顶部的设置图标。然后，对于 Auto Scaling groups attributes（自动扩缩组属性），打开或关闭每个列，然后选择 Confirm（确认）。

更改后验证 Auto Scaling 组的大小

实例列显示当前正在运行的实例数。启动或终止实例时，状态列显示正在更新容量状态，如下图所示。



请等待几分钟，然后刷新视图以查看最新状态。某个扩缩活动完成后，Instances（实例）列将会显示更新后的值。

您可以在 Instance management（实例管理）选项卡的 Instances（实例）下查看当前正在运行的实例数量和状态。

为 Auto Scaling 组设置原定设置实例预热

CloudWatch 在 Auto Scaling 实例中收集和聚合使用情况数据，例如 CPU 和网络 I/O。您可以使用这些指标来创建扩缩策略，以随着所选指标值的增减调整 Auto Scaling 组中的实例数量。

您可以指定实例在达到 `InService` 状态后等待多长时间才能向聚合指标提供使用数据。这一指定的时间称为默认实例预热。此项可防止动态扩缩受到尚未处理应用程序流量，并且可能暂时存在计算资源使用率较高的单个实例的指标影响。

为了优化目标跟踪扩缩策略和步进扩缩策略的性能，强烈建议您启用并配置默认实例预热功能。默认情况下，将不会启用或配置此项。

启用默认实例预热时，请记住，如果自动扩缩组设置为使用实例维护策略，或者使用实例刷新来替换实例，则可以防止实例在完成初始化之前计入最小运行正常百分比。

内容

扩缩性能注意事项

对于大多数应用程序来说，拥有一个适用于所有功能的默认实例预热时间（而不是不同功能使用不同的预热时间），是非常有用的。例如，如果您未设置默认实例预热，则实例刷新功能将使用运行状况检查宽限期作为默认预热时间。如果您有任何目标跟踪扩缩策略和步进扩缩策略，这些策略会使用为默认冷却时间设置的值作为默认预热时间。如果您有任何预测性扩展策略，则这些策略没有默认预热时间。

当实例正在预热时，仅当未预热实例的指标值大于策略的警报阈值上限（或目标跟踪扩缩策略的目标利用率）时，您的动态扩缩策略才会横向扩展。如果需求下降，横向缩减将变得更加保守，以保护应用程序的可用性。这样可以阻止动态扩缩的横向缩减活动，直到新实例完成预热。

在扩展时，Amazon A EC2 uto Scaling 在决定向该组添加多少实例时，会将正在预热的实例视为组容量的一部分。因此，需要添加相似容量的多个警报违规会导致一次扩缩活动。旨在持续横向扩展，但不会过度扩缩。

如果未启用默认实例预热，则实例在向其发送指标 CloudWatch 并将其计入当前容量之前等待的时间将因实例而异。因此，与正在发生的实际工作负载相比，可能无法预测您的扩缩策略的执行情况。

例如，假设一个具有重复 on-and-off工作负载模式的应用程序。预测性扩缩策略用于对是否增加实例数量做出反复决策。由于预测性扩展策略没有默认预热时间，因此这些实例会立即开始为聚合指标做出贡献。如果这些实例在启动时资源使用量较高，那么添加实例可能会导致聚合指标出现峰值。这可能会影响使用这些指标的任何动态扩缩策略，具体取决于使用量需要多长时间才能稳定下来。如果突破了动态扩缩策略的警告阈值上限，则该组的大小会再次增加。在新实例预热期间，横向缩减活动将被阻止。

选择默认的实例预热时间

设置原定设置实例预热的关键是确定实例需要多长时间才能完成初始化，以及资源消耗在达到 `InService` 状态后需要多长时间才能稳定下来。选择实例预热时间时，尝试优化平衡采集合法流量的使用数据与尽量减少与启动时的临时使用峰值相关的数据采集。

假设您有一个附加到 Elastic Load Balancing 负载均衡器的自动扩缩组。当新实例完成启动后，它们将在进入 `InService` 状态之前注册到负载均衡器。在实例进入 `InService` 状态之后，资源消耗仍然可能会经历暂时的高峰，然后才会逐渐稳定下来。例如，与无需下载大型资产的轻量级 Web 服务器相比，对于必须下载并缓存大型资产的应用程序服务器，其资源消耗将需要更长的时间才能稳定。实例预热提供了稳定资源消耗所需的时间延迟。

> **Important**
>
> 如果您不确定需要多长时间预热，则可以从 300 秒开始。然后逐渐减少或增加该时间，直到您的应用程序获得最佳扩缩性能为止。可能需要执行几次此操作才能得出正确设置。或者，如果您的任何扩缩策略有自己的预热时间（`EstimatedInstanceWarmup`），则可以使用此值开始。有关更多信息，请参阅 查找具有先前设置实例预热时间的扩缩策略。

对于需要在启动时运行配置任务或脚本的使用案例，应考虑使用生命周期挂钩。生命周期挂钩可以将实例投入使用的时间延迟到实例完成初始化之后。如果引导启动脚本需要一段时间才能完成，则生命周期钩子将特别有用。如果您添加了生命周期挂钩，则可以降低原定设置实例预热的值。有关使用生命周期钩子的更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

为组启用原定设置实例预热

您可以在创建 Auto Scaling 组时启用原定设置实例预热。也可以为现有的组启用此功能。

通过启用默认实例预热功能，您不必再为以下功能指定预热参数值：

- 实例刷新
- 目标跟踪扩缩
- 分步扩缩

Console

为新的组启用原定设置实例预热（控制台）

创建 Auto Scaling 组时，在 Configure advanced options（配置高级选项）页面的 Additional settings（其他设置）下，选择 Enable default instance warmup（启用原定设置实例预热）选项。选择应用程序所需的预热时间。

Amazon CLI

为新的组启用原定设置实例预热（Amazon CLI）

要为 Auto Scaling 组启用原定设置实例预热，请添加 `--default-instance-warmup` 选项并指定一个介于 0 到 3600 之间的值（以秒为单位）。启用此功能后，将值设为 `-1` 将会关闭此设置。

以下create-auto-scaling-group命令使用名称创建一个 Auto Scaling 组，*my-asg*并启用值为*120*秒的默认实例预热。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120 ...
```

> **Tip**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

Console

为现有的组启用原定设置实例预热（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
5. 对于默认实例预热，选择应用程序所需的预热时间。
6. 选择更新。

Amazon CLI

为现有的组启用原定设置实例预热（Amazon CLI）

以下示例使用update-auto-scaling-group命令为名*my-asg*为的现有 Auto Scaling 组启用默认实例预热，其值为*120*秒。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120
```

> **Tip**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

验证组的默认实例预热时间

使用 Amazon CLI按照以下程序验证自动扩缩组的默认实例预热时间。

验证自动扩缩组的默认实例预热时间

使用以下 describe-auto-scaling-groups 命令。*my-asg*替换为您的 Auto Scaling 组的名称。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下为响应示例。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "DefaultInstanceWarmup": 120
        }
    ]
}
```

查找具有先前设置实例预热时间的扩缩策略

要确定您的策略是否有为 EstimatedInstanceWarmup 设置的自己的预热时间，请使用 Amazon CLI运行以下 describe-policies 命令。*my-asg*替换为您的 Auto Scaling 组的名称。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?EstimatedInstanceWarmup!=`null`]'
```

下面是示例输出。

```
[
  {
    "AutoScalingGroupName":"my-asg",
    "PolicyName":"cpu50-target-tracking-scaling-policy",
    "PolicyARN":"arn",
    "PolicyType":"TargetTrackingScaling",
    "StepAdjustments":[],
    "EstimatedInstanceWarmup":120,
    "Alarms":[{
        "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-
asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
      },
      {
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }],
    "TargetTrackingConfiguration":{
      "PredefinedMetricSpecification":{
        "PredefinedMetricType":"ASGAverageCPUUtilization"
      },
      "TargetValue":50.0,
      "DisableScaleIn":false
    },
    "Enabled":true
  },

    ... additional policies ...

]
```

清除先前为扩缩策略设置的实例预热

启用默认实例预热后，更新仍有自己的预热时间的所有扩缩策略，以清除先前设置的值。否则，它将覆盖原定设置实例预热。

您可以使用控制台 Amazon CLI、或更新扩展策略 Amazon SDKs。本节介绍控制台的操作步骤。如果您使用 Amazon CLI 或 Amazon SDKs，请确保保留现有的策略配置，但要移除该`EstimatedInstanceWarmup`属性。更新现有扩展策略时，该策略将替换为您以编程方式调用PutScalingPolicy时指定的策略。不保留原始值。

清除先前为扩缩策略设置的实例预热（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在自动扩缩选项卡的动态扩缩策略中，选择您感兴趣的策略，然后依次选择操作、编辑。
4. 对于实例预热，请清除实例预热值，改用默认实例预热值。
5. 选择更新。

Amazon A EC2 uto Scaling 的手动扩展

您可以随时手动调整 Auto Scaling 组中的 EC2 实例数量。手动更改实例数的这一过程称为手动扩缩。手动扩缩是自动扩缩的替代方案，尤其是在您想要进行一次性容量更改时。

在您手动扩展群组后，Amazon A EC2 uto Scaling 会根据您定义的扩展策略和计划操作恢复正常的自动扩展活动。对于启用了默认实例预热的组，任何新实例都要经过一段预热时间，然后才会开始贡献用于自动扩缩的指标。此预热时间有助于稳定组的新容量。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

有时，您可能需要在手动扩缩组之前暂时禁用扩缩策略和计划操作。这样可以防止手动扩缩操作和自动扩缩活动之间产生冲突。有关更多信息，请参阅 关闭扩缩活动。

内容

更改现有自动扩缩组的所需容量

当您更改 Auto Scaling 组的所需容量时，Amazon A EC2 uto Scaling 会管理启动和终止实例以达到新的所需容量的过程。

Console

要更改您自动扩缩组的大小

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部显示一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Group details（组详细信息）、Edit (编辑)。
4. 对于所需容量，增加或减少所需容量。例如，要将组的大小加一，如果当前值为 1，则输入 2。

   如果所需容量的新值大于所需的最小容量和所需的最大容量，则所需的最大容量将自动增加到新的所需容量值。
5. 完成后，选择 Update（更新）。

验证您指定的组大小是否导致启动相同数量的实例。例如，如果您将组的大小加一，则请验证您的自动扩缩组是否已另外启动一个实例。

要验证 Auto Scaling 组的容量是否已更改

1. 在活动选项卡的活动历史记录中，您可以查看与自动扩缩组关联的活动的进度。Status（状态）列显示您实例的当前状态。当您的实例启动时，状态列将显示 `Not yet in service`。该实例启动后，状态会变为 `Successful`。您还可以使用刷新图标来查看实例的当前状态。有关更多信息，请参阅 验证 Auto Scaling 组的扩缩活动。
2. 在实例管理选项卡的实例中，您可以查看实例的状态。启动实例只需很短的时间。
   - Lifecycle（生命周期）列显示您的实例的状态。最初，您的实例处于 `Pending` 状态。在实例准备好接收流量时，其状态为 `InService`。
   - 运行状况列显示对您的实例进行 Amazon A EC2 uto Scaling 运行状况检查的结果。

Amazon CLI

以下示例假设您创建了一个最小大小为 1，最大大小为 5 的 Auto Scaling 组。因此，该组目前正在运行的实例是 1 个。

要更改 Auto Scaling 组的大小

使用set-desired-capacity命令更改 Auto Scaling 组的大小，如以下示例所示。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2
```

如果您选择遵守 Auto Scaling 组的默认冷却时间，必须指定 `--honor-cooldown` 选项，如下面的示例所示。有关更多信息，请参阅 扩展 Amazon A EC2 uto Scaling 的冷却时间。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2 --honor-cooldown
```

验证 Auto Scaling 组的大小

使用describe-auto-scaling-groups命令确认 Auto Scaling 组的大小是否已更改，如以下示例所示。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下是示例输出，其提供关于组和已启动实例的详细信息。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 300,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
```

```
                    },
                    "InstanceId": "i-0c20ac468fa3049e8",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            "CreatedTime": "2019-03-18T23:30:42.611Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-c87f2be0",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn",
            "TrafficSources": []
        }
    ]
}
```

注意 DesiredCapacity 显示了新值。Auto Scaling 组已启动另一个实例。

终止您自动扩缩组中的实例（Amazon CLI）

有时，您可能想要在自动扩缩组中手动进行横向缩减，但同时又想终止指定的实例。您可以使用-scaling-group 命令在 Auto terminate-instance-in-autoScaling-group 中手动缩放，并指定要终止的实例的 ID 和--should-decrement-desired-capacity选项，如以下示例所示。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-026e4c9f62c3e448c --should-decrement-desired-capacity
```

以下是示例输出，其提供有关扩缩活动的详细信息。

```
{
    "Activities": [
        {
            "ActivityId": "b8d62b03-10d8-9df4-7377-e464ab6bd0cb",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-026e4c9f62c3e448c",
```

```
            "Cause": "At 2023-09-23T06:39:59Z instance i-026e4c9f62c3e448c was taken
 out of service in response to a user request, shrinking the capacity from 1 to 0.",
            "StartTime": "2023-09-23T06:39:59.015000+00:00",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\"us-
west-2c\"}"
        }
    ]
}
```

此选项在控制台中不可用。但是，您可以使用亚马逊 EC2 控制台的实例页面来终止您的 Auto Scaling 组中的实例。当您执行此操作时，Amazon A EC2 uto Scaling 会检测到该实例已停止运行，并在运行状况检查过程中自动替换该实例。在终止该实例后需要一两分钟的时间才能启动新实例。有关如何终止实例的信息，请参阅 Amazon EC2 用户指南中的终止实例。

如果您终止组中的实例，从而导致可用区之间的分布不均衡，Amazon A EC2 uto Scaling 会重新平衡该组以重新建立均衡分配，除非您暂停该AZRebalance流程。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。

Amazon A EC2 uto Scaling 的计划扩展

借助计划扩缩，您可以根据可预测的负载变化来设置应用程序的自动扩缩。您可以创建计划操作，在特定时间增加或减少组的所需容量。

例如，您每周遇到规律的流量模式，即负载在一周的中间增加，而在接近周末时会下降。您可以在 Amazon A EC2 uto Scaling 中配置符合以下模式的扩展计划：

- 周三上午，一项计划操作通过增加先前设置的自动扩缩组所需容量来增加容量。
- 周五晚上，另一项计划操作通过降低先前设置的自动扩缩组所需容量来减少容量。

利用这些计划的扩缩操作，您可以优化成本和性能。您的应用程序有足够的容量来处理一周中间的流量高峰，但不会在其他时间过度预置不需要的容量。

您可以同时使用计划扩缩和扩缩策略，以获得两种扩缩方法的优势。运行计划的扩缩操作后，扩缩策略可以继续决定是否进一步扩缩容量。这有助于确保您有足够的容量来处理应用程序的负载。当您的应用程序扩展以满足需求时，当前容量必须在计划操作设置的最小容量和最大容量范围内。

内容

计划扩缩的工作原理

要使用计划扩展，请创建计划操作，告知 Amazon A EC2 uto Scaling 在特定时间执行扩展活动。创建计划操作时，请指定自动扩缩组、应进行扩缩活动的时间以及可选的新最小容量和新最大容量。您可以创建仅扩展一次或按重复计划扩展的计划操作。

在指定时间，Amazon A EC2 uto Scaling 会根据新的容量值进行扩展，方法是将当前容量与指定的所需容量进行比较。

- 如果当前容量小于指定的所需容量，Amazon A EC2 uto Scaling 将扩展或添加实例到指定的所需容量。
- 如果当前容量大于指定的所需容量，Amazon A EC2 uto Scaling 会缩减或移除实例到指定的所需容量。

计划操作会设置指定日期和时间组的所需、最小和最大容量。一次只能为其中一种容量（例如，所需容量）创建计划操作。但是，在某些情况下，您必须包括最小和最大容量，以确保您在操作中指定的所需容量不会超出这些限制。

定期安排

要使用 Amazon CLI 或 SDK 创建定期计划，请指定 cron 表达式和时区来描述该计划操作何时重演。您可以选择指定开始时间和/或结束时间的日期和时间。

要使用创建重复计划 Amazon Web Services Management Console，请指定计划操作的重复模式、时区、开始时间和可选的结束时间。所有重复模式选项都基于 cron 表达式。或者，您可以编写您自己的 Cron 表达式。

受支持的 cron 表达式格式由用空格分隔的五个字段组成：[Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]。例如，Cron 表达式 `30 6 * * 2` 配置每周二的早上 6:30 再执行的计划操作。星号用作通配符，以匹配字段的所有值。有关 cron 表达式的其他示例，请参见https://crontab.guru/examples.html。有关以此格式编写您自己的 cron 表达式的信息，请参阅 Crontab。

仔细选择您的开始时间和结束时间。记住以下内容：

- 如果您指定开始时间，Amazon A EC2 uto Scaling 将在此时执行操作，然后根据指定的周期执行操作。
- 如果指定结束时间，则操作在此时间之后停止重复。在到达计划操作的结束时间后，它不会保留在您的账户中。
- 如果重复时间与结束时间完全匹配，则 Amazon A EC2 uto Scaling 将不会在结束时间执行预定的操作。
- 使用 Amazon CLI 或 SDK 时，必须以 UTC 格式设置开始时间和结束时间。

时区

预设情况下，您设置的重复计划采用协调世界时 (UTC)。您可以更改时间以符合本地时区或您的网络中其他部分的时区。如果您指定的时区遵守 Daylight Saving Time (DST)，它会自动调整 DST。

有效值是互联网编号分配机构（IANA）时区数据库中时区的规范名称。例如，美国东部时间通常标识为 `America/New_York`。有关更多信息，请参阅 https://www.iana.org/time-zones。

基于位置的时区（如 `America/New_York`）会根据 DST 自动调整。但是，基于 UTC 的时区（如 `Etc/UTC` 是绝对时间，不会针对 DST 进行调整。

例如对于您有一个定期计划，其时区为 `America/New_York`。第一个扩缩操作发生在 `America/New_York` DST 开始之前的时区。下一个扩缩操作发生在 `America/New_York` DST 开始之后的时区。第一个动作从当地时间 UTC-5 上午 8 点开始，而第二个时间从当地时间 UTC-4 凌晨 8 点开始。

如果您使用创建计划操作 Amazon Web Services Management Console 并指定遵守 DST 的时区，则循环计划以及开始和结束时间都会自动调整 DST。

注意事项

创建计划的操作时，请记住以下内容：

- 可以保证同组内计划操作的执行顺序正确，但不保证跨组的计划操作的执行顺序正确。

- 计划操作的运行时间一般为几秒钟。然而，该操作可能会比计划的开始时间延迟最多两分钟。因为系统将按照计划操作的顺序来执行 Auto Scaling 组内的操作，所以计划开始时间彼此接近的计划操作可能需要更长时间才能执行。
- 您可以暂时关闭 Auto Scaling 组的计划扩展，方法是暂停 `ScheduledActions` 过程。这有助于防止计划操作处于活动状态，而无需将其删除。然后，当您想要再次使用时，您可以恢复计划的扩展。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 创建计划操作后，您可以更新除名称之外的任何设置。

限制

- 每个 Auto Scaling 组，计划操作的名称必须是唯一的。
- 计划的操作必须具有唯一时间值。如果您尝试计划在已计划另一个扩展活动的时间进行活动，则该调用将被拒绝，并返回一个错误消息，指示已存在已存在已计划启动时间的计划操作。
- 您最多可以为每个 Auto Scaling 组创建 125 个计划的操作。

创建计划的操作

要为自动扩缩组创建计划操作，请使用以下方法之一：

Console

创建计划的操作

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Automatic scaling（自动扩展）选项卡上的 Scheduled actions（计划操作）中，选择 Create scheduled action（创建计划操作）。
4. 为计划操作输入名称。
5. 适用于所需容量、最小值、最大值中，选择新的所需组容量以及新的最小和最大大小限制。所需容量必须大于等于组的最小大小，小于等于组的最大大小。
6. 对于 Recurrence（循环），请选择下列可用选项之一。

   - 如果您想按定期计划进行扩展，请选择 Amazon A EC2 uto Scaling 运行计划操作的频率。

- 如果您选择以 Every 开头的选项，则将为您创建 Cron 表达式。
- 如果您选择 Cron，请输入 Cron 表达式，此表达式指定了执行操作的时间。
- 如果只想缩放一次，请选择 Once（一次）。

7. 对于时区，请选择时区。默认为 `Etc/UTC`。

   列出的所有时区均来自 IANA 时区数据库。欲了解更多信息，请参阅 https://en.wikipedia。org/wiki/List_of_tz_database_time_zones。

8. 定义日期和时间特定开始时间。

   - 如果您选择了循环计划，则开始时间将定义循环系列中第一个计划操作的运行时间。
   - 如果您选择了 Once 作为重复，则开始时间定义运行计划操作的日期和时间。

9. （可选）对于循环计划，您可以通过选择设置 End Time，然后选择一个日期和时间 End Time。
10. 选择创建。控制台将显示 Auto Scaling 组的计划操作。

Amazon CLI

要创建计划的操作，可以使用以下示例命令之一。将每个 *user input placeholder* 替换为您自己的信息。

示例：仅扩展一次

使用以下带有`--start-time "YYYY-MM-DDThh:mm:ssZ"`和`--desired-capacity`选项的 put-scheduled-update-group-action 命令。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
  --auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

示例：根据周期性计划来计划扩缩

使用以下带有`--recurrence "cron expression"`和`--desired-capacity`选项的 put-scheduled-update-group-action 命令。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
```

```
  --auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

默认情况下，Amazon A EC2 uto Scaling 会根据 UTC 时区运行指定的重复计划。要指定不同的时区，请包含 `--time-zone` 选项和 IANA 时区的名称，如下例所示。

```
--time-zone "America/New_York"
```

欲了解更多信息，请参阅 https://en.wikipedia。 org/wiki/List_of_tz_database_time_zones。

查看计划操作详细信息

要查看自动扩缩组即将执行的计划操作的详细信息，请使用以下方法之一：

Console

查看计划操作详细信息

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择您的 Auto Scaling 组。
3. 在自动扩缩选项卡的计划操作中，您可以查看即将执行的计划操作。

请注意，控制会以您的当地时间显示开始时间和结束时间值，并采用指定日期和时间生效的 UTC 偏移。UTC 偏移量是本地时间与 UTC 之间的差值，以小时和分钟为单位。时区的值显示您请求的时区，例如 `America/New_York`。

Amazon CLI

使用以下 describe-scheduled-actions 命令。

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

如果成功，该命令返回类似以下内容的输出。

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
```

```
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

验证扩缩活动

要验证与计划扩缩相关联的扩缩活动，请参阅 验证 Auto Scaling 组的扩缩活动。

删除计划的操作

要删除计划操作，请使用以下方法之一：

Console

删除计划操作

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择您的 Auto Scaling 组。
3. 在 Automatic scaling（自动扩展）选项卡上的 Scheduled actions（计划操作）中，选择计划操作。
4. 依次选择 Actions（操作）和 Delete（删除）。
5. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。

Amazon CLI

使用以下 delete-scheduled-action 命令。

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \
  --scheduled-action-name my-recurring-action
```

Amazon A EC2 uto Scaling 的动态扩展

动态扩缩会根据流量的变化扩展自动扩缩组的容量。

Amazon A EC2 uto Scaling 支持以下类型的动态扩展策略：

- 目标跟踪扩展-根据 Amazon CloudWatch 指标和目标值增加和减少群组的当前容量。这与温控器保持家里温度的方式类似——您只需选择一个温度，剩余的工作将由温控器完成。
- Step scaling（步进分步）– 通过一系列扩缩调整（也称步进调整）来增加和减少组的当前容量，具体调整因警报严重程度而异。
- Simple scaling（简单扩缩）– 通过单次扩缩调整来增加和减少组的当前容量，每次扩缩活动之间有一个冷却时间。

强烈建议您使用目标跟踪扩缩策略，并选择一个与自动扩缩组容量变化成反比的指标。因此，如果将自动扩缩组的大小增加一倍，则该指标将降低 50%。这使指标数据能够准确触发按比例扩缩事件。其中包括平均 CPU 利用率或每个目标的平均请求数等指标。

使用目标跟踪时，自动扩缩组会根据应用程序的实际负载按比例扩缩。这意味着，除根据负载变化满足当前容量需求外，目标跟踪策略还可以适应持续的负载变化，例如由于季节性变化而导致的负载变化。

目标跟踪策略还无需手动定义 CloudWatch 警报和缩放调整。Amazon A EC2 uto Scaling 会根据你设定的目标自动处理这个问题。

内容

动态扩缩策略的工作方式

动态扩展策略指示 Amazon A EC2 uto Scaling 跟踪特定 CloudWatch 指标，并定义当相关 CloudWatch 警报处于警报状态时要采取的操作。用于调用警报状态的指标是来自自动扩缩组中所有实例的指标聚合。（例如，假设您有一个 Auto Scaling 组，其中有两个实例，一个实例的 CPU 利用率为 60%，另一个实例的 CPU 利用率为 40%。CPU 平均利用率为 50%。）策略生效后，当警报阈值被突破时，Amazon A EC2 uto Scaling 会向上或向下调整组的所需容量。

调用动态扩展策略时，如果容量计算得出的数字超出了组的最小和最大大小范围，Amazon A EC2 uto Scaling 将确保新容量永远不会超出最小和最大大小限制。容量通过以下两种方式衡量：使用您在以实例数设置所需容量时选择的相同单位；或者使用容量单位（如果应用了实例权重）。

- 示例 1：Auto Scaling 组的最大容量为 3，当前容量为 2，并有增加 3 个实例的动态扩缩策略。在调用此策略时，Amazon A EC2 uto Scaling 仅向该组添加 1 个实例，以防止该组超过其最大大小。
- 示例 2：Auto Scaling 组的最小容量为 2，当前容量为 3，并有移除 2 个实例的动态扩缩策略。在调用此策略时，Amazon A EC2 uto Scaling 仅从该组中移除 1 个实例，以防止该组小于其最小大小。

当所需容量达到最大大小限制时，向外扩展停止。如果需求下降而容量减少，Amazon A EC2 uto Scaling 可以再次进行扩展。

但使用实例权重时例外。在这种情况下，Amazon A EC2 uto Scaling 可以扩展到超过最大大小限制，但只能扩展到您的最大实例重量。其目的是尽可能接近新的所需容量，但仍然遵循为该组指定的分配战略。分配策略决定要启动的实例类型。权重根据实例类型确定每个实例向所需的组容量贡献的容量单位数。

- 示例 3：Auto Scaling 组的最大容量为 12，当前容量为 10，并有增加 5 个容量单位的动态扩缩策略。向实例类型分配三个权重之一：1、4 或 6。在调用该策略时，Amazon A EC2 uto Scaling 会根据分配策略选择启动权重为 6 的实例类型。此横向扩展事件的结果是得到所需容量为 12、当前容量为 16 的组。

多个动态扩缩策略

在大多数情况下，目标跟踪扩缩策略就足以将您的 Auto Scaling 组配置为自动扩展和缩减。目标跟踪扩缩策略允许您选择所需结果，并让 Auto Scaling 组根据需要添加和删除实例以实现该结果。

对于高级扩缩配置，您的 Auto Scaling 组可以有多个扩缩策略。例如，您可以定义一个或多个目标跟踪扩展策略，一个或多个步进扩展策略，或者同时使用两种策略。这样可以更灵活地覆盖多种场景。

为了说明多个动态扩展策略是如何协同工作的，可以考虑一个使用 Auto Scaling 组和 Amazon SQS 队列向单个 EC2 实例发送请求的应用程序。为了帮助确保应用程序性能达到最佳级别，有两个策略用于控制何时扩缩 Auto Scaling 组。一个是使用自定义指标、根据队列中的 SQS 消息数增加和移除容量的目标跟踪策略。另一种是分步扩展策略，当实例在指定时间长 CloudWatch `CPUUtilization`度内超过 90% 的使用率时，该策略使用 Amazon 指标来增加容量。

如果同时实施多个策略，各个策略可能会同时指示 Auto Scaling 组扩展（或缩减）。例如，在 SQS 自定义`CPUUtilization`指标达到峰值并突破自定义指标 CloudWatch 警报阈值的同时，指标可能会达到峰值并突破警报的阈值。

发生这些情况时，Amazon A EC2 uto Scaling 会选择为横向扩展和向内扩展提供最大容量的策略。例如，假定 `CPUUtilization` 策略启动一个实例，而 SQS 队列的策略启动两个实例。如果同时满足两个策略的扩展标准，Amazon A EC2 uto Scaling 会优先使用 SQS 队列策略。这会导致 Auto Scaling 组启动两个实例。

即使策略使用不同的扩展条件，使提供最大容量的策略具有优先级的方法也适用。例如，如果一个策略终止了三个实例，另一个策略将实例数量减少了 25%，并且该组在缩小规模时有八个实例，那么 Amazon A EC2 uto Scaling 会优先考虑为该组提供最大实例数量的策略。这会导致 Auto Scaling 组终止两个实例（8 X 25% = 2）。其目的是防止 Amazon A EC2 uto Scaling 删除过多的实例。

不过，在将目标跟踪扩展策略与步进扩展策略结合使用时，我们建议您务必谨慎，因为这些策略之间的冲突可能会导致意外的行为。例如，如果步进扩缩策略在目标跟踪策略准备执行横向缩减之前启动横向缩减活动，则不会阻止横向缩减活动。在横向缩减活动完成后，目标跟踪策略可能会指示组重新横向扩展。

Amazon A EC2 uto Scaling 的目标跟踪扩展政策

目标跟踪扩缩策略根据目标指标值自动扩缩自动扩缩组的容量。它会自动适应您各个应用程序的独特使用模式。这使您的应用程序无需手动干预即可保持 EC2 实例的最佳性能和高利用率，从而提高成本效益。

通过目标跟踪，您可以选择一个指标和一个目标值，目标值用来表示应用程序的理想平均利用率或吞吐量水平。Amazon A EC2 uto Scaling 创建并管理在指标偏离目标时调用扩展事件的 CloudWatch 警报。举例来说，这与恒温器保持目标温度的方式类似。

例如，假设您当前有一个在两个实例上运行的 Web 应用程序，并希望在应用程序负载变化时将自动扩缩组的 CPU 利用率保持在 50% 左右。这为您提供额外容量以处理流量高峰，而无需维护过多的空闲资源。

创建一个将目标平均 CPU 利用率设置为 50% 的目标跟踪扩缩策略即可满足此需求。然后，当 CPU 使用率超过 50% 时，自动扩缩组将横向扩展或增加容量，以处理增加的负载。当 CPU 利用率降至 50% 以下时，该组将横向缩减或减少容量，以便在利用率低的时期优化成本。

主题

多个目标跟踪扩缩策略

为帮助优化扩缩性能，您可以将多个目标跟踪扩缩策略组合使用，但前提是其中的每个策略都各自使用不同的指标。例如，利用率和吞吐量可能会相互影响。每当其中一个指标发生变化时，通常意味着其他指标也将受到影响。因此，使用多个指标可以提供有关自动扩缩组所承担负载的额外信息。这可以帮助 Amazon A EC2 uto Scaling 在确定向您的群组添加多少容量时做出更明智的决定。

Amazon A EC2 uto Scaling 的目的是始终优先考虑可用性。如果任何目标跟踪策略已准备好横向扩展，则它将横向扩展自动扩缩组。仅在所有目标跟踪策略（已启用横向缩减部分）都准备好横向缩减时，才会进行横向缩减。

选择指标

您可以使用预定义的指标或自定义指标，创建目标跟踪扩展策略。预定义指标可让您更轻松地访问最常用的扩展指标。自定义指标允许您根据其他可用 CloudWatch 指标进行扩展，包括以更精细的间隔发布的高分辨率指标，这些指标在几秒钟左右的时间间隔内发布。您可以发布自己的高分辨率指标或其他 Amazon 服务发布的指标。

有关使用高分辨率指标创建目标跟踪策略的更多信息，请参阅使用高分辨率指标创建目标跟踪策略以加快响应速度。

目标跟踪支持以下预定义指标：

- `ASGAverageCPUUtilization`—Auto Scaling 组的平均 CPU 利用率。
- `ASGAverageNetworkIn`—Auto Scaling 组在所有网络接口上收到的平均字节数。
- `ASGAverageNetworkOut`—Auto Scaling 组在所有网络接口上发送的平均字节数。
- `ALBRequestCountPerTarget` — Auto Scaling 组的每个目标的平均 Application Load Balancer 请求计数。

> **Important**
>
> 有关 CPU 利用率、网络 I/O 和每个目标的 Application Load Balancer 请求数等 CloudWatch 指标的其他重要信息，分别可在亚马逊 EC2 用户指南中的列出您的实例的可用CloudWatch 指标主题和应用程序负载均衡器用户指南中的应用程序负载均衡器指标主题中找到。

您可以 CloudWatch 通过指定自定义 CloudWatch 指标来选择其他可用指标或您自己的指标。有关使用为目标跟踪扩展策略指定自定义指标规范的示例 Amazon CLI，请参阅Amazon CLI的扩缩策略示例。

选择指标时请记住原则：

- 我们建议您仅使用间隔为一分钟或更短的指标，以帮助您更快地扩展以应对利用率的变化。以较低间隔发布的指标允许目标跟踪策略检测并更快地响应 Auto Scaling 组利用率的变化。
- 如果您选择由 Amazon 发布的预定义指标 EC2，例如 CPU 利用率，我们建议您启用详细监控。默认情况下，所有 Amazon EC2 指标均以五分钟为间隔发布，但通过启用详细监控，可以将其配置为较低的一分钟间隔。有关如何启用详细监控的信息，请参阅配置 Auto Scaling 实例的监控。
- 并非所有自定义指标都适用于目标跟踪。指标必须是有效的使用率指标，它用于描述实例的繁忙程度。指标值必须随着 Auto Scaling 组中的实例数按比例增加或减少。这样，指标数据可用于随实例数量按比例扩展或缩减。例如，如果 Auto Scaling 组的负载分布在各个实例上，则 Auto Scaling 组的 CPU 利用率（即`CPUUtilization`带有 EC2 指标维度的亚马逊指标`AutoScalingGroupName`）起作用。
- 以下指标不适用于目标跟踪：
  - Auto Scaling 组前面的负载均衡器收到的请求数（即，Elastic Load Balancing 指标 `RequestCount`）。负载均衡器收到的请求数不会根据 Auto Scaling 组的使用率而发生变化。
  - 负载均衡器请求延迟（即，Elastic Load Balancing 指标 `Latency`）。请求延迟可能会根据使用率增加而增加，但不一定按比例变化。
  - CloudWatch 亚马逊 SQS 队列指标。`ApproximateNumberOfMessagesVisible`队列中的消息数可能不会随着处理队列中的消息的 Auto Scaling 组的大小按比例发生变化。但是，衡量 Auto

Scaling 组中每个 EC2 实例队列中消息数量的自定义指标可以起作用。有关更多信息，请参阅 基于 Amazon SQS 的扩缩策略。

- 要使用 ALBRequestCountPerTarget 指标，您必须指定 ResourceLabel 参数以标识与该指标关联的负载均衡器目标组。有关使用为目标跟踪扩展策略指定ResourceLabel参数的示例 Amazon CLI，请参阅Amazon CLI的扩缩策略示例。
- 当某个指标向 CloudWatch （例如ALBRequestCountPerTarget）发出实数 0 值时，当您的应用程序持续一段时间内没有流量时，Auto Scaling 组可以缩减到 0。要在没有请求路由时将自动扩缩组横向缩减至 0，组的最小容量必须设置为 0。
- 您可以使用指标数学组合现有指标，而不必发布要在扩缩策略中使用的新指标。有关更多信息，请参阅 使用指标数学创建目标跟踪扩缩策略。

定义目标值

创建目标跟踪扩缩策略时，必须指定一个目标值。目标值表示自动扩缩组的最佳平均利用率或吞吐量。为了经济高效地使用资源，目标值的设置应尽可能高，并为流量的意外增加提供合的理缓冲。当应用程序针对正常流量进行最佳横向扩展时，实际指标值应等于或略低于目标值。

当扩缩策略基于吞吐量（例如，每目标的应用程序负载均衡器请求计数、网络 I/O 或其他计数指标）时，目标值表示单个实例在一分钟内的最佳平均吞吐量。

定义实例预热时间

您可以指定新启动实例的预热时间（单位为秒）。在指定的预热时间到期之前，实例不会计入 Auto Scaling 组的聚合 EC2 实例指标。

当实例处于预热时间时，仅当未预热实例的指标值大于策略的目标利用率时，您的扩缩策略才会横向扩展。

如果组再次横向扩展，则仍在预热的实例将计算为下一横向扩展活动所需容量的一部分。旨在持续 (但不过度) 扩大。

当横向扩展活动正在进行时，通过扩缩策略启动的所有横向缩减活动都将被阻止，直到实例完成预热。当实例完成预热后，如果发生横向缩减事件，那么在计算新的所需容量时，当前正在终止的任何实例都将计入该组的当前容量。因此不会从 Auto Scaling 组中删除更多实例。

默认值

如果未设置任何值，则扩缩策略将使用默认值，即为该组定义的默认实例预热值。如果默认实例预热为空，则将回退到默认冷却时间值。建议使用默认实例预热，以便在预热时间发生变化时更轻松地更新所有扩缩策略。

注意事项

使用目标跟踪扩缩策略时，需要注意以下事项：

- 请勿创建、编辑或删除与目标跟踪扩展策略一起使用的 CloudWatch 警报。Amazon A EC2 uto Scaling 会创建和管理与您的目标跟踪扩展策略关联的 CloudWatch警报，并且可以在必要时对其进行编辑、替换或删除，以便为您的应用程序及其不断变化的使用模式自定义扩展体验。
- 在流量波动时，目标跟踪扩缩策略会优先考虑可用性，在流量减少时以更缓步的方式横向缩减。如果您想获得更大的控制权，则步进缩放策略可能是更好的选择。您可以暂时禁用目标跟踪策略的缩减部分。这有助于保持最少的实例数量，以便成功部署。
- 如果指标缺少数据点，则会导致 CloudWatch 警报状态更改为`INSUFFICIENT_DATA`。发生这种情况时，在找到新的数据点之前，Amazon A EC2 uto Scaling 无法扩展您的群组。
- 如果设计为很少报告指标，则指标数学可能会很有帮助。例如，要使用最新的值，则使用`FILL(m1,REPEAT)` 函数，其中 `m1` 是指标。
- 您可能会看到目标值与实际指标数据点之间存在差距。这是因为我们在确定需要添加或删除多少个实例时通过向上或向下取整来保守地进行操作，以免添加的实例数量不足或删除的实例数量过多。但是，对于实例较少的 Auto Scaling 组，组的使用率可能偏离目标值较远。例如，假设您将 CPU 使用率的目标值设置为 50%，然后 Auto Scaling 组超过了该目标。我们可以确定，添加 1.5 个实例会将 CPU 使用率降低到接近 50%。由于不可能添加 1.5 个实例，我们将该值向上取整，添加两个实例。这可能会将 CPU 使用率降至 50% 以下，但可确保应用程序具有充足的支持资源。同样，如果我们确定删除 1.5 个实例可使 CPU 使用率提高到 50% 以上，我们将只删除一个实例。

  对于包含更多实例的更大 Auto Scaling 组，使用率分布在更多实例上，在这种情况下，添加或删除实例会导致目标值与实际指标数据点之间的差距缩小。
- 目标跟踪扩缩策略假设它应该在指定指标高于目标值时扩展 Auto Scaling 组。在指定指标低于目标值时，不能使用目标跟踪扩缩策略来横向扩展自动扩缩组。

创建目标跟踪扩缩策略

要为自动扩缩组创建目标跟踪扩缩策略，请使用以下方法之一。

在开始之前，请确认您的首选指标每隔 1 分钟可用（相比之下，Amazon EC2 指标的默认间隔为 5 分钟）。

Console

为新的自动扩缩组创建目标跟踪扩展策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择 Create Auto Scaling group（创建 Auto Scaling 组）。
3. 在步骤 1、2 和 3 中，根据需要选择选项，然后继续步骤 4：配置组大小和扩缩策略。
4. 在扩缩下，通过更新最小容量和最大容量来指定您想要在其间扩缩的范围。这两个设置允许您的 Auto Scaling 组动态扩缩。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
5. 在自动扩缩下，选择目标跟踪扩缩策略。
6. 要定义策略，请执行以下操作：
   a. 指定策略的名称。
   b. 对于 Metric type（指标类型），选择一个指标。

      如果您选择了每个目标的 Application Load Balancer 请求计数，请在目标组中选择目标组。
   c. 为指标指定 Target value（目标值）。
   d. （可选）对于实例预热，请根据需要更新实例预热值。
   e. （可选）选择 Disable scale in to create only a scale-out policy（禁用缩减以创建仅扩展策略）。这样，可以根据需要创建独立的其他类型的缩减策略。
7. 继续创建 Auto Scaling 组。您的扩展策略将在创建 Auto Scaling 组后创建。

为现有 Auto Scaling 组创建目标跟踪扩展策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 验证是否正确设置了扩缩限制。例如，如果您的组所需的容量已经是最大，则指定一个新的最大值才能向外扩展。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
4. 在 Automatic scaling（自动扩展）选项卡的 Dynamic scaling policies（动态扩展策略）中，选择 Create dynamic scaling policy（创建动态扩展策略）。

5. 要定义策略，请执行以下操作：

   a. 对于 策略类型，保留默认的 目标跟踪扩展。

   b. 指定策略的名称。

   c. 对于 Metric type（指标类型），选择一个指标。您只能选择一种指标类型。要使用多个指标，请创建多个策略。

      如果您选择了每个目标的 Application Load Balancer 请求计数，请在目标组中选择目标组。

   d. 为指标指定 Target value（目标值）。

   e. （可选）对于实例预热，请根据需要更新实例预热值。

   f. （可选）选择 Disable scale in to create only a scale-out policy（禁用缩减以创建仅扩展策略）。这样，可以根据需要创建独立的其他类型的缩减策略。

6. 选择创建。

Amazon CLI

要创建目标跟踪扩缩策略，可以使用以下示例来帮助您开始。将每个 *user input placeholder* 替换为您自己的信息。

> **Note**
>
> 有关更多示例，请参阅Amazon CLI的扩缩策略示例。

创建目标跟踪扩缩策略（Amazon CLI）

1. 使用以下 `cat` 命令可以在您主目录的名为 `config.json` 的 JSON 文件中存储扩缩策略的目标值和预定义指标规范。以下是将 CPU 平均使用率保持在 50% 的示例目标跟踪配置。

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
}
```

有关更多信息，请参阅PredefinedMetricSpecification《Amazon A EC2 uto Scaling API 参考》。

2. 使用 put-scaling-policy 命令以及在前面的步骤中创建的 config.json 文件创建扩展策略。

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

如果成功，此命令将返回代表您创建的两个 CloudWatch 警报的 ARNs 和名称。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu50-target-tracking-
scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

使用高分辨率指标创建目标跟踪策略以加快响应速度

目标跟踪支持具有秒级数据点的高分辨率 CloudWatch 指标，这些数据点的发布间隔小于一分钟。配置目标跟踪策略，通过高分辨率 CloudWatch 指标监控需求模式不稳定的应用程序的利用率，例如客户服

务 APIs、直播服务、电子商务网站和按需数据处理。为了提高容量与需求匹配的精度，目标跟踪使用这种精细的监控来更快地检测和响应不断变化的需求和 EC2 实例利用率。

有关如何以高分辨率发布指标的更多信息，请参阅 Amazon CloudWatch 用户指南中的发布自定义指标。要访问和发布 EC2 指标，例如高分辨率下的 CPU 利用率，可能需要使用CloudWatch 代理。

Amazon Web Services 区域

使用高分辨率指标的目标跟踪功能适用于 Amazon Web Services 区域 除外 Amazon GovCloud (US) Regions。

具有高分辨率指标的目标跟踪策略的工作原理

您可以通过定义要跟踪的指标和要为该指标保留的目标值来创建目标跟踪策略。要根据高分辨率指标进行缩放，请指定该指标的名称，并将目标跟踪观察该指标的指标周期设置为低于 60 秒的值。目前支持的最低间隔为 10 秒。您可以按比此更短的时间间隔发布指标。

> **Note**
>
> 不支持大于 60 的指标周期。

您可以对单个 CloudWatch 指标配置目标跟踪，也可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的单个时间序列。这两个选项都允许您定义指标周期。

示例

示例 1

以下示例基于高分辨率 CloudWatch 指标创建目标跟踪策略。该指标以 10 秒的分辨率发布。通过定义周期，您可以启用目标跟踪，以 10 秒的粒度监控该指标。将每个 *user input placeholder* 替换为您自己的信息。

```
$ cat ~/config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
      "MetricName": "MyHighResolutionMetric",
      "Namespace": "MyNamespace",
      "Dimensions": [
        {
          "Name": "MyOptionalDimensionName",
```

```
          "Value": "MyOptionalMetricDimensionValue"
        }
      ],
      "Statistic": "Average",
      "Unit": "None"
      "Period": "10
  }
}
```

示例 2

您可以使用公制数学表达式将多个指标组合成单个时间序列以进行扩展。指标数学对于将现有指标转换为每个实例的平均值特别有用。转换指标至关重要，因为目标跟踪假设该指标与 Auto Scaling 组的容量成反比。因此，当容量增加时，该指标的减少比例应几乎相同。

例如，假设您有一个指标来表示您的应用程序要处理的待处理任务。您可以使用公制数学将待处理任务除以 Auto Scaling 组的运行容量。Auto Scaling 以 1 分钟的粒度发布容量指标，因此该指标在亚分钟间隔内不会有任何值。如果您想使用更高的分辨率进行扩展，这可能会导致容量与待处理任务指标之间的周期不匹配。为避免这种不匹配，我们建议您使用 FILL 表达式用前一分钟时间戳中记录的容量编号填充缺失值。

以下示例使用公制数学将待处理任务指标除以容量。在一段时间内，我们将两个指标都设置为 10 秒。由于该指标每隔 1 分钟发布一次，因此我们在容量指标上使用 FILL 操作。

使用公制数学来修改多个指标

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Pending jobs to be processed",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "MyPendingJobsMetric",
                        "Namespace": "Custom",
                    },
                    "Stat": "Sum"
                    "Period": 10
                },
                "ReturnData": false
            },
```

```
            {
                "Label": "Get the running instance capacity (matching the period to
 that of the m1)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "GroupInService",
                        "Namespace": "AWS/AutoScaling",
                        "Dimensions": [
                            {
                                "Name": "AutoScalingGroupName",
                                "Value": "my-asg"
                            }
                        ]
                    },
                    "Stat": "Average"
                    "Period": 10
                },
                "ReturnData": false
            },
            {
                "Label": "Calculate the pending job per capacity (note the use of the
 FILL expression)",
                "Id": "e1",
                "Expression": "m1 / FILL(m2,REPEAT)",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
}
```

注意事项

使用目标跟踪和高分辨率指标时，请考虑以下几点。

- 为确保不会丢失可能导致不想要的自动缩放结果的数据点，您的 CloudWatch 指标必须以与您指定的时间段相同或更高的分辨率发布。
- 将目标值定义为要为 Auto Scaling 组保留的 per-instance-per-minute指标值。如果您使用的指标的值可以根据指标的周期相乘，则设置适当的目标值至关重要。例如，任何基于计数的指标，例如使用 SUM 统计数据的请求计数或待处理任务，都将具有不同的指标值，具体取决于所选的时段。您仍然应该假设您正在根据每分钟平均值设定目标。

- 尽管使用 Amazon A EC2 uto Scaling 不收取任何额外费用，但您必须为诸如亚马逊 EC2 实例、CloudWatch 指标和 CloudWatch 警报之类的资源付费。在前面的示例中创建的高分辨率警报的价格与标准 CloudWatch 警报的价格不同。有关 CloudWatch 定价的更多信息，请参阅 Amazon CloudWatch 定价。
- 目标跟踪要求指标代表实例的每个 EC2 实例的平均利用率。为此，您可以使用指标数学运算作为目标跟踪策略配置的一部分。将您的指标除以 Auto Scaling 组的运行容量。确保为用于创建单个时间序列的每个指标定义相同的指标周期。如果这些指标以不同的时间间隔发布，请对间隔较高的指标使用 FILL 操作来填充缺失的数据点。

使用指标数学创建目标跟踪扩缩策略

使用指标数学，您可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的时间序列。您可以在 CloudWatch 控制台中可视化生成的时间序列并将其添加到仪表板中。有关指标数学的更多信息，请参阅 Amazon CloudWatch 用户指南中的使用指标数学。

以下考虑因素适用于指标数学表达式：

- 您可以查询任何可用的 CloudWatch 指标。每个指标都是指标名称、命名空间和零个或多个维度的唯一组合。
- 您可以使用任何算术运算符 (+-*/^)、统计函数（例如 AVG 或 SUM）或其他支持的函数。CloudWatch
- 您可以在数学表达式的公式中同时使用指标和其他数学表达式的结果。
- 指标规范中使用的任何表达式最终都必须返回一个单个时间序列。
- 您可以使用 CloudWatch 控制台或 CloudWatch GetMetricDataAPI 验证指标数学表达式是否有效。

示例：每个实例的 Amazon SQS 队列积压

要计算每个实例的 Amazon SQS caling 队列积压，请获取可用于从队列中检索的消息的大致数量，然后将该数字除以自动扩缩组的运行容量，该数字即为处于 `InService` 状态的实例数量。有关更多信息，请参阅 基于 Amazon SQS 的扩缩策略。

表达式的逻辑如下：

```
sum of (number of messages in the queue)/(number of InService instances)
```

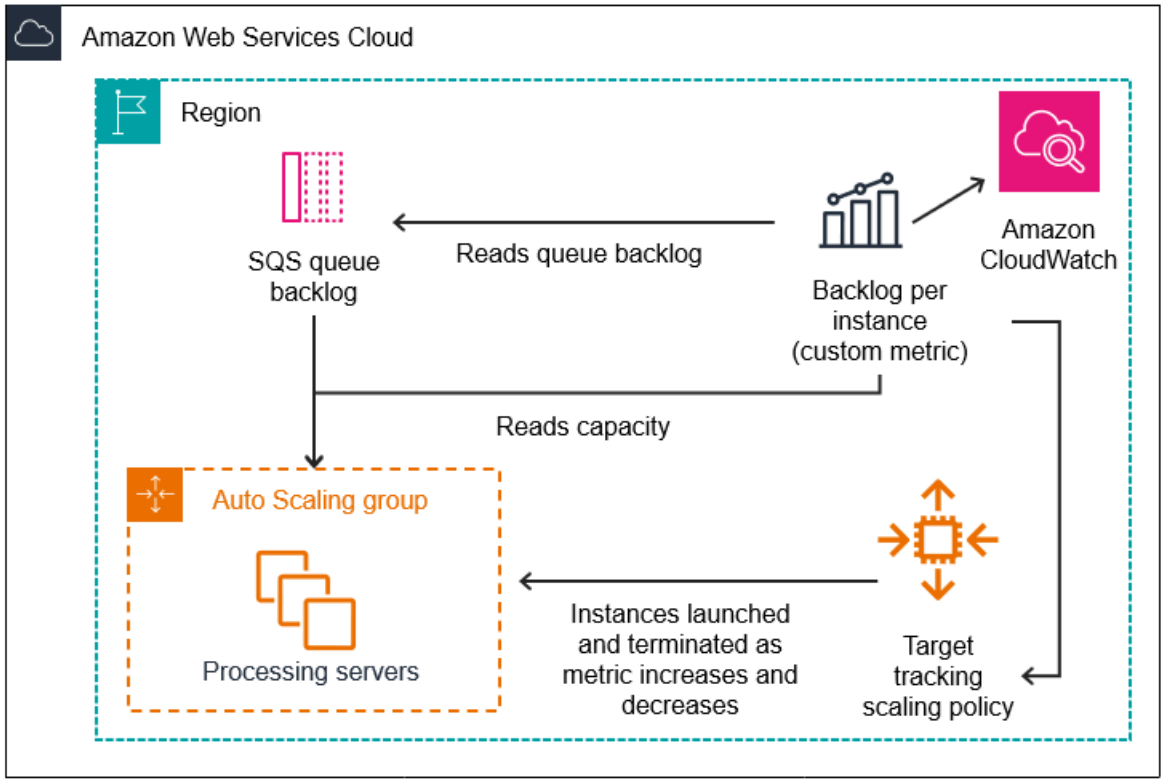
那么您的 CloudWatch 指标信息如下所示。

ID	CloudWatch 指标	Statistic	周期
m1	ApproximateNumberOfMessagesVisible	Sum	1 minute
m2	GroupInServiceInstances	平均值	1 minute

您的指标数学 ID 和表达式如下所示。

ID	Expression
e1	`(m1)/(m2)`

下图阐明了此指标的架构：



使用该指标数学来创建目标跟踪扩展策略 (Amazon CLI)

1. 将指标数学表达式作为自定义指标规范的一部分存储在名为 `config.json` 的 JSON 文件中。

使用下面的示例帮助您快速开始。将每个 *user input placeholder* 替换为您自己的信息。

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Get the queue size (the number of messages waiting to be
 processed)",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "ApproximateNumberOfMessagesVisible",
                        "Namespace": "AWS/SQS",
                        "Dimensions": [
                            {
                                "Name": "QueueName",
                                "Value": "my-queue"
                            }
                        ]
                    },
                    "Stat": "Sum"
                },
                "ReturnData": false
            },
            {
                "Label": "Get the group size (the number of InService instances)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "GroupInServiceInstances",
                        "Namespace": "AWS/AutoScaling",
                        "Dimensions": [
                            {
                                "Name": "AutoScalingGroupName",
                                "Value": "my-asg"
                            }
                        ]
                    },
                    "Stat": "Average"
                },
                "ReturnData": false
            },
            {
```

```
                "Label": "Calculate the backlog per instance",
                "Id": "e1",
                "Expression": "m1 / m2",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
}
```

有关更多信息，请参阅TargetTrackingConfiguration《Amazon A EC2 uto Scaling API 参考》。

> **Note**
>
> 以下是一些其他资源，可以帮助您查找指标名称、命名空间、维度和指标 CloudWatch统计信息：
>
> - 有关 Amazon 服务的可用指标的信息，请参阅《亚马逊 CloudWatch 用户指南》中发布 CloudWatch 指标的Amazon 服务。
> - 要使用获取指标的确切指标名称、命名空间和维度（如果适用） Amazon CLI，请参阅列表 CloudWatch 指标。

2. 要创建此策略，请使用 JSON 文件作为输入运行put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon 资源名称 (ARN) 和代表您创建 ARNs 的两个 CloudWatch 警报中的一个。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/sqs-backlog-target-
tracking-scaling-policy",
    "Alarms": [
        {
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

> **Note**
>
> 如果此命令引发错误，请确保已将 Amazon CLI 本地版本更新到最新版本。

Amazon A EC2 uto Scaling 的步骤和简单扩展策略

分步扩展和简单扩展策略可根据 CloudWatch 警报以预定义的增量扩展 Auto Scaling 组的容量。您可以定义单独的扩缩策略，以便在超过警报阈值时处理横向扩展（增加容量）和横向缩减（减少容量）。

通过步进缩放和简单扩展，您可以创建和管理调用扩展过程的 CloudWatch 警报。当警报被违反时，Amazon A EC2 uto Scaling 会启动与该警报关联的扩展策略。

强烈建议您使用目标跟踪扩缩策略，根据类似于平均 CPU 利用率或每个目标的平均请求数等指标进行扩展。使用目标跟踪，可以通过在容量增加时减少以及在容量减少时增加的指标，按比例扩展或缩减实例数。这有助于确保 Amazon A EC2 uto Scaling 密切关注您的应用程序的需求曲线。有关更多信息，请参阅 目标跟踪扩展策略。

内容

- 注意事项
- 为横向扩展创建步进扩缩策略
- 为横向缩减创建步进扩缩策略
- 简单扩展策略

分步扩缩策略的工作原理

要使用步进缩放，您需要先创建一个 CloudWatch 警报，用于监控 Auto Scaling 组的指标。定义确定触发警报的指标、阈值和评估周期数。然后，创建步进扩缩策略，定义在突破警报阈值时如何扩缩您的组。您可以将 Auto Scaling 组当前容量的百分比或容量单位用于缩放调整类型。有关更多信息，请参阅 扩展调整类型。

在策略中添加分步调整。您可以根据警报的阈值突破大小定义不同的分步调整。例如：

- 如果警报指标达到 60%，则横向扩展 10 个实例
- 如果警报指标达到 75%，则横向扩展 30 个实例
- 如果警报指标达到 85%，则横向扩展 40 个实例

在指定的评估周期内超过警报阈值时，Amazon A EC2 uto Scaling 将应用策略中定义的步骤调整。针对其他警报触发情况，可以继续进行调整，直到警报状态恢复为 0K。

每个实例都有一个预热时间，以防止扩缩活动对短时间内发生的变化反应过快。您可以选择为扩缩策略配置预热时间。不过，建议使用默认实例预热，以便在预热时间发生变化时更轻松地更新所有扩缩策略。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

简单扩缩策略与步进扩缩策略类似，不同之处在于这些策略基于单次扩缩调整，每次扩缩活动之间有一个冷却时间。有关更多信息，请参阅 简单扩展策略。

步进扩展的分步调整

在创建分步扩展策略时，您可以指定一个或多个步进调整，这些步进调整会根据警报违规的大小自动扩展实例数量。每个分步调整指定以下内容：

- 指标值的下限
- 指标值的上限
- 要扩展的数量（基于扩展调整类型）

CloudWatch 根据与 CloudWatch 警报关联的指标的统计数据聚合指标数据点。超过警报时，将调用相应的扩缩策略。Amazon A EC2 uto Scaling 将聚合类型应用于来自的最新指标数据点 CloudWatch（而不是原始指标数据）。它将此聚合指标值与步进调整定义的上限和下限进行比较，以确定执行哪个步进调整。

您可以指定相对于违例阈值的上限和下限。例如，假设您针对指标高于 50% 时发出了 CloudWatch 警报并制定了扩展策略。然后，当指标低于 50% 时，又发出了另一个警报并采取横向缩减策略。对于每个策略，您可进行一组调整类型为 `PercentChangeInCapacity`（或控制台中的组中占比）的步进调整：

示例：扩展策略的步进调整

下限	上限	调整
0	10	0
10	20	10
20	null	30

示例：缩放策略的步进调整

下限	上限	调整
-10	0	0
-20	-10	-10
null	-20	-30

这将创建以下扩展配置。

```
Metric value

-infinity          30%    40%          60%     70%             infinity
-------------------------------------------------------------------------
          -30%      | -10% | Unchanged   | +10%  |       +30%
-------------------------------------------------------------------------
```

现在，假设您在当前容量和所需容量均为 10 的自动扩缩组上使用此扩缩配置。以下几点总结了扩展配置相对于组的所需容量和当前容量的行为：

- 在聚合指标值大于 40 并小于 60 时，将保留所需容量和当前容量。
- 如果指标值达到 60，根据扩展策略的第二个分步调整 (增加 10 个实例的 10%)，组的所需容量将增加 1 个实例，总共达到 11 个实例。在新实例运行并且其指定的预热时间过期后，当前组容量将增加到 11 个实例。如果指标值即使在这一容量增加之后也上升到 70，则组的所需容量将再增加 3 个实例，达到 14 个实例。这基于扩展策略的第三个步进调整（添加 11 个实例的 30%，即 3.3 个实例，向下舍入到 3 个实例）。
- 如果指标值降到 40，根据缩减策略的第二个分步调整 (删除 14 个实例的 10%，向下取整为 1 个实例)，组的所需容量将减小 1 个实例，达到 13 个实例。如果指标值即使在这一容量减少之后也下降至 30，则组的所需容量将再减少 3 个实例，达到 10 个实例。这基于缩减策略的第三个步进调整（移除 13 个实例的 30%，即 3.9 个实例，向下舍入为 3 个实例）。

为扩展策略指定步进调整时，请注意以下事项：

- 如果使用 Amazon Web Services Management Console，则可以将上限和下限指定为绝对值。如果您使用 Amazon CLI 或 SDK，则需要指定相对于违规阈值的上限和下限。
- 分步调整范围不能重叠或有间隙。
- 只有一个分步调整可以有空下限 (负无穷)。如果一个分步调整有负下限，则必须有一个分步调整有空下限。
- 只有一个分步调整可以有空上限 (正无穷)。如果一个分步调整有正上限，则必须有一个分步调整有空上限。
- 同一分步调整中的上限和下限不能为空。
- 如果指标值高于违例阈值，则含下限而不含上限。如果指标值低于违例阈值，则不含下限而含上限。

扩展调整类型

您可以根据您选择的扩展调整类型来定义执行最佳扩展操作的扩展策略。您可以将调整类型指定为 Auto Scaling 组当前容量的百分比或以容量单位指定。通常，一个容量单位表示一个实例，除非您使用实例权重功能。

Amazon A EC2 uto Scaling 支持以下调整类型，用于步进缩放和简单扩展：

- ChangeInCapacity — 将当前组容量增加或减少指定的值。正值将增加容量，负调整值将减小容量。例如：如果组的当前容量为 3，调整值为 5，那么当执行此策略时，我们会向容量添加 5 个容量单位，共计 8 个容量单位。
- ExactCapacity — 将组的当前容量更改为指定值。为此调整类型指定一个非负值。例如：如果当前组容量为 3，调整值为 5，则当执行此策略时，我们会将容量更改为 5 个容量单位。
- PercentChangeInCapacity — 将当前组容量增加或减少指定的百分比。正值将增加容量，负值将减少容量。例如：如果当前容量为 10，调整值为 10%，那么当执行此策略时，我们会向容量添加 1 个容量单位，总共 11 个容量单位。

> **Note**
>
> 如果得出的值不是整数，将按如下所示进行取整：
>
> - 大于 1 的值向下取整。例如，12.7 取整为 12。
> - 0 和 1 之间的值舍入到 1。例如，.67 取整为 1。
> - 0 和 -1 之间的值舍入到 -1。例如，-.58 取整为 -1。
> - 小于 -1 的值向上取整。例如，-6.67 取整为 -6。

通过 PercentChangeInCapacity，您还可以使用 MinAdjustmentMagnitude 参数指定要扩展的最小实例。例如，假定您创建一个增加 25% 的策略，并且您指定最小增量为 2 个实例。如果您有一个 Auto Scaling 组包含 4 个实例，而要执行该扩展策略，则 4 的 25% 就是 1 个实例。但是，因为您指定了最小增量 2，则将添加 2 个实例。

使用实例权重时，将 MinAdjustmentMagnitude 参数设置为非零值的效果会发生变化。该值以容量单位为单位。如需设置所要扩展的最小实例数，请将此参数设置为至少与最大实例权重一样大的值。

如果您使用实例权重，请记住，自动扩缩组的当前容量可以根据需要超过所需容量。如果要递减的绝对数或百分比表示递减的数量小于当前容量和所需容量之间的差值，则不会执行任何扩展操作。在您查看超过警报阈值时扩缩策略的结果时，必须考虑这些行为。例如，假设所需容量为 30，当前容量为 32。超过警报时，如果扩缩策略将所需容量减少 1，则不会执行扩缩操作。

实例预热

对于扩缩步骤，您可以指定新启动实例的预热时间（单位为秒）。在指定的预热时间到期之前，实例不会计入 Auto Scaling 组的聚合 EC2 实例指标。

当实例处于预热时间，仅当未预热实例的指标值大于策略的警报阈值上限时，您的扩缩策略才会横向扩展。

如果组再次横向扩展，则仍在预热的实例将计算为下一横向扩展活动所需容量的一部分。因此，落入同一分步调整中的多个警报违例只会导致一个扩展活动。旨在持续 (但不过度) 扩大。

例如，假设您通过两个步骤创建策略。第一步，当指标达到 60 时，增加 10%；第二步，当指标达到 70%时，增加 30%。自动扩缩组的所需容量和当前容量为 10。在聚合指标值小于 60 时，所需容量和当前容量不变。假设指标达到 60，因此添加 1 个实例（10 个实例中的 10%）。然后，在新实例仍在预热期间，指标达到 62。扩缩策略根据当前容量（仍为 10）计算新的所需容量。不过，组的所需容量已经增加至 11 个实例，因此，扩展策略不会进一步增加所需的容量。如果指标在新实例仍在预热时增长到 70，我们应添加 3 个实例（10 个实例的 30%）。但该组的所需容量已经是 11，所以我们只添加 2 个实例，因为新的所需容量为 13 个实例。

当横向扩展活动正在进行时，通过扩缩策略启动的所有横向缩减活动都将被阻止，直到实例完成预热。当实例完成预热后，如果发生横向缩减事件，那么在计算新的所需容量时，当前正在终止的任何实例都将计入该组的当前容量。因此不会从 Auto Scaling 组中删除更多实例。例如，当实例已经终止时，如果警报超出将所需容量减 1 的相同步进调整范围，则不会执行扩缩操作。

默认值

如果未设置任何值，则扩缩策略将使用默认值，即为该组定义的默认实例预热值。如果默认实例预热为空，则将回退到默认冷却时间值。

注意事项

使用分步和简单扩缩策略时，需要注意以下事项：

- 考虑是否可以足够准确地预测应用程序上的分步调整，以便使用分步扩缩。如果您的扩缩指标的升高或降低与可扩展目标的容量成比例，则建议您使用目标跟踪扩缩策略。您仍然可以选择使用步进扩展作为附加策略来实现更高级的配置。例如，您可以在利用率达到特定级别时配置更积极的响应。
- 确保在横向扩展和横向缩减之间选择足够的余量，以防止摇摆。摆动是横向缩减和横向扩展的无限循环。也就是说，如果采取扩展操作，则指标值将更改并启动另一个相反方向的扩展操作。

为横向扩展创建步进扩缩策略

要为您的自动扩缩组创建横向扩展的步进扩缩策略，请使用以下方法之一：

Console

步骤 1：为指标高阈值创建 CloudWatch 警报

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。

2. 如果需要，可以更改区域。从导航栏中，选择您的自动扩缩组所在的区域。
3. 在导航窗格中，选择 Alarms, All alarms（警报，所有警报），然后选择 Create alarm（创建警报）。
4. 选择选择指标。
5. 在 “所有指标” 选项卡上 EC2，选择 “按 Auto Scaling 组”，然后在搜索字段中输入 Auto Scaling 组的名称。然后，选择 `CPUUtilization` 并选择 Select metric（选择指标）。将显示 Specify metric and conditions（指定指标和条件）页面，其中显示一个图表以及有关指标的其他信息。
6. 在 Period（周期）下，选择警报的评估周期，例如 1 分钟。评估警报时，每个周期都聚合到一个数据点。

   > **Note**
   >
   > 周期越短，创建的警报越敏感。

7. 在 Conditions（条件）下，执行以下操作：

   - 对于 Threshold type（阈值类型），选择 Static（静态）。
   - 对于每当 **`CPUUtilization`** 为，请指定您是希望指标值大于还是大于等于阈值以触发警报。然后，在 than（大于/小于）下，输入您希望超过警报的阈值。

8. 在其他配置下，执行以下操作：

   - 对于 Datapoints to alarm（触发警报的数据点数），输入指标值必须满足阈值条件才会触发警报的数据点（评估时间段）数。例如，2 个连续的 5 分钟时间段需要花 10 分钟才会调用警报状态。
   - 对于 Missing data treatment（缺失数据处理），选择 Treat missing data as bad (breaching threshold)（将丢失的数据视为不良数据（违反阈值））。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的配置 CloudWatch 警报如何处理丢失的数据。

9. 选择下一步。

   Configure actions（配置操作）页面会显示。

10. 在 Notification（通知）下面，选择一个在警报处于 `ALARM`、`OK` 或 `INSUFFICIENT_DATA` 状态时通知的 Amazon SNS 主题。

    要使告警为相同告警状态或不同告警状态发送多个通知，请选择添加通知。

    要让警报不发送通知，请选择删除。

11. 您可以保留 Configure actions（配置操作）页面的其他部分为空。将其他部分留空会创建警报，而不会将其与扩展策略相关联。然后，您可以将警报与 Amazon A EC2 uto Scaling 控制台中的扩展策略相关联。
12. 选择 Next（下一步）。
13. 输入警报的名称（例如，`Step-Scaling-AlarmHigh-AddCapacity`）和可选的描述，然后选择 Next（下一步）。
14. 选择创建警报。

创建 CloudWatch 警报后，按照以下步骤继续从上次中断的地方继续。

步骤 2：为横向扩展创建步进扩缩策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 验证是否正确设置了扩缩限制。例如，如果您的组所需的容量已经是最大，则指定一个新的最大值才能向外扩展。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
4. 在 Automatic scaling（自动扩展）选项卡的 Dynamic scaling policies（动态扩展策略）中，选择 Create dynamic scaling policy（创建动态扩展策略）。
5. 对于策略类型，选择步进扩展，然后指定该策略的名称。
6. 要获得CloudWatch 警报，请选择您的闹钟。如果您尚未创建警报，请选择创建警 CloudWatch 报，然后完成上一个过程中的步骤 4 到步骤 14 以创建警报。
7. 指定在使用 Take the action (执行操作) 来完成操作时，此策略对当前组大小进行的更改。您可以添加特定数量的实例或现有组大小的百分比，也可将组设置为准确的大小。

   例如，要创建将组容量增加 30% 的横向扩展策略，请选择 `Add`，在下一个字段中输入 `30`，然后选择 `percent of group`。默认情况下，此步骤调整的下限为警报阈值，上限为正 (+) 无穷。
8. 要添加另一个步骤，请选择 Add step (添加步进)，然后定义要缩放的量以及步进相对于警报阈值的下限和上限。
9. 要设置可扩展的最少实例数，请更新 Add capacity units in increments of at least (添加容量单位的增量至少为) 1 capacity units (容量单位) 中的数量字段。
10. （可选）对于实例预热，请根据需要更新实例预热值。

11. 选择创建。

Amazon CLI

要创建横向扩展（增加容量）的步进扩缩策略，可以使用以下示例命令。将每个 *user input placeholder* 替换为您自己的信息。

使用时 Amazon CLI，首先要创建分步扩展策略，该策略向 Amazon A EC2 uto Scaling 提供有关在指标值增加时如何扩展的说明。然后，通过确定要监控的指标、为警报定义指标的高阈值和其他详细信息，并将警报与扩缩策略关联，您可以创建警报。

步骤 1：为横向扩展创建策略

使用以下put-scaling-policy命令创建名为的分步扩展策略`my-step-scale-out-policy`，其调整类型为`PercentChangeInCapacity`，该策略可根据以下步骤调整来增加组的容量（假设 CloudWatch 警报阈值为 60%）：

- 当指标值大于或等于 60% 但小于 75% 时，将实例计数增加 10%
- 当指标值大于或等于 75% 但小于 85% 时，将实例计数增加 20%
- 当指标值大于或等于 85% 时，将实例计数增加 30%

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \
                     MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

记下策略的 Amazon Resource Name (ARN)。您需要它来为策略创建 CloudWatch 警报。

```
{
    "PolicyARN":
 "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-
aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy
```

```
}
```

步骤 2：为指标高阈值创建 CloudWatch 警报

使用以下 CloudWatch put-metric-alarm命令创建警报，根据平均 CPU 阈值增加 Auto Scaling 组的大小，至少连续两个评估期（两分钟）。要使用您自己的自定义指标，请在 `--metric-name` 中指定其名称，并在 `--namespace` 指定其命名空间。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 60 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

为横向缩减创建步进扩缩策略

要为您的自动扩缩组创建横向缩减的步进扩缩策略，请使用以下方法之一：

Console

步骤 1：为指标低阈值创建 CloudWatch 警报

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 如果需要，可以更改区域。从导航栏中，选择您的自动扩缩组所在的区域。
3. 在导航窗格中，选择 Alarms, All alarms（警报，所有警报），然后选择 Create alarm（创建警报）。
4. 选择选择指标。
5. 在 “所有指标” 选项卡上 EC2，选择 “按 Auto Scaling 组”，然后在搜索字段中输入 Auto Scaling 组的名称。然后，选择 `CPUUtilization` 并选择 Select metric（选择指标）。将显示 Specify metric and conditions（指定指标和条件）页面，其中显示一个图表以及有关指标的其他信息。
6. 在 Period（周期）下，选择警报的评估周期，例如 1 分钟。评估警报时，每个周期都聚合到一个数据点。

> **Note**
>
> 周期越短，创建的警报越敏感。

7. 在 Conditions（条件）下，执行以下操作：

   - 对于 Threshold type（阈值类型），选择 Static（静态）。
   - 对于每当 **`CPUUtilization`** 为，请指定您是希望指标值小于还是小于等于阈值以触发警报。然后，在 than（大于/小于）下，输入您希望超过警报的阈值。

   > **Important**
   >
   > 要使警报与横向缩减策略（警报的低阈值）一起使用，请确保不要选择大于或大于等于阈值。

8. 在其他配置下，执行以下操作：

   - 对于 Datapoints to alarm（触发警报的数据点数），输入指标值必须满足阈值条件才会触发警报的数据点（评估时间段）数。例如，2 个连续的 5 分钟时间段需要花 10 分钟才会调用警报状态。
   - 对于 Missing data treatment（缺失数据处理），选择 Treat missing data as bad (breaching threshold)（将丢失的数据视为不良数据（违反阈值））。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的配置 CloudWatch 警报如何处理丢失的数据。

9. 选择下一步。

   Configure actions（配置操作）页面会显示。

10. 在 Notification（通知）下面，选择一个在警报处于 `ALARM`、`OK` 或 `INSUFFICIENT_DATA` 状态时通知的 Amazon SNS 主题。

    要使告警为相同告警状态或不同告警状态发送多个通知，请选择添加通知。

    要让警报不发送通知，请选择删除。

11. 您可以保留 Configure actions（配置操作）页面的其他部分为空。将其他部分留空会创建警报，而不会将其与扩展策略相关联。然后，您可以将警报与 Amazon A EC2 uto Scaling 控制台中的扩展策略相关联。

12. 选择 Next（下一步）。

13. 输入警报的名称（例如，Step-Scaling-AlarmLow-RemoveCapacity）和可选的描述，然后选择 Next（下一步）。
14. 选择创建警报。

创建 CloudWatch 警报后，按照以下步骤继续从上次中断的地方继续。

步骤 2：为横向缩减创建步进扩缩策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 验证是否正确设置了扩缩限制。例如，如果您的组所需容量已经是最小，则需要指定一个新的最小值才能横向缩减。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
4. 在 Automatic scaling（自动扩展）选项卡的 Dynamic scaling policies（动态扩展策略）中，选择 Create dynamic scaling policy（创建动态扩展策略）。
5. 对于策略类型，选择步进扩展，然后指定该策略的名称。
6. 要获得CloudWatch 警报，请选择您的闹钟。如果您尚未创建警报，请选择创建警 CloudWatch 报，然后完成上一个过程中的步骤 4 到步骤 14 以创建警报。
7. 指定在使用 Take the action (执行操作) 来完成操作时，此策略对当前组大小进行的更改。您可以删除特定数量的实例或现有组大小的百分比，也可将组设置为准确的大小。

   例如，要创建将组容量减少两个实例的横向缩减策略，请选择 Remove，在下一个字段中输入 2，然后选择 capacity units。默认情况下，此步骤调整的上限为警报阈值，下限为负 (-) 无穷。
8. 要添加另一个步骤，请选择 Add step (添加步进)，然后定义要缩放的量以及步进相对于警报阈值的下限和上限。
9. 选择创建。

Amazon CLI

要创建横向缩减（减少容量）的步进扩缩策略，可以使用以下示例命令。将每个 *user input placeholder* 替换为您自己的信息。

使用时 Amazon CLI，首先要创建分步扩展策略，该策略向 Amazon A EC2 uto Scaling 提供有关在指标值减少时如何缩减的说明。然后，通过确定要监控的指标、为警报定义指标的低阈值和其他详细信息，并将警报与扩缩策略关联，您可以创建警报。

步骤 1：为横向缩减创建策略

使用以下put-scaling-policy命令创建名为的分步扩展策略my-step-scale-in-policy，其调整类型为，当关联的ChangeInCapacity CloudWatch 警报违反指标低阈值时，该策略会将组的容量减少 2 个实例。

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

记下策略的 Amazon Resource Name (ARN)。您需要它来为策略创建 CloudWatch 警报。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy
}
```

步骤 2：为指标低阈值创建 CloudWatch 警报

使用以下 CloudWatch put-metric-alarm命令创建警报，根据平均 CPU 阈值 40% 减小 Auto Scaling 组的大小，至少连续两个评估周期（两分钟）。要使用您自己的自定义指标，请在 --metric-name 中指定其名称，并在 --namespace 指定其命名空间。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmLow-RemoveCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 40 \
  --comparison-operator LessThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

简单扩展策略

以下示例说明如何使用 CLI 命令创建简单扩缩策略。本文档中将保留这些命令，以供希望使用它们的客户进行参考，但建议您改用目标跟踪或步进扩缩策略。

与步进扩展策略类似，简单的扩展策略要求您为扩展策略创建 CloudWatch 警报。在您创建的策略中，您还必须定义添加还是删除实例及实例的数量，或者将组设置为确切的大小。

步进扩缩策略和简单扩缩策略之间的主要区别之一是步进扩缩策略可以进行步进调整。通过步进扩缩，您可以根据自己指定的步进调整对组的大小进行更大或更小的更改。

简单扩缩策略还必须等待正在进行的扩缩活动或运行状况检查替换完成并且冷却时间结束，然后才能响应其他警报。与之对比，步进扩缩策略会继续响应其他警报，甚至在进行扩缩活动或运行状况检查替换时也是如此。这意味着 Amazon A EC2 uto Scaling 会在收到警报消息时对所有警报违规行为进行评估。因此，建议您使用步进扩缩策略，即使您只有一个扩缩调整。

Amazon A EC2 uto Scaling 最初仅支持简单的扩展策略。如果您在引入目标跟踪和步进扩缩策略前已创建自己的扩缩策略，则您的策略将被视为简单扩缩策略。

为横向扩展创建简单扩缩策略

使用以下put-scaling-policy命令创建一个名为的简单扩展策略`my-simple-scale-out-policy`，其调整类型为，当关联的`PercentChangeInCapacity` CloudWatch 警报违反指标高阈值时，该策略会将组的容量增加 30%。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity
```

记下策略的 Amazon Resource Name (ARN)。您需要它来为策略创建 CloudWatch 警报。

为横向缩减创建简单扩缩策略

使用以下put-scaling-policy命令创建名为的简单扩展策略`my-simple-scale-in-policy`，其调整类型为，当关联的`ChangeInCapacity` CloudWatch 警报违反指标低阈值时，该策略会将组的容量减少一个实例。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
```

```
  --adjustment-type ChangeInCapacity --cooldown 180
```

记下策略的 Amazon Resource Name (ARN)。您需要它来为策略创建 CloudWatch 警报。

扩展 Amazon A EC2 uto Scaling 的冷却时间

> **Important**
>
> 作为最佳实践，我们建议您不要使用简单扩缩策略和扩缩冷却。目标跟踪扩缩策略或步进扩缩策略更能提升扩缩性能。对于随着扩缩指标值的减小或增加而按比例更改 Auto Scaling 组大小的扩缩策略，我们建议采用目标跟踪而不是简单扩缩或分步扩缩。

当您为自动扩缩组创建简单的扩缩策略时，我们建议您同时配置扩缩冷却。

在您的 Auto Scaling 组启动或终止实例后，它首先等待冷却时间结束，然后才能启动简单扩缩策略启动的任何扩缩活动。冷却时间的用途是让您的自动扩缩组在先前的扩缩活动产生明显的作用之前保持稳定，防止其启动或终止其他实例。

例如，假设一个有关 CPU 利用率的简单扩缩策略建议启动两个实例。Amazon A EC2 uto Scaling 会启动两个实例，然后暂停扩展活动，直到冷却时间结束。冷却时间结束后，简单扩缩策略启动的任何扩缩活动都可以恢复。如果 CPU 利用率再次超过告警阈值上限，则 Auto Scaling 组将再次横向扩展，而冷却时间也会再次生效。不过，如果两个实例足以将指标值降为正常水平，该组会保持其当前大小。

内容

注意事项

在使用简单扩缩策略和扩缩冷却时，应注意以下事项：

- 目标跟踪扩缩策略和步进扩缩策略可以立即启动横向扩展活动，而无需等待冷却时间结束。相反，每当您的自动扩缩组启动实例时，单个实例都有一个预热时间。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。
- 当计划的操作在计划的时间开始时，它还可能会立即启动扩缩活动，而无需等待冷却时间结束。

- 如果实例运行状况不佳，Amazon A EC2 uto Scaling 不会等到冷却时间结束后再替换运行状况不佳的实例。
- 当多个实例启动或终止时，冷却时间（无论是原定设置冷却还是特定于扩缩策略的冷却）会从最后一个实例完成启动或终止时开始生效。
- 当您手动扩展 Auto Scaling 组时，预设情况下不会等待冷却时间结束。但是，在使用 Amazon CLI 或 SDK 进行手动缩放时，您可以覆盖此行为并遵守默认的冷却时间。
- 预设情况下，Elastic Load Balancing 会等待 300 秒，以便完成注销（Connection Draining）过程。如果组位于 Elastic Load Balance 负载均衡器后面，它将等待终止的实例完成注销，然后再开始计算冷却时间。

生命周期挂钩可能会导致额外的延迟

如果触发了生命周期钩子，则冷却时间将在您完成生命周期操作后或超时时间结束后开始计算。例如，假设某个 Auto Scaling 组具有一个有关实例启动的生命周期钩子。当应用程序需求增加时，该组会启动实例以增加容量。由于存在生命周期钩子，所以实例将置于等待状态，并且由简单扩缩策略触发的扩缩活动将被暂停。当实例进入 `InService` 状态时，冷却时间开始。冷却时间结束后，将恢复简单扩缩策略活动。

启用 Elastic Load Balancing 时，对于横向缩减，冷却时间从选择终止的实例开始连接耗尽（取消注册延迟）时开始。冷却时间不会等待连接耗尽完成或生命周期挂钩完成其操作。这意味着，横向缩减事件的结果反映在组的容量中后，因简单扩缩策略触发的任何扩缩活动都可以立即恢复。否则，等待所有三个活动（Connection Draining、生命周期钩子和冷却时间）完成会显著增加 Auto Scaling 组暂停扩缩所需的时间量。

更改原定设置冷却时间

最初在 Amazon A EC2 uto Scaling 控制台中创建 Auto Scaling 组时，您无法设置默认冷却时间。预设情况下，此冷却时间设置为 300 秒（5 分钟）。如果需要，您可以在创建组后更新此设置。

更改原定设置冷却时间（控制台）

创建 Auto Scaling 组后，在 Details（详细信息）选项卡上，依次选择Advanced configurations（高级配置）、Edit（编辑）。对于 Default cooldown（原定设置冷却），请根据实例启动时间或其他应用程序需求选择所需的时长。

更改原定设置冷却时间（Amazon CLI）

使用以下命令更改新的或现有 Auto Scaling 组的原定设置冷却。如果未定义原定设置冷却，则使用 300 秒的原定设置。

- create-auto-scaling-group
- update-auto-scaling-group

要确认默认冷却时间值，请使用describe-auto-scaling-groups命令。

为特定的简单扩缩策略设置冷却时间

预设情况下，所有简单扩缩策略都使用为 Auto Scaling 组定义的原定设置冷却时间。要设置特定简单扩缩策略的冷却时间，请在创建或更新策略时使用可选的冷却参数。在为策略指定冷却时间时，它将覆盖原定设置冷却。

对于扩缩策略特定的冷却时间，一个常见的使用场景是横向缩减策略。由于此策略会终止实例，因此 Amazon A EC2 uto Scaling 需要更少的时间来确定是否终止其他实例。终止实例应该是比启动实例快得多的操作。因此，300 秒的默认冷却时间太长。在这种情况下，为横向缩减策略设置一个具有较低值的扩缩策略特定冷却时间，可让该组更快横向缩减，从而帮助降低成本。

要在控制台中创建或更新简单扩缩策略，请在创建该组之后选择 Automatic scaling（弹性伸缩）选项卡。要使用创建或更新简单的扩展策略 Amazon CLI，请使用put-scaling-policy命令。有关更多信息，请参阅 步进和简单扩展策略。

基于 Amazon SQS 的扩缩策略

> **Important**
>
> 以下信息和步骤向您展示了在将队列属性作为自定义指标发布到之前，如何使用队列属性计算每个实例的 Amazon SQS `ApproximateNumberOfMessages` 队列待办事项。 CloudWatch 但是，您现在可以使用指标数学来节省发布自己的指标所花费的成本和精力。有关更多信息，请参阅 使用指标数学创建目标跟踪扩缩策略。

您可以根据亚马逊简单队列服务 (Amazon SQS) 队列中系统负载的变化来扩展 Auto Scaling 组。要了解有关如何使用 Amazon SQS 的更多信息，请参阅 Amazon 简单队列服务开发人员指南。

在一些情况下，您可能会考虑根据 Amazon SQS 队列中的活动扩展。例如，假设您有 Web 应用程序，让用户上传图像并联机使用。在此场景中，每个图像需要先调整大小并编码，然后才能发布。该应用程序在 Auto Scaling 组中的 EC2 实例上运行，并且配置为处理您的典型上传速率。不正常的实例将终止并进行替换，以始终保持当前的实例等级。该应用程序将图像的原始位图数据放在 SQS 队列中以进行处理。它处理这些图像，然后将处理的图像发布到某个位置以供用户查看。如果上传的图像数不随

时间波动，则适用于此场景的架构可以良好运作。但是，如果上传数量随着时间波动，您可以考虑使用动态扩展来缩放 Auto Scaling 组的容量。

内容

将目标跟踪与合适的指标结合使用

如果您使用基于自定义 Amazon SQS 队列指标的目标跟踪扩展策略，则动态扩展可以更有效地适应应用程序的需求曲线。有关为目标跟踪选择指标的更多信息，请参阅 选择指标。

使用诸如目标跟踪之类的 A CloudWatch mazon SQS 指标的问题
在ApproximateNumberOfMessagesVisible于，队列中的消息数量可能不会与处理队列消息的 Auto Scaling 组的大小成比例变化。这是因为 SQS 队列中的消息数不仅仅定义所需的实例数。Auto Scaling 组中的实例数可能由多种因素决定，包括处理消息所需的时间以及可接受的延迟长度（队列延迟）。

该解决方案使用要维护的每个实例的积压 指标以及每个实例的可接受积压 的目标值。您可以按以下所示计算这些数字：

- 每个实例的积压：要计算您每个实例的积压，首先通过 ApproximateNumberOfMessages 度列属性确定 SQS 队列的长度（可从队列中检索的消息数）。将该数字除以队列的运行容量（对于 Auto Scaling 组，这是处于 InService 状态的实例数量），以获得每个实例的积压。
- 每个实例的可接受积压：要计算您的目标值，请先确定您的应用程序可以接受的延迟。然后，将可接受的延迟值除以 EC2 实例处理消息所需的平均时间。

例如，假设您当前有一个包含 10 个实例的自动扩缩组，并且队列中的可见消息数量（ApproximateNumberOfMessages）是 1500。如果每条消息的平均处理时间为 0.1 秒，最长可接受延迟为 10 秒，则每个实例的可接受积压为 10/0.1，即 100 条消息。这意味着您的目标跟踪策略的目标值为 100。当每个实例的积压达到目标值时，将发生横向扩展事件。由于每个实例已经积压了 150 条消息（10 个实例 1500 条消息），该组会横向扩展并增加了 5 个实例以维持与目标值的比例。

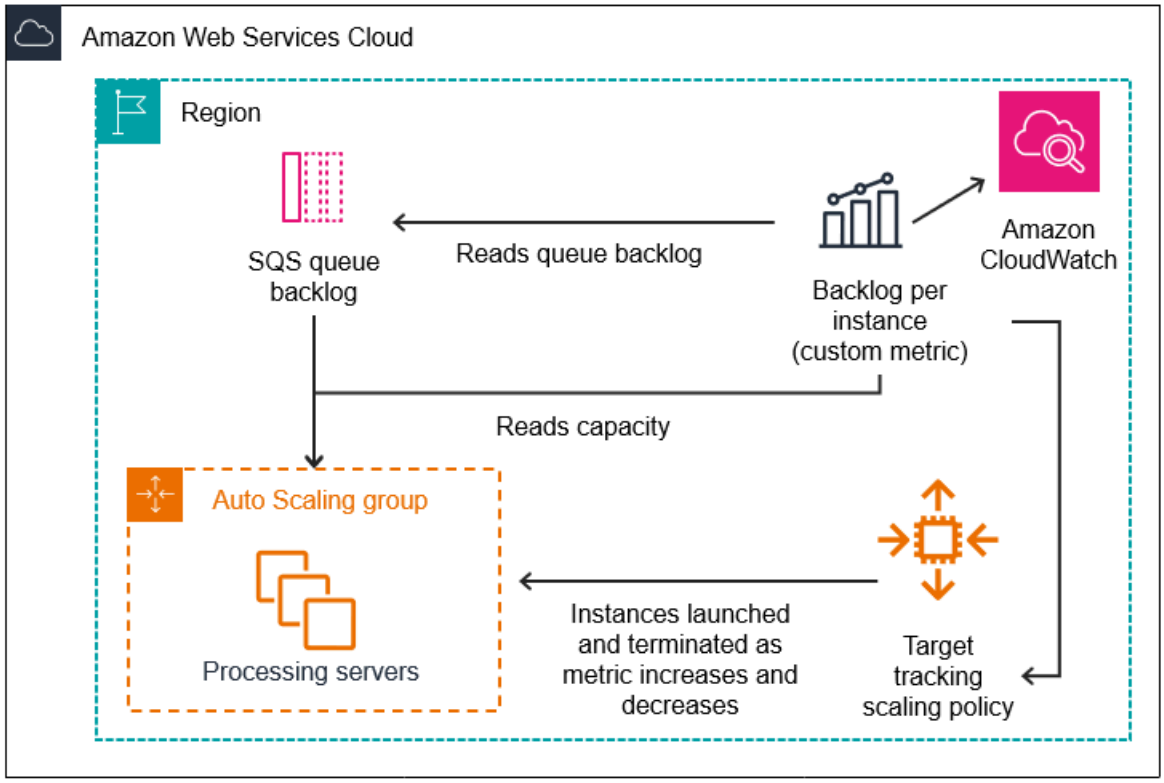
以下过程演示如何发布自定义指标和创建目标跟踪扩展策略，以根据这些计算值来配置您的 Auto Scaling 组进行扩展。

> **Important**
>
> 请记住，为了降低成本，请改用指标数学。有关更多信息，请参阅 使用指标数学创建目标跟踪扩缩策略。

此配置有三个主要部分：

- 一个 Auto Scaling 组，用于管理 EC2 实例，以便处理来自 SQS 队列的消息。
- 发送 CloudWatch 到 Amazon 的自定义指标，用于衡量 Auto Scaling 组中每个 EC2 实例队列中的消息数量。
- 一种目标跟踪策略，用于将 Auto Scaling 组配置为根据自定义指标和设定的目标值进行扩展。CloudWatch 警报会调用扩展策略。

下图演示了此配置的架构。



限制

您必须使用 Amazon CLI 或 SDK 将您的自定义指标发布到 CloudWatch。然后，您可以使用监控您的指标 Amazon Web Services Management Console。

在以下各节中， Amazon CLI 您将使用来完成需要执行的任务。例如，要获取反映队列当前使用情况的指标数据，可以使用 SQS get-queue-attributes命令。

配置基于 Amazon SQS 的缩放

以下过程介绍如何基于 Amazon SQS 配置自动扩展。您将学习如何创建 CloudWatch 自定义指标、如何使用设置目标跟踪策略以及如何测试您的配置。 Amazon CLI

在开始之前，请确保已 Amazon CLI 安装并配置。此外，您必须有一个 Amazon SQS 队列才能使用。以下任务假设您已经有一个队列（标准队列或 FIFO）、一个 Auto Scaling 组以及运行使用该队列的应用程序的 EC2 实例。

有关 Amazon SQS 权限的更多信息，请参阅 Amazon 简单队列服务开发人员指南。

任务

- 步骤 1：创建 CloudWatch 自定义指标
- 步骤 2：创建目标跟踪扩展策略
- 步骤 3：测试扩展策略

步骤 1：创建 CloudWatch 自定义指标

自定义指标是使用您选择的指标名称和命名空间定义的。自定义指标的命名空间不能以 AWS/ 开头。有关发布自定义指标的更多信息，请参阅 Amazon CloudWatch 用户指南中的发布自定义指标主题。

按照此步骤首先从您的 Amazon 账户中读取信息来创建自定义指标。然后，计算前面的章节中建议的每个实例的积压指标。最后，以 1 分钟为间 CloudWatch 隔发布此数字。只要可能，我们强烈建议您按 1 分钟粒度的指标进行扩展，以确保更快地响应系统负载变化。

创建 CloudWatch 自定义指标 (Amazon CLI)

1. 使用 SQS get-queue-attributes 命令获取在队列中等待的消息数 (`ApproximateNumberOfMessages`)：

```
aws sqs get-queue-attributes --queue-url https://
sqs.region.amazonaws.com/123456789/MyQueue \
  --attribute-names ApproximateNumberOfMessages
```

2. 使用 describe-auto-scaling-groups 命令获取组的运行容量，这是处于 `InService` 生命周期状态的实例数。此命令返回 Auto Scaling 组的实例及其生命周期状态。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. 将该队列中可供检索的大致消息数量除以该组的运行容量，计算得出每个实例的积压。
4. 创建每分钟运行一次的脚本，以检索每个实例的待办事项值并将其发布到 CloudWatch 自定义指标。发布自定义指标时，需要指定该指标的名称、命名空间、单位、值以及零个或多个维度。维度由维度名称和维度值组成。

   要发布您的自定义指标，请将中的*italics*占位符值替换为首选指标名称、指标值、命名空间（只要不是以 AWS “” 开头）和维度（可选），然后运行以下put-metric-data命令。

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --
namespace MyNamespace \
  --unit None --value 20 --
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

应用程序发出所需的指标之后，数据发送到 CloudWatch。该指标在 CloudWatch 控制台中可见。您可以通过登录 Amazon Web Services Management Console 并导航到该 CloudWatch 页面来访问它。然后，通过导航到指标页面或者使用搜索框搜索指标来查看指标。有关查看指标的信息，请参阅 Amazon CloudWatch 用户指南中的查看可用指标。

步骤 2：创建目标跟踪扩展策略

现在，您可以将创建的指标添加到目标跟踪扩缩策略中。

创建目标跟踪扩缩策略（Amazon CLI）

1. 使用以下cat命令可以在您的主目录的名为config.json的 JSON 文件中存储扩缩策略的目标值和自定义指标规范。将每个 *user input placeholder* 替换为您自己的信息。对于 TargetValue，计算每个实例的可接受积压指标并在此处输入。要计算此数值，请考虑正常延迟值并将其除以处理一条消息所需的平均时间（如前面的章节所述）。

   如果您没有为步骤 1 中创建的指标指定任何维度，请不要在自定义指标规范中包含任何维度。

```
$ cat ~/config.json
{
   "TargetValue":100,
   "CustomizedMetricSpecification":{
      "MetricName":"MyBacklogPerInstance",
      "Namespace":"MyNamespace",
```

```
      "Dimensions":[
         {
            "Name":"MyOptionalMetricDimensionName",
            "Value":"MyOptionalMetricDimensionValue"
         }
      ],
      "Statistic":"Average",
      "Unit":"None"
   }
}
```

2. 使用 put-scaling-policy 命令以及在前面的步骤中创建的 config.json 文件创建扩展策略。

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://~/config.json
```

这会创建两个警报：一个用于增加实例数量，另一个用于减少实例数量。它还会返回向其注册的策略的 Amazon 资源名称 (ARN) CloudWatch，该名称 CloudWatch 用于在指标阈值被违反时调用缩放。

步骤 3：测试扩展策略

在您完成设置后，验证您的扩展策略是否正常工作。您可以通过增加 SQS 队列中的消息数量，然后验证您的 Auto Scaling 组是否已启动其他 EC2 实例来对其进行测试。您也可以通过减少 SQS 队列中的消息数量，然后验证 Auto Scaling 组是否已终止 EC2实例来对其进行测试。

测试扩展函数

1. 按照创建 Amazon SQS 标准队列并发送消息或创建 Amazon SQS FIFO 队列并发送消息中的步骤，将消息添加到您的队列。请确保您增加了队列中的消息数量，使得每个实例的积压指标超过目标值。

   您的更改调用警报可能需要几分钟时间。

2. 使用 describe-auto-scaling-groups 命令确认该组已启动一个实例。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

测试横向缩减功能

1. 按照接收和删除消息（控制台）中的步骤，从队列中删除消息。请确保您减少了队列中的消息数量，使得每个实例的积压指标低于目标值。

   您的更改调用警报可能需要几分钟时间。
2. 使用 describe-auto-scaling-groups 命令确认该组已终止一个实例。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQS 和实例缩减保护

实例被终止时还未处理的消息将返回 SQS 队列中，可由仍在运行的实例进行处理。对于执行长时间运行任务的应用程序，您可以选择使用实例扩展保护来控制 Auto Scaling 组扩展时终止哪些队列工作程序。

以下伪代码显示了保护长时间运行、队列驱动的工件进程免受缩放终止的一种方法。

```
while (true)
{
  SetInstanceProtection(False);
  Work = GetNextWorkUnit();
  SetInstanceProtection(True);
  ProcessWorkUnit(Work);
  SetInstanceProtection(False);
}
```

有关更多信息，请参阅 设计您的应用程序以妥善处理实例终止。

验证 Auto Scaling 组的扩缩活动

在亚马逊 EC2 控制台的 Amazon A EC2 uto Scaling 部分，Auto Scaling 组的活动历史记录允许您查看当前正在进行的扩展活动的当前状态。扩展活动完成后，您可以看到它是否成功。在创建 Auto Scaling 组或向现有组添加扩展条件时，此方法特别有用。

当您将目标跟踪、步骤或简单扩展策略添加到 Auto Scaling 组时，Amazon A EC2 uto Scaling 会立即开始根据该指标评估策略。在指标超过阈值达到指定数量的评估期时，指标警报将变为 ALARM（警报）状态。这意味着扩展策略可能会在创建后立即引发扩展活动。在 Amazon A EC2 uto Scaling 根据扩展策略调整所需容量后，您可以验证账户中的扩展活动。如果您想收到来自 Amazon A EC2 uto

Scaling 的电子邮件通知，告知您有关扩展活动的信息，请按照中的说明进行操作亚马逊 Auto Scaling 的亚马 EC2 逊 SNS 通知选项。

> **Tip**
>
> 在以下过程中，您需要查看 Auto Scaling 组的 Activity history（活动历史记录）和 Instances（实例）部分。这两个部分都会已经显示已命名的列。要显示隐藏的列或更改显示的行数，请选择每个部分右上角的齿轮图标以打开首选项模式，根据需要更新设置，然后选择 Confirm（确认）。

要查看 Auto Scaling 组的扩缩活动（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的区域。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
4. 在 Activity（活动）选项卡的 Activity history（活动历史记录）下，Status（状态）列显示您的 Auto Scaling 组是否已成功启动或终止实例，或者扩展活动是否仍在进行中。
5. （可选）如果有很多扩缩活动，您可以选择活动历史记录顶部边缘的 > 图标，来查看下一页的扩缩活动。
6. 在实例管理选项卡的实例下，生命周期列显示实例的状态。在实例开启并且任何生命周期钩子结束后，其生命周期状态将更改为 `InService`。Health stat us 列显示对您的 EC2 实例进行实例运行状况检查的结果。

要查看 Auto Scaling 组的扩缩活动 (Amazon CLI)

使用以下 describe-scaling-activities 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

下面是示例输出。

扩展活动按开始时间排序。首先描述仍在进行的活动。

```
{
```

```
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-
b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-
tracking-policy changing the desired capacity from 3 to 2.  At 2020-02-11T18:34:31Z
 an instance was taken out of service in response to a difference between desired and
 actual capacity, shrinking the capacity from 3 to 2.  At 2020-02-11T18:34:31Z instance
 i-06c4794c2499af1df was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn"
    },
...
  ]
}
```

有关输出中字段的描述，请参阅 Amazon A EC2 uto Scaling API 参考中的活动。

有关检索已删除组的扩展活动的帮助，以及有关可能会遇到的错误类型以及如何处理这些错误的信息，请参阅 对 Amazon A EC2 uto Scaling 中的问题进行故障排除。

禁用 Auto Scaling 组的扩缩策略

本主题介绍如何临时禁用扩展策略，使其不会发起对 Auto Scaling 组包含的实例数进行的更改。禁用扩展策略时，配置详细信息将保留，以便您可以快速重新启用策略。相比在不需要时暂时删除策略并在以后重新创建，这种方法更容易。

在禁用扩展策略时，Auto Scaling 组不会为违反的指标警报进行扩展或缩减。但是，任何仍在进行的扩展活动都不会停止。

请注意，已禁用的扩展策略仍会计入您可以添加到 Auto Scaling 组的扩展策略数量配额中。

要禁用扩缩策略（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。

2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。

3. 在 Automatic scaling（自动扩展）选项卡的 Dynamic scaling policies（动态扩展策略）下，选中所需扩展策略右上角的复选框。

4. 滚动到 Dynamic scaling policies（动态扩展策略）部分的顶部，然后选择 Actions（操作）、Disable（禁用）。

当您准备好重新启用扩展策略时，请重复这些步骤，然后选择 Actions（操作）、Enable（启用）。重新启用扩展策略后，如果当前有任何警报处于 ALARM 状态，您的 Auto Scaling 组可能会立即启动扩展操作。

禁用扩展策略 (Amazon CLI)

将 put-scaling-policy 命令与 `--no-enabled` 选项一起使用，如下所示。采用与在创建策略时相同的指定方式，在命令中指定所有选项。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
   --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
   --no-enabled
```

重新启用扩展策略 (Amazon CLI)

将 put-scaling-policy 命令与 `--enabled` 选项一起使用，如下所示。采用与在创建策略时相同的指定方式，在命令中指定所有选项。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
   --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
   --enabled
```

描述扩展策略 (Amazon CLI)

使用 describe-policies 命令验证扩展策略的启用状态。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \
   --policy-names my-scaling-policy
```

下面是示例输出。

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "my-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-9ca53fdd-7cf5-4223-938a-ae1199204502",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 70.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
```

```
}
```

删除自动扩缩组的扩缩策略

当您不再需要某个扩展策略时，可将其删除。根据扩展策略的类型，您可能还需要删除 CloudWatch 警报。删除目标跟踪扩展策略也会删除所有关联的 CloudWatch 警报。删除分步扩展策略或简单的扩展策略会删除底层警报操作，但不会删除 CloudWatch 警报，即使警报不再具有关联操作也是如此。

删除扩展策略（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Automatic scaling（自动扩展）选项卡的 Dynamic scaling policies（动态扩展策略）下，选中所需扩展策略右上角的复选框。
4. 滚动到 Dynamic scaling policies（动态扩展策略）部分的顶部，然后选择 Actions（操作）、Delete（删除）。
5. 当系统提示进行确认时，选择 Yes, Delete（是，删除）。
6. （可选）如果您删除了分步扩展策略或简单扩展策略，请执行以下操作以删除与该策略关联的 CloudWatch 警报。您可以跳过这些子步骤来保留警报以供将来使用。

   a. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。

   b. 在导航窗格上，选择 Alarms（警报）。

   c. 选择警报（例如 `Step-Scaling-AlarmHigh-AddCapacity`），然后选择 Action（操作）、Delete（删除）。

   d. 当系统提示进行确认时，选择 Delete（删除）。

要获取 Auto Scaling 组 (Amazon CLI) 的扩展策略

在您删除扩展策略之前，请使用以下 describe-policies 命令查看为 Auto Scaling 组创建了哪些扩展策略。可以在删除策略和 CloudWatch 警报时使用输出。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

可以使用 --query 参数按扩展策略类型筛选结果。此 query 语法仅在 Linux 或 macOS 上有效。在 Windows 上，请将单引号更改为双引号。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

下面是示例输出。

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "PolicyName": "cpu50-target-tracking-scaling-policy",
        "PolicyARN": "PolicyARN",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [
            {
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
                "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
                "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
            }
        ],
        "TargetTrackingConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ASGAverageCPUUtilization"
            },
            "TargetValue": 50.0,
            "DisableScaleIn": false
        },
        "Enabled": true
    }
]
```

删除扩展策略 (Amazon CLI)

使用以下 delete-policy 命令。

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \
  --policy-name cpu50-target-tracking-scaling-policy
```

要删除您的 CloudWatch 警报 (Amazon CLI)

对于分步和简单扩展策略，请使用 delete-alarms 命令删除与策略关联的 CloudWatch 警报。您可以跳过此步骤来保留该警报以备将来使用。您可以一次删除一个或多个警报。例如，使用以下命令可删除 Step-Scaling-AlarmHigh-AddCapacity 和 Step-Scaling-AlarmLow-RemoveCapacity 警报：

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-
Scaling-AlarmLow-RemoveCapacity
```

Amazon CLI的扩缩策略示例

您可以通过 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或为 Amazon A EC2 uto Scaling 创建扩展策略 SDKs。

以下示例显示了如何使用 Amazon CLI put-scaling-policy命令为 Amazon A EC2 uto Scaling 创建扩展策略。将每个 *user input placeholder* 替换为您自己的信息。

要开始使用编写扩展策略 Amazon CLI，请参阅目标跟踪扩展策略和中的入门练习步进和简单扩展策略。

示例 1：应用具有预定义指标规范的目标跟踪扩展策略

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ASGAverageCPUUtilization"
  }
}
```

有关更多信息，请参阅PredefinedMetricSpecification《Amazon A EC2 uto Scaling API 参考》。

> **Note**
>
> 如果文件不在当前目录中，请键入文件的完整路径。有关从文件读取 Amazon CLI 参数值的更多信息，请参阅《 Amazon Command Line Interface 用户指南》中的从文件加载 Amazon CLI 参数。

示例 2：应用具有自定义指标规范的目标跟踪扩展策略

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

有关更多信息，请参阅CustomizedMetricSpecification《Amazon A EC2 uto Scaling API 参考》。

示例 3：为仅向外扩展应用目标跟踪扩展策略

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
  },
```

```
  "DisableScaleIn": true
}
```

示例 4：为向外扩展应用步进扩展策略

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
                     MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

记下策略的 Amazon Resource Name (ARN)。创建警报时需要 ARN。 CloudWatch

示例 5：为缩减应用步进扩展策略

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

记下策略的 Amazon Resource Name (ARN)。创建警报时需要 ARN。 CloudWatch

示例 6：为向外扩展应用简单扩展策略

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2
```

记下策略的 Amazon Resource Name (ARN)。创建警报时需要 ARN。 CloudWatch

示例 7：为缩减应用简单扩展策略

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
```

```
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

记下策略的 Amazon Resource Name (ARN)。创建警报时需要 ARN。 CloudWatch

Amazon A EC2 uto Scaling 的预测性扩展

预测性扩展通过分析历史负载数据来检测流量中的每日或每周模式。它使用这些信息来预测未来的容量需求，因此 Amazon A EC2 uto Scaling 可以主动增加您的 Auto Scaling 组的容量以匹配预期的负载。

预测式扩展非常适合以下情况：

- 周期性流量，例如正常营业时间内的高资源利用率以及晚上和周末的低资源利用率
- 重复 on-and-off的工作负载模式，例如批处理、测试或定期数据分析
- 初始化需要很长时间的应用程序，从而在向外扩展事件期间对应用程序性能造成明显的延迟影响

一般来说，如果您有定期的流量增长模式以及需要很长时间才能初始化的应用程序，则应考虑使用预测式扩展。与仅使用动态扩展相比，预测式扩展可以通过在预测负载之前启动容量来帮助您更快地扩展。预测性扩展还可以帮助您避免过度配置容量，从而有可能为您节省 EC2 账单上的开支。

例如，考虑在营业时间内具有高利用率以及夜间具有低利用率的应用程序。在每个工作日开始时，预测式扩展可以在流量第一次涌入之前增加容量。这有助于您的应用程序在利用率较低的时期内保持高可用性和性能。您不必等待动态扩展来响应不断变化的流量。您也不必花时间查看应用程序的负载模式，并尝试使用计划扩展计划适当的容量。

主题

预测式扩展的工作方式

本主题说明预测性扩展的工作原理，并描述创建预测性扩展策略时需要考虑的内容。

主题

工作方式

要使用预测扩展，请创建预测性扩展策略，指定要监控和分析的 CloudWatch 指标。要使预测性扩展开始预测未来值，此指标必须包含至少 24 小时的数据。

创建策略后，预测性扩展将开始分析最多过去 14 天的指标数据，以确定模式。其使用此分析生成未来 48 小时容量需求的每小时预测。使用最新 CloudWatch 数据，预测每 6 小时更新一次。随着新数据的出现，预测性扩展能够不断提高未来预测的准确性。

首次启用预测性扩展时，该功能将在仅预测模式下运行。在此模式下，它会生成容量预测，但不会根据这些预测实际扩展自动扩缩组。这使您可以评估预测的准确性和适用性。您可以使用 `GetPredictiveScalingForecast` API 操作或 Amazon Web Services Management Console查看预测数据。

查看预测数据并决定根据该数据开始扩展后，将扩展策略切换到预测和扩展模式。在此模式中：

- 如果预测预计负载会增加，Amazon A EC2 uto Scaling 将通过扩展来增加容量。
- 如果预测预计负载会减少，则不会横向缩减以减少容量。如果要移除不再需要的容量，则必须创建动态扩缩策略。

默认情况下，Amazon A EC2 uto Scaling 会在每小时开始时根据该小时的预测对您的 Auto Scaling 群组进行扩展。您可以选择使用 `PutScalingPolicy` API 操作中的 `SchedulingBufferTime` 属性或 Amazon Web Services Management Console中的预启动实例设置来指定更早的开始时间。这会导致 Amazon A EC2 uto Scaling 在预测的需求之前启动新实例，从而使它们有时间启动并准备好处理流量。

为了支持在预测的需求之前启动新实例，强烈建议您为自动扩缩组启用默认实例预热。这指定了横向扩展活动结束后的一段时间，在此期间，即使动态扩展策略指示应减少容量，Amazon A EC2 uto Scaling 也不会缩减规模。这可以帮助您确保新启动的实例在考虑进行横向缩减操作之前，有足够的时间开始为增加的流量提供服务。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

最大容量限制

Auto Scaling 组具有最大容量设置，该设置限制了可以为该组启动的最大 EC2实例数。默认情况下，设置扩缩策略后，这些策略无法将容量增加到高于最大容量。

或者，您可以允许在预测容量接近或超过自动扩缩组的最大容量时，自动增加组的最大容量。要启用此行为，请使用 `PutScalingPolicy` API 操作中的 `MaxCapacityBreachBehavior` 和 `MaxCapacityBuffer` 属性或 Amazon Web Services Management Console中的最大容量行为设置。

> **Warning**
>
> 允许自动增加最大容量时应谨慎行事。如果不对增加的最大容量进行监控和管理，则可能导致启动的实例数量超出预期。然后，增加的最大容量将变为自动扩缩组新的正常最大容量，直到您手动对其进行更新。最大容量不会自动降回到原来的最大值。

注意事项

- 确认预测式扩展是否适合您的工作负载。如果工作负载显示特定于星期几的什么时间的重复负载模式，则工作负载非常适合预测式扩展。若要检查这一点，请在仅预测模式下配置预测性扩展策略，然后参考控制台中的建议。Amazon A EC2 uto Scaling 根据对潜在政策绩效的观察结果提供建议。在允许预测式扩展主动扩展应用程序之前，请评估预测及建议。
- 预测式扩展至少需要 24 小时的历史数据才能开始预测。但是，如果历史数据跨越整整两周，预测会更有效。如果您通过创建新的 Auto Scaling 组并删除旧组来更新应用程序，则新的 Auto Scaling 组需要 24 小时的历史加载数据，然后预测式扩展才能再次开始生成预测。您可以使用自定义指标来聚合新旧自动扩缩组之间的指标。否则，您可能需要等待几天才能获得更准确的预测。
- 选择一个负载指标，该指标应准确代表应用程序的全部负载，并且是应用程序中对扩缩最重要的方面。
- 将动态扩缩与预测性扩展结合使用可帮助您紧密跟踪应用程序的需求曲线，在低流量期间进行横向缩减并在流量高于预期时横向扩展。当多个扩展策略处于活动状态时，每个策略将独立确定所需容量，并将所需容量设置为其中的最大容量。例如，如果要求 10 个实例保持目标跟踪扩展策略中的目标利用率，并且需要 8 个实例保持在预测式扩展策略中的目标利用率，则组的所需容量设置为 10。如果您不熟悉动态扩缩，则建议您使用目标跟踪扩缩策略。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。
- 预测性扩展的核心假设是 Auto Scaling 组是同质的，且所有实例的容量相等。如果您的组不是这样，预测的容量可能不准确。因此，在为混合实例组创建预测性扩展策略时务必谨慎，因为可以预置容量不相等的不同类型实例。以下是一些预测容量不准确的例子：

- 您的预测性扩展策略基于 CPU 利用率，但是每个 Auto Scaling 实例CPUs 上的 v 数因实例类型而异。
- 您的预测性扩缩策略基于网络进入或网络外出，但每个 Auto Scaling 实例的网络带宽吞吐量因实例类型而异。例如，M5 和 M5n 实例类型相似，但 M5n 实例类型显著提高了网络吞吐量。

支持的区域

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- 亚太地区（香港）
- 亚太地区（雅加达）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 加拿大（中部）
- 中国（北京）
- 中国（宁夏）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- 欧洲地区（斯德哥尔摩）
- 中东（巴林）
- 中东（阿联酋）

- 南美洲（圣保罗）
- Amazon GovCloud （美国东部）
- Amazon GovCloud （美国西部）

创建自动扩缩组的预测性扩展策略

以下过程可帮助您使用 Amazon Web Services Management Console 或创建预测性扩展策略 Amazon CLI。

如果 Auto Scaling 组是新的，则它必须提供至少 24 小时的数据，Amazon A EC2 uto Scaling 才能为其生成预测。

内容

创建预测式扩展策略（控制台）

如果这是您第一次创建预测性扩展策略，则建议您使用控制台在仅预测模式下创建多个预测性扩展策略。这样便可测试不同指标和目标值的潜在影响。您可以为每个 Auto Scaling 组创建多个预测式扩展策略，但只能将其中一个策略用于主动扩展。

通过控制台创建预测性扩缩策略（预定义指标）

按照以下程序，使用预定义的指标（每个目标的 CPU、网络 I/O 或应用程序负载均衡器请求数）创建预测性扩缩策略。创建预测性扩缩策略的最简单方式，是使用预定义指标。如果您希望使用自定义指标，请参阅 通过控制台创建预测性扩缩策略（自定义指标）。

创建预测式扩展策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在自动扩展选项卡的扩展策略中，选择创建预测式扩展策略。
4. 输入策略的名称。

5. 开启 “基于预测扩展”，授予 Amazon A EC2 uto Scaling 立即开始扩展的权限。

   若要将策略保持在仅预测模式，请保持根据预测进行扩展关闭状态。

6. 对于指标，从选项列表中选择指标。选项包括 CPU、网络输入、网络输出、Application Load Balancer 请求计数和自定义指标对。

   如果您选择每目标的 Application Load Balancer 请求计数，请在目标组中选择目标组。每目标的 Application Load Balancer 请求计数仅在您已将 Application Load Balancer 目标组附加到 Auto Scaling 组时才支持。

   如果您选择自定义指标对，请从负载指标和扩展指标的下拉列表中选择单个指标。

7. 对于目标利用率，输入 Amazon A EC2 uto Scaling 应保持的目标值。Amazon A EC2 uto Scaling 会扩展您的容量，直到平均利用率达到目标利用率，或者直到达到您指定的最大实例数。

如果您的扩展指标是...	然后目标利用率表示...
CPU	每个实例理想情况下应使用的 CPU 百分比。
网络输入	每个实例理想情况下应接收的平均每分钟字节数。
网络输出	每个实例理想情况下应发出的平均每分钟字节数。
每目标的 Application Load Balancer 请求计数	每个实例理想情况下应接收的平均每分钟请求数。

8. （可选）对于预启动实例，请选择您希望在预测调用增加负载之前启动实例的距离。
9. （可选）对于最大容量行为，请选择当预测容量超过定义的最大容量时，是否让 Amazon A EC2 uto Scaling 向外扩展到高于组的最大容量。通过开启此设置，您将可以在预测您的流量会触及最大值期间进行横向扩展。
10. （可选）对于高于预测容量的缓冲区最大容量，选择在预测容量接近或超过最大容量时要使用的附加容量。该值是作为相对于预测容量的百分比指定的。例如，如果缓冲区为 10，这意味着 10% 的缓冲区。因此，如果预测容量为 50，最大容量为 40，则有效的最大容量是 55。

    如果设置为 0，Amazon A EC2 uto Scaling 可能会将容量扩展到高于最大容量以等于但不超过预测容量。

11. 选择创建预测式扩展策略。

通过控制台创建预测性扩缩策略（自定义指标）

按照以下程序，使用自定义指标创建预测性扩缩策略。自定义指标可以包括提供的其他指标 CloudWatch 或您发布到的指标 CloudWatch。要使用每个目标的 CPU、网络 I/O 或应用程序负载均衡器请求数指标，请参阅 通过控制台创建预测性扩缩策略（预定义指标）。

要使用自定义指标创建预测性扩缩策略，您必须执行以下操作：

- 您必须提供原始查询，让 Amazon A EC2 uto Scaling 与中的指标进行交互 CloudWatch。有关更多信息，请参阅 使用自定义指标的高级预测性扩展策略。为确保 Amazon A EC2 uto Scaling 可以从中提取指标数据 CloudWatch，请确认每个查询都返回了数据点。使用 CloudWatch 控制台或 CloudWatch GetMetricDataAPI 操作进行确认。

  > **Note**
  >
  > 我们在 Amazon A EC2 uto Scaling 控制台的 JSON 编辑器中提供了示例 JSON 有效负载。这些示例为您提供了添加由提供的其他 CloudWatch 指标 Amazon 或您之前发布到的指标所需的键值对的参考。 CloudWatch您可以将这些示例负载作为起点，然后根据需要进行自定义。

- 如果您使用任何指标数学，则必须手动构造适合您的独特应用场景的 JSON。有关更多信息，请参阅使用指标数学表达式。在策略中使用指标数学之前，请确认基于指标数学表达式的指标查询有效并返回了单个时间序列。使用 CloudWatch 控制台或 CloudWatch GetMetricDataAPI 操作进行确认。

如果因提供的数据错误（例如自动扩缩组名称出错），您的查询出现错误，则预测将没有任何数据。要排查自定义指标问题，请参阅 预测性扩展策略中自定义指标的注意事项。

创建预测式扩展策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在自动扩展选项卡的扩展策略中，选择创建预测式扩展策略。
4. 输入策略的名称。
5. 开启“基于预测扩展”，授予 Amazon A EC2 uto Scaling 立即开始扩展的权限。

若要将策略保持在仅预测模式，请保持根据预测进行扩展关闭状态。

6. 对于 Metrics（指标），选择 Custom metric pair（自定义指标对）。

   a. 对于加载指标，选择自定义 CloudWatch 指标以使用自定义指标。构造包含策略的负载指标定义的 JSON 负载，然后将其粘贴到 JSON 编辑器框中，替换其中已有的内容。

   b. 对于缩放指标，请选择自定义 CloudWatch 指标以使用自定义指标。构造包含策略的扩缩指标定义的 JSON 负载，然后将其粘贴到 JSON 编辑器框中，替换其中已有的内容。

   c. （可选）要添加自定义容量指标，请选中 Add custom capacity metric（添加自定义容量指标）复选框。构造包含策略的容量指标定义的 JSON 负载，然后将其粘贴到 JSON 编辑器框中，替换其中已有的内容。

      如果您的容量指标数据跨越多个自动扩缩组，则只需启用此选项即可创建新的容量时间序列。在这种情况下，必须使用指标数学将数据聚合成单个时间序列。

7. 对于目标利用率，输入 Amazon A EC2 uto Scaling 应保持的目标值。Amazon A EC2 uto Scaling 会扩展您的容量，直到平均利用率达到目标利用率，或者直到达到您指定的最大实例数。

8. （可选）对于预启动实例，请选择您希望在预测调用增加负载之前启动实例的距离。

9. （可选）对于最大容量行为，请选择当预测容量超过定义的最大容量时，是否让 Amazon A EC2 uto Scaling 向外扩展到高于组的最大容量。通过开启此设置，您将可以在预测您的流量会触及最大值期间进行横向扩展。

10. （可选）对于高于预测容量的缓冲区最大容量，选择在预测容量接近或超过最大容量时要使用的附加容量。该值是作为相对于预测容量的百分比指定的。例如，如果缓冲区为 10，这意味着 10% 的缓冲区。因此，如果预测容量为 50，最大容量为 40，则有效的最大容量是 55。

    如果设置为 0，Amazon A EC2 uto Scaling 可能会将容量扩展到高于最大容量以等于但不超过预测容量。

11. 选择创建预测式扩展策略。

创建预测式扩展策略 (Amazon CLI)

使用以下方法为您 Amazon CLI 的 Auto Scaling 组配置预测性扩展策略。将每个 *user input placeholder* 替换为您自己的信息。

有关您可以指定的 CloudWatch 指标的更多信息，请参阅PredictiveScalingMetricSpecification《Amazon A EC2 uto Scaling API 参考》。

示例 1：创建预测但不扩展的预测式扩展策略

以下示例策略显示了完整的策略配置，该配置使用 CPU 利用率指标进行预测式扩展，其中目标利用率为 40。默认使用 ForecastOnly 模式，除非您明确指定要使用的模式。将此配置保存在名为 config.json 的文件中。

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ]
}
```

要从命令行创建策略，请在指定配置文件的情况下运行该put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-
policy",
  "Alarms": []
}
```

示例 2：预测和扩展的预测式扩展策略

要获得允许 Amazon A EC2 uto Scaling 进行预测和扩展的策略，请添加值为的属性ModeForecastAndScale。以下示例显示了使用 Application Load Balancer 请求计数指标的策略配置。目标利用率是 1000，并且预测式扩展设置为 ForecastAndScale 模式。

```
{
    "MetricSpecifications": [
        {
```

```
            "TargetValue": 1000,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ALBRequestCount",
                "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
            }
        }
    ],
    "Mode": "ForecastAndScale"
}
```

要创建此策略，请使用指定的配置文件运行put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-
asg:policyName/alb1000-predictive-scaling-policy",
  "Alarms": []
}
```

示例 3：可扩展大于最大容量的预测式扩展策略

以下示例显示如何创建一个策略，该策略可以在您需要它来处理高于正常负载时扩展高于组的最大大小限制。默认情况下，Amazon A EC2 uto Scaling 不会将您的 EC2 容量扩展到高于您定义的最大容量。但是，如果让它扩展更高，容量稍微增加以避免性能或可用性问题，可能会有所帮助。

要在 Amazon A EC2 uto Scaling 预计容量将达到或非常接近您的组的最大容量时为其预置额外容量提供空间，请指定MaxCapacityBreachBehavior和MaxCapacityBuffer属性，如以下示例所示。您必须指定值为 IncreaseMaxCapacity 的 MaxCapacityBreachBehavior。您的组可以具有的最大实例数取决于 MaxCapacityBuffer 值。

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 70,
```

```
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ],
    "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
    "MaxCapacityBuffer": 10
}
```

在本示例中，该策略配置为使用 10% 缓冲区 ("MaxCapacityBuffer": 10)，因此如果预测容量为 50 并且最大容量为 40，则实际的最大容量为 55。如果策略可以将容量扩展到高于最大容量以等于但不超过预测容量，则缓冲区为 0 ("MaxCapacityBuffer": 0)。

要创建此策略，请使用指定的配置文件运行put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-
policy",
  "Alarms": []
}
```

评估预测性扩展策略

在使用预测性扩展策略扩展 Auto Scaling 组之前，请在 Amazon A EC2 uto Scaling 控制台中查看针对您的策略的建议和其他数据。这很重要，因为在确定预测准确之前，您不希望使用预测扩展策略来扩展实际容量。

如果 Auto Scaling 组是新的，请给 Amazon A EC2 uto Scaling 24 小时的时间来创建第一个预测。

当 Amazon A EC2 uto Scaling 创建预测时，它会使用历史数据。如果您的 Auto Scaling 组还没有太多最近的历史数据，Amazon A EC2 uto Scaling 可能会临时使用根据当前可用的历史聚合创建的聚合来回填预测。预测会在策略创建日期前最多回填两周。

内容

- 查看您的预测性扩展建议
- 查看预测性扩展监控图表
- 使用监控预测性扩展指标 CloudWatch

查看您的预测性扩展建议

为了进行有效的分析，Amazon A EC2 uto Scaling 应至少有两个预测性扩展策略可供比较。（但您仍可以查看单个策略的调查结果。）创建多个策略时，您可以对使用一个指标的策略和使用另一个指标的策略进行评估。您还可以评估不同目标值和指标组合的影响。创建预测性扩展策略后，Amazon A EC2 uto Scaling 会立即开始评估哪种策略可以更好地扩展您的群组。

在 Amazon A EC2 uto Scaling 控制台中查看您的建议

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Auto Scaling 选项卡的预测性扩展策略下，您可以查看有关策略的详细信息以及我们的建议。该建议告诉您预测性扩展策略是否比不使用预测性扩展策略做得更好。

   如果您不确定预测性扩展策略是否适合您的组，请查看可用性影响和成本影响列以选择正确的策略。每列的信息告诉您该策略的影响。

   - 可用性影响：描述与不使用策略相比，该策略是否可以通过预置足够的实例来处理工作负载，从而避免对可用性的负面影响。
   - 成本影响：描述与不使用策略相比，该策略是否可以通过不过度预置实例而避免对您的成本产生负面影响。过度预置会导致您的实例未得到充分利用或处于闲置状态，这只会增加对成本的影响。

   如果您有多个策略，则在以较低成本提供最大可用性优势的策略名称旁边显示最佳预测标签。对可用性的影响给予了更多的重视。
4. （可选）要为建议结果选择所需的时间段，请从评估期下拉列表中选择您的首选值：2 天、1 周、2 周、4 周、6 周或 8 周。默认情况下，评估期为过去两周。更长的评估期可为建议结果提供更多的数据点。但是，如果您的负载模式发生了变化，例如在需求异常之后，添加更多数据点可能不会改善结果。在这种情况下，您可以通过查看最新数据来获得更有针对性的建议。

> **Note**
>
> 仅为处于仅预测模式的策略生成建议。当策略在整个评估期内处于仅预测模式时，建议功能的效果会更好。如果您在预测和扩展模式下启动策略，稍后将其切换为仅预测模式，则该策略的调查结果可能会有偏差。这是因为该策略已经为实际容量做出了贡献。

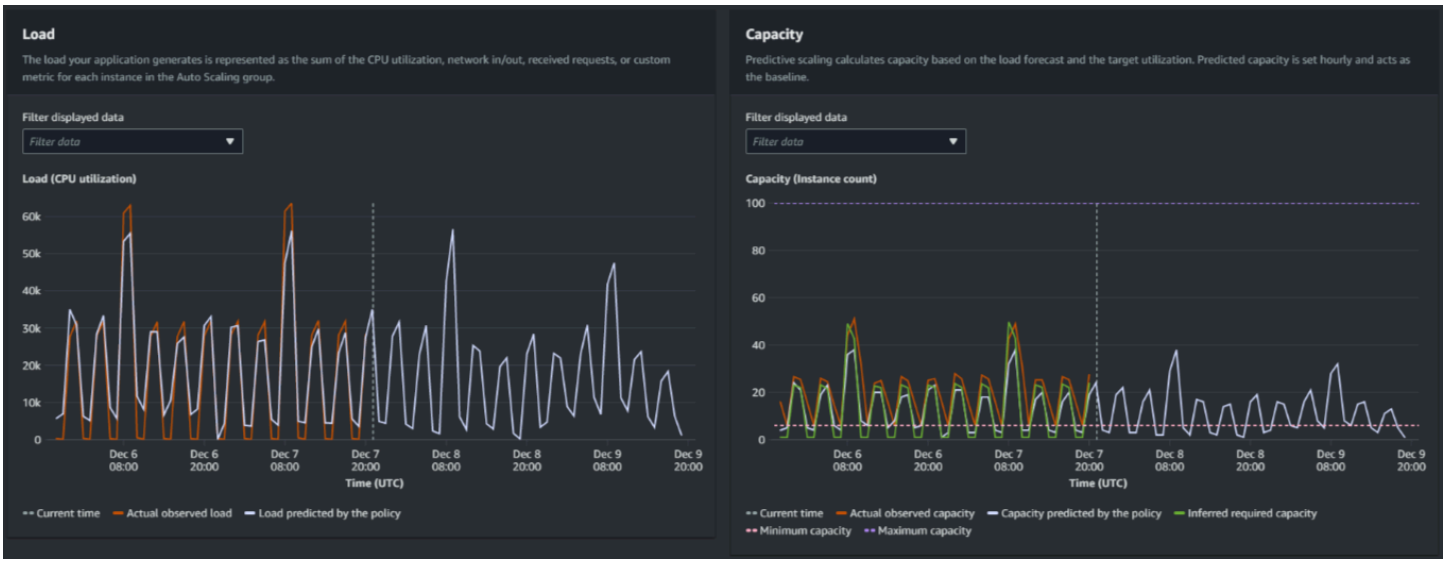
查看预测性扩展监控图表

在 Amazon A EC2 uto Scaling 控制台中，您可以查看前几天、几周或几个月的预测，以可视化策略在一段时间内的执行情况。在决定是否让策略扩展您的实际容量时，您还可以使用这些信息来评估预测的准确性。

在 Amazon A EC2 uto Scaling 控制台中查看预测性扩展监控图表

1. 从预测性扩展策略列表中选择一个策略。
2. 在监控部分中，您可以根据实际值查看策略对过去和未来负载和容量的预测。负载图表显示所选负载指标的负载预测和实际值。容量图表显示策略预测的实例数量。它还包括启动实例的实际数量。垂直线将历史值与未来预测分开。创建策略后不久，这些图表可用。
3. （可选）要更改图表中显示的历史数据量，请从页面顶部的评估期下拉列表中选择您的首选值。评估期不会以任何方式转换此页面上的数据。它只会更改显示的历史数据量。

下图显示了多次应用预测时的负载和容量图表。预测性扩展会根据您的历史负载数据来预测负载。应用程序生成的负载由自动扩缩组中每个实例的 CPU 利用率、网络输入/输出、收到的请求或自定义指标的总和来表示。预测性扩展会根据负载预测和您希望扩展指标达到的目标利用率来计算未来的容量需求。



比较负载图表中的数据

每条水平线代表一小时间隔内报告的一组不同的数据点：

1. 实际观测负荷使用所选负荷指标的总和统计数据来显示过去的每小时总负荷。
2. 策略预测的负载显示每小时的负载预测。该预测基于前两周的实际负载观测结果。

比较容量图表中的数据

每条水平线代表一小时间隔内报告的一组不同的数据点：

1. 实际观察到的容量显示了启用该GroupTotalInstances指标时您的 Auto Scaling 组过去的实际容量。此容量取决于您的其他扩展策略和所选时间段内的最小组大小。
2. 策略预测的容量显示当策略处于预测和扩展模式时，您在每个小时开始时期望拥有的基准容量。
3. 推断的所需容量显示将扩展指标维持在您所选择目标值的理想容量。
4. 最小容量显示自动扩缩组的最小容量。
5. 最大容量显示自动扩缩组的最大容量。

为了计算推断的所需容量，我们首先假设每个实例在指定目标值下的利用率相等。实际上，实例的利用率并不均等。但是，通过假设实例之间的利用率分布均匀，我们可以对所需的容量进行可能的估计。然后计算容量需求与您在预测性扩展策略中使用的扩展指标成反比。换句话说，随着容量的增加，扩展指标会以相同的速率减少。例如，如果容量翻倍，则扩展指标必定会减少一半。

推断的所需容量的公式：

```
sum of (actualCapacityUnits*scalingMetricValue)/(targetUtilization)
```

例如，我们在给定一小时内使用 `actualCapacityUnits` (`10`) 和 `scalingMetricValue` (`30`)。然后，我们采用您在预测扩展策略 (`60`) 中指定的 `targetUtilization`，计算同一小时的推断所需容量。这会返回值 5。这意味着 5 是推断出的维持容量所需的容量，与扩展指标的目标值成反比。

> **Note**
>
> 您可以使用各种杠杆来调整和提高应用程序的成本节约和可用性。
>
> - 您可以使用预测性扩展来计算基准容量，使用动态扩展来处理额外的容量。动态扩展独立于预测性扩展，会根据当前的利用率向内和向外扩展。首先，Amazon A EC2 uto Scaling 会计算每个动态扩展策略的推荐实例数量。然后，它会根据提供最多实例的策略进行扩展。

- 要允许在负载减少时进行横向缩减，您的自动扩缩组应始终至少有一个启用了横向缩减部分的动态扩展策略。
- 您可以通过确保最小和最大容量不太严格来提高扩展性能。如果策略中包含的推荐实例数不在最小和最大容量范围内，则将阻止横向缩减和横向扩展。

使用监控预测性扩展指标 CloudWatch

根据您的需求，您可能更愿意从亚马逊 CloudWatch 而不是从 Amazon A EC2 uto Scaling 控制台访问用于预测性扩展的监控数据。创建预测性扩缩策略后，该策略将收集用于预测未来负载和容量的数据。收集这些数据后，系统会定期自动将其存储。 CloudWatch 然后，您可以使用可视 CloudWatch 化策略在一段时间内的执行情况。您还可以创建 CloudWatch 警报，以便在绩效指标变化超出您在中定义的限制时通知您 CloudWatch。

主题

可视化显示历史预测数据

您可以在中查看预测性扩展策略的负载和容量预测数据 CloudWatch。在单个图表中根据其他 CloudWatch指标对预测进行可视化时，这可能很有用。您还可以查看更大的时间范围，以了解长期的趋势，这也非常有益。您可以访问长达 15 个月的历史指标，以更好地了解您的策略性能。

有关更多信息，请参阅 预测性扩缩指标和维度。

使用 CloudWatch 控制台查看历史预测数据

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 Metrics（指标），然后选择 All metrics（所有指标）。
3. 选择 Auto Scaling（自动扩缩）指标命名空间。
4. 选择下面的一个选项，以查看负载预测或容量预测指标：
   - Predictive Scaling Load Forecasts（预测性扩缩负载预测）
   - Predictive Scaling Capacity Forecasts（预测性扩缩容量预测）
5. 在搜索字段中，输入预测性扩缩策略的名称或自动扩缩组的名称，然后按 Enter 键以筛选结果。

6. 要为指标绘制图表，请选中该指标旁的复选框。要更改图表的名称，请选择铅笔图标。要更改时间范围，请选择某个预定义的值或选择 custom。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的绘制指标图表。
7. 要更改统计数据，请选择 Graphed metrics（已绘制图表指标）选项卡。选择列标题或单个值，然后选择其他统计数据。尽管您可以为每个指标选择任何统计数据，但并非所有统计数据都对PredictiveScalingLoadForecast和PredictiveScalingCapacityForecast指标有用。例如，Average（平均）、Minimum（最小）和 Maximum（最大）统计数据非常有用，但 Sum（总和）统计数据用处不大。
8. 要在图表中添加其他指标，请在 All（全部）下选择 Browse（浏览），找到特定的指标，然后选中它旁边的复选框。您最多可以添加 10 个指标。

   例如，要将 CPU 利用率的实际值添加到图表中，请选择EC2命名空间，然后选择 B y Auto Scaling Group。然后，选中该CPUUtilization指标和特定的 Auto Scaling 组对应的复选框。
9. （可选）要将图表添加到 CloudWatch 仪表板，请选择操作，然后选择添加到仪表板。

使用指标数学创建准确度指标

使用指标数学，您可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的时间序列。您可以在 CloudWatch 控制台上可视化生成的时间序列并将其添加到仪表板中。有关指标数学的更多信息，请参阅 Amazon CloudWatch 用户指南中的使用指标数学。

使用指标数学，您可以用不同的方式绘制 Amazon A EC2 uto Scaling 为预测性扩展生成的数据。这可帮助您监控随时间变化的策略性能，并帮助您了解是否可以改进指标组合。

例如，您可以使用指标数学表达式来监控平均绝对百分比误差（MAPE）。MAPE 指标可帮助监控在给定预测时段内预测值与实际观测值之间的差异。MAPE 值的变化可能表明随着应用性质的变化，策略的性能是否会随着时间的推移而下降。MAPE 增加说明着预测值和实际值之间的差异加大。

示例：指标数学表达式

要开始使用此类图表，您可以创建一个与下例中类似的指标数学表达式。

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
```

```
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "m2",
      "Label": "ForecastedLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/AutoScaling",
          "MetricName": "PredictiveScalingLoadForecast",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            },
            {
              "Name": "PolicyName",
              "Value": "my-predictive-scaling-policy"
            },
            {
              "Name": "PairIndex",
              "Value": "0"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Average"
```

```
      },
      "ReturnData": false
    }
  ]
}
```

这不是单个指标，而是一组针对 MetricDataQueries 的指标数据查询结构。MetricDataQueries 中的每一项都会获取一个指标或执行一个数学表达式。第一项 e1 是一个数学表达式。指定的表达式将 ReturnData 参数设置为 true，这最终会生成单个时间序列。对于所有其他指标，ReturnData 值为 false。

在示例中，指定的表达式使用实际值和预测值作为输入，并返回新的指标 (MAPE)。 m1是包含实际负载值的 CloudWatch指标（假设 CPU 利用率是最初为名为的策略指定的负载指标my-predictive-scaling-policy）。 m2是包含预测负荷值的 CloudWatch指标。MAPE 指标的数学语法如下所示：

(((实际值 - 预测值)/(实际值)) 的绝对值) 的平均值

可视化显示准确度指标并设置警报

要可视化准确度指标数据，请在 CloudWatch 控制台中选择 Metrics 选项卡。您可以在此处绘制数据图表。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的向 CloudWatch 图表添加数学表达式。

您还可以在 Metrics（指标）部分为您监控的指标设置警报。在 Graphed metrics（绘制的指标）选项卡中，选择 Actions（操作）列下的 Create alarm（创建警报）。Create alarm（创建警报）图标用一个小铃铛表示。有关更多信息和通知选项，请参阅《Amazon 用户指南》中的基于指标数学表达式创建警报和通知 CloudWatch 用户警报更改。 CloudWatch

或者，您可以使用GetMetricData和PutMetricAlarm使用公制数学进行计算，并根据输出创建警报。

使用计划操作覆盖预测值

有时，您可能会获得有关未来应用程序需求的其他信息，预测计算无法考虑这些信息。例如，预测计算可能会低估即将举行的市场营销活动所需的容量。您可以使用计划操作在未来时段内临时覆盖预测。计划操作可以循环运行，也可以在出现一次性需求波动的特定日期和时间运行。

例如，您可以创建具有高于预测容量的最小容量的计划操作。在运行时，Amazon A EC2 uto Scaling 会更新您的 Auto Scaling 组的最小容量。由于预测式扩展可针对容量进行优化，因此执行最小容量高于预测值的计划操作。这样可以防止容量低于预期。要停止覆盖预测，请使用第二个计划操作将最小容量恢复到其原始设置。

以下过程概述了在将来期间覆盖预测的步骤。

主题

> **Important**
>
> 本主题假设您尝试覆盖预测，以扩展到比预测更高的容量。如果您需要在不受预测性扩展策略干扰的情况下暂时减少容量，则请改用仅预测模式。在仅预测模式下，预测性扩展将继续生成预测，但不会自动增加容量。然后，您可以监控资源利用率，并根据需要手动缩减组大小。有关手动扩缩的更多信息，请参阅Amazon A EC2 uto Scaling 的手动扩展。

步骤 1：（可选）分析时间序列数据

首先分析预测时间序列数据。这是一个可选步骤，但如果您想了解预测的详细信息，它会很有帮助。

1. 检索预测

   创建预测后，您可以查询预测中的特定时间段。查询的目的是获得特定时间段的时间序列数据的完整视图。

   您的查询最多可以包含两天的未来预测数据。如果您已经使用了一段时间预测式扩展，您还可以访问过去的预测数据。但是，开始和结束时间之间的最长持续时间为 30 天。

   要使用命令获取预测，get-predictive-scaling-forecast Amazon CLI 请在命令中提供以下参数：

   - 在 `--auto-scaling-group-name` 参数中输入 Auto Scaling 组的名称。
   - 在 `--policy-name` 参数中输入策略的名称。
   - 在 `--start-time` 参数中输入开始时间以仅返回在指定时间或之后的预测数据。
   - 在 `--end-time` 参数中输入结束时间以仅返回在指定时间之前的预测数据。

   ```
   aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \
       --policy-name cpu40-predictive-scaling-policy \
       --start-time "2021-05-19T17:00:00Z" \
       --end-time "2021-05-19T23:00:00Z"
   ```

   如果成功，该命令将返回类似于以下示例的数据。

```
{
    "LoadForecast": [
        {
            "Timestamps": [
                "2021-05-19T17:00:00+00:00",
                "2021-05-19T18:00:00+00:00",
                "2021-05-19T19:00:00+00:00",
                "2021-05-19T20:00:00+00:00",
                "2021-05-19T21:00:00+00:00",
                "2021-05-19T22:00:00+00:00",
                "2021-05-19T23:00:00+00:00"
            ],
            "Values": [
                153.0655799339254,
                128.8288551285919,
                107.1179447150675,
                197.3601844551528,
                626.4039934516954,
                596.9441277518481,
                677.9675713779869
            ],
            "MetricSpecification": {
                "TargetValue": 40.0,
                "PredefinedMetricPairSpecification": {
                    "PredefinedMetricType": "ASGCPUUtilization"
                }
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
```

```
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

此响应包括两个预测：LoadForecast 和 CapacityForecast。LoadForecast 显示每小时负载预测。CapacityForecast 显示每小时处理预测负载所需的容量的预测值，同时保持 TargetValue 为 40.0（平均 CPU 利用率为 40%）。

2. 确定目标时间段

   确定应发生一次性需求波动的小时数。请记住，预测中显示的日期和时间以 UTC 为单位。

步骤 2：创建两个计划操作

接下来，在应用程序的负载高于预测负载的特定时间段内创建两个计划操作。例如，如果您的营销活动会在有限时间段内使网站的流量增加，则可计划一个一次性操作以在其启动时更新最小容量。然后，安排另一个操作，以便在事件结束时将最小容量返回到原始设置。

为一次性事件创建两个计划操作（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Automatic scaling（自动扩展）选项卡上的 Scheduled actions（计划操作）中，选择 Create scheduled action（创建计划操作）。
4. 填写以下计划操作设置：

   a. 为计划操作输入名称。

   b. 对于最小值，输入 Auto Scaling 组的最小新容量。最小值必须小于或等于组的最大大小。如果最小值大于该组的最大大小，则必须更新最大值。

   c. 对于 Recurrence (重复次数)，选择 Once (一次)。

   d. 对于时区，请选择时区。如果未选择任何时区，预设情况下使用 ETC/UTC。

e. 定义特定开始时间。

5. 选择创建。

   控制台将显示 Auto Scaling 组的计划操作。

6. 配置第二个计划操作，以在事件结束时将最小容量返回原始设置。预测式扩展只能在设置最小值小于预测值时扩展容量。

为一次性事件创建两个计划操作 (Amazon CLI)

要使用创建计划操作，请使用 put-scheduled-update-group-action 命令。 Amazon CLI

例如，让我们定义一个时间表，在 5 月 19 日下午 5:00 时保持最少三个实例的容量，持续 8 小时。以下命令显示如何实现此方案。

第一个put-scheduled-update-group操作命令指示 Amazon A EC2 uto Scaling 在世界标准时间 2021 年 5 月 19 日下午 5:00 更新指定 Auto Scaling 组的最小容量。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-
start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-
capacity 3
```

第二个命令指示 Amazon A EC2 uto Scaling 在世界标准时间 2021 年 5 月 20 日凌晨 1:00 将群组的最小容量设置为 1。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end
 \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-
capacity 1
```

将这些计划操作添加到 Auto Scaling 组后，Amazon A EC2 uto Scaling 会执行以下操作：

- 2021 年 5 月 19 日下午 5:00，第一个计划的操作将运行。如果组中当前已少于三个实例，则该组会扩展到三个实例。在这段时间和接下来的八小时内，如果预测的容量高于实际容量，或者动态扩展策略生效，Amazon A EC2 uto Scaling 可以继续扩展。
- 2021 年 5 月 20 日上午 1:00，将运行第二个计划的操作。这将在事件结束时将最小容量恢复为其原始设置。

根据重复性计划进行扩展

要覆盖每周相同时间段的预测，请创建两个计划操作，并使用 cron 表达式提供时间和日期逻辑。

此 cron 表达式格式包含五个空格分隔的字段：[Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]。字段可以包含任何允许的值，包括特殊字符。

例如，下面的 cron 表达式在每周二上午 6:30 运行操作。星号用作通配符，以匹配字段的所有值。

```
30 6 * * 2
```

另请参阅

有关如何创建、列出、编辑和删除计划操作的更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。

使用自定义指标的高级预测性扩展策略

在预测性扩展策略中，您可以使用预定义指标或自定义指标。当预定义指标（CPU、网络 I/O 和 Application Load Balancer 请求计数）未充分描述应用程序负载时，自定义指标非常有用。

使用自定义指标创建预测性扩展策略时，您可以指定由提供的其他 CloudWatch 指标 Amazon，也可以指定自己定义和发布的指标。您还可以使用公制数学来汇总现有指标并将其转换为 Amazon 不会自动跟踪的新时间序列。例如，通过计算新的总和或平均值来组合数据中的值时，该操作称为执行聚合。生成的数据称为聚合。

以下部分包含了有关如何为构造策略的 JSON 结构的最佳实践和示例。

主题

- 最佳实践
- 先决条件
- 构造自定义指标的 JSON
- 预测性扩展策略中自定义指标的注意事项
- 限制

最佳实践

以下最佳实践可帮助您更有效地使用自定义指标：

- 对于负载指标规范，最有用的指标是作为一个整体表示 Auto Scaling 组负载的指标，而不管该组的容量如何。
- 对于扩展指标规范，要扩展的最有用指标是每个实例的平均吞吐量或利用率指标。
- 扩展指标必须与容量成反比。也就是说，如果 Auto Scaling 组中的实例数量增加，则扩展指标应该减少大致相同的比例。为确保预测性扩展按预期采取行动，负载指标和扩展指标还必须彼此之间密切关联。
- 目标利用率必须与扩展指标的类型匹配。对于使用 CPU 利用率的策略配置，这是目标百分比。对于使用吞吐量（例如请求数或消息数）的策略配置，这是在任何一分钟间隔内每个实例的目标请求数或目标消息数。
- 如果未遵循这些建议，那么时间序列的预测未来值可能会不正确。要验证数据是否正确，您可以在 Amazon A EC2 uto Scaling 控制台中查看预测值。或者，在创建预测性扩展策略后，检查调用 GetPredictiveScalingForecastAPI 返回的`LoadForecast`和`CapacityForecast`对象。
- 我们强烈建议您在仅预测模式下配置预测式扩展，以便在预测式扩展开始主动扩展容量之前对预测进行评估。

先决条件

要将自定义指标添加到预测性扩缩策略，您必须具有 `cloudwatch:GetMetricData` 权限。

要指定您自己的指标而不是 Amazon 提供的指标，您必须先将指标发布到 CloudWatch。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的发布自定义指标。

如果发布自己的指标，请确保以至少五分钟的频率发布数据点。Amazon A EC2 uto Scaling 会 CloudWatch 根据所需的周期长度从中检索数据点。例如，负载指标规范使用每小时指标来衡量应用程序的负载。 CloudWatch 使用您发布的指标数据，通过将所有数据点与每个一小时内的时间戳聚合在一起，为任何一小时的时间段提供单个数据值。

构造自定义指标的 JSON

以下部分包含有关如何配置预测缩放以从中查询数据的示例 CloudWatch。配置此选项有两种不同的方法，您选择的方法会影响您为预测性扩缩策略构造 JSON 时使用的格式。使用指标数学时，JSON 格式会根据所执行的指标数学进一步变化。

1. 要创建可直接从提供的其他 CloudWatch 指标 Amazon 或您发布到的指标中获取数据的策略 CloudWatch，请参阅包含自定义负载和扩缩指标的预测性扩缩策略示例（Amazon CLI）。
2. 要创建可查询多个 CloudWatch 指标并使用数学表达式根据这些指标创建新时间序列的策略，请参阅使用指标数学表达式。

包含自定义负载和扩缩指标的预测性扩缩策略示例（Amazon CLI）

要使用创建带有自定义负载和扩展指标的预测性扩展策略 Amazon CLI，请将的参数存储`--predictive-scaling-configuration`在名为的 JSON 文件中`config.json`。

您可以将以下示例中的可替换值替换为您的指标和目标利用率值，从而开始添加自定义指标。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
```

```
                ]
              },
              "Stat": "Sum"
            }
          }
        ]
      }
    }
  ]
}
```

有关更多信息，请参阅MetricDataQuery《Amazon A EC2 uto Scaling API 参考》。

> **Note**
>
> 以下是一些其他资源，可以帮助您查找指标名称、命名空间、维度和指标 CloudWatch 统计信息：
>
> - 有关 Amazon 服务的可用指标的信息，请参阅《Amazon CloudWatch 用户指南》中发布 CloudWatch 指标的Amazon 服务。
> - 要使用获取指标的确切指标名称、命名空间和维度（如果适用） Amazon CLI，请参阅列表 CloudWatch 指标。

要创建此策略，请使用 JSON 文件作为输入运行put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
}
```

使用指标数学表达式

以下部分提供了预测性扩缩策略的信息和示例，这些示例演示了如何在策略中使用指标数学。

主题

了解指标数学

如果您只想汇总现有指标数据，那么公 CloudWatch 制数学可以为您节省向其发布另一个指标的工作量和成本 CloudWatch。您可以使用 Amazon 提供的任何指标，也可以使用在应用程序中定义的指标。例如，您可能想要计算每个实例的 Amazon SQS 队列积压。您可以从队列中获取用于检索的可用消息的大约数量，然后将该数量除以 Auto Scaling 组的运行容量来实现这一点。

有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用公制数学。

如果您选择在预测性扩展策略中使用指标数学表达式，请考虑以下几点：

- 指标数学运算使用指标名称、命名空间和维度键/值对指标的唯一组合的数据点。
- 您可以使用任何算术运算符 (+-*/^)、统计函数（例如 AVG 或 SUM）或其他支持的函数。CloudWatch
- 您可以在数学表达式的公式中同时使用指标和其他数学表达式的结果。
- 指标数学表达式可以由不同的聚合组成。但是，得到最终聚合结果的最佳实践是针对扩展指标使用 `Average` 以及针对负载指标使用 `Sum`。
- 指标规范中使用的任何表达式最终都必须返回一个单个时间序列。

要使用指标数学，请执行以下操作：

- 选择一个或多个 CloudWatch 指标。然后，创建表达式。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用公制数学。
- 使用 CloudWatch控制台或 CloudWatch GetMetricDataAPI 验证指标数学表达式是否有效。

使用指标数学组合指标的预测性扩缩策略示例（Amazon CLI）

有时，您可能需要首先以某种方式处理其数据，而不是直接指定指标。例如，您可能有一个从 Amazon SQS 队列中提取工作的应用程序，并且可能希望使用队列中的项目数作为预测性扩展的标准。队列中的消息数不仅仅定义您需要的实例数。因此，需要执行更多工作来创建可用于计算每个实例的积压的指标。有关更多信息，请参阅 基于 Amazon SQS 的扩缩策略。

以下示例是适用于此场景的预测扩展策略示例。它指定了基于 Amazon SQS `ApproximateNumberOfMessagesVisible` 指标的扩展和负载指标，即可从队列中获取的用于检索的消息数量。它还使用 Amazon A EC2 uto Scaling `GroupInServiceInstances` 指标和数学表达式来计算扩展指标中每个实例的待办事项数量。

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
```

```
                    "Value": "my-asg"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
              },
              "Stat": "Sum"
            },
            "ReturnData": true
          }
        ]
      }
    }
  ]
}
```

该示例返回策略的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

在蓝/绿部署场景中使用的预测扩缩策略示例（Amazon CLI）

搜索表达式提供了一个高级选项，您可以在其中查询多个 Auto Scaling 组中的指标并对其执行数学表达式。此选项对于蓝/绿部署尤其有效。

> **Note**
>
> 蓝/绿部署是一种部署方法，您可以在其中创建两个独立但相同的 Auto Scaling 组。只有其中一个组接收生产流量。用户流量最初定向到较早的Auto Scaling 组（“蓝色”），而新组（“绿色”）用于测试和评估应用程序或服务的新版本。测试并接受新部署后，用户流量将转移到“绿色”的 Auto Scaling 组。然后，您可以在部署成功后删除“蓝色”组。

作为蓝/绿部署的一部分创建新的 Auto Scaling 组时，每个组的指标历史记录可以自动包含在预测性扩展策略中，而无需更改其指标规范。有关更多信息，请参阅 C Amazon omp EC2 ute 博客上的 “将 Auto Scaling 预测性扩展策略用于蓝/绿部署”。

以下示例策略说明了如何执行此操作。在此示例中，该策略使用了 Amazon EC2 发布的`CPUUtilization`指标。它使用 Amazon A EC2 uto Scaling `GroupInServiceInstances` 指标和数学表达式来计算每个实例的扩展指标的值。它还指定了一个容量指标规范来获取 `GroupInServiceInstances` 指标。

根据指定的搜索条件，搜索表达式查找多个 Auto Scaling 组中实例的 `CPUUtilization`。如果您稍后创建了匹配相同搜索条件的新 Auto Scaling 组，则自动包含新 Auto Scaling 组中实例的 `CPUUtilization`。

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
```

```
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
          }
        ]
      },
      "CustomizedCapacityMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
          }
        ]
      }
    }
  ]
}
```

该示例返回策略的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-scaling-policy",
  "Alarms": []
}
```

预测性扩展策略中自定义指标的注意事项

如果在使用自定义指标时出现问题，建议您执行以下操作：

- 如果提供了错误消息，请阅读该消息并解决其报告的问题（如果可能）。
- 如果在蓝/绿部署方案中尝试使用搜索表达式时出现问题，请首先确保您了解如何创建查找部分匹配而非完全匹配的搜索表达式。此外，请检查您的查询是否只查找运行特定应用程序的 Auto Scaling 组。有关搜索表达式语法的更多信息，请参阅 Amazon CloudWatch 用户指南中的CloudWatch 搜索表达式语法。
- 如果您未事先验证表达式，则该put-scaling-policy命令会在您创建扩展策略时对其进行验证。但是，此命令有可能无法识别所检测错误的确切原因。要修复这些问题，请对您在get-metric-data命令请求的响应中收到的错误进行故障排除。您也可以从 CloudWatch 控制台对表达式进行故障排除。
- 查看控制台中的 Load (负载) 和 Capacity (容量) 图表时，Capacity (容量) 图表可能不显示任何数据。为确保图表具有完整的数据，请确保始终为 Auto Scaling 组启用组指标。有关更多信息，请参阅 启用 Auto Scaling 组指标（控制台）。
- 只有具备在其生命周期内于不同的 Auto Scaling 组中运行的应用程序时，容量指标规范才适用于蓝色/绿色部署。此自定义指标允许您提供多个 Auto Scaling 组的总容量。预测性扩展使用此功能在控制台中的 Capacity (容量) 图表内显示历史数据。
- 如果 `MetricDataQueries` 自行指定 SEARCH () 函数，而没有像 SUM () 这样的数学函数，则必须为 `ReturnData` 指定 `false`。原因在于搜索表达式可能返回多个时间序列，而基于表达式的指标规范仅可以返回一个时间序列。
- 搜索表达式中涉及的所有指标均应该具有相同的分辨率。

限制

- 您可以在一个指标规范中查询最多 10 个指标的数据点。
- 为满足此限制，一个表达式算作一个指标。

控制在横向缩减过程中要终止的 Auto Scaling 实例

Amazon A EC2 uto Scaling 使用终止策略来决定终止实例的顺序。您可以使用预定义策略或创建自定义策略以满足特定要求。通过使用自定义策略或实例横向缩减保护，您还可以防止自动扩缩组终止尚未准备好终止的实例。

内容

当 Amazon A EC2 uto Scaling 使用终止政策时

以下各节介绍了 Amazon A EC2 uto Scaling 使用终止策略的场景。

内容

横向缩减事件

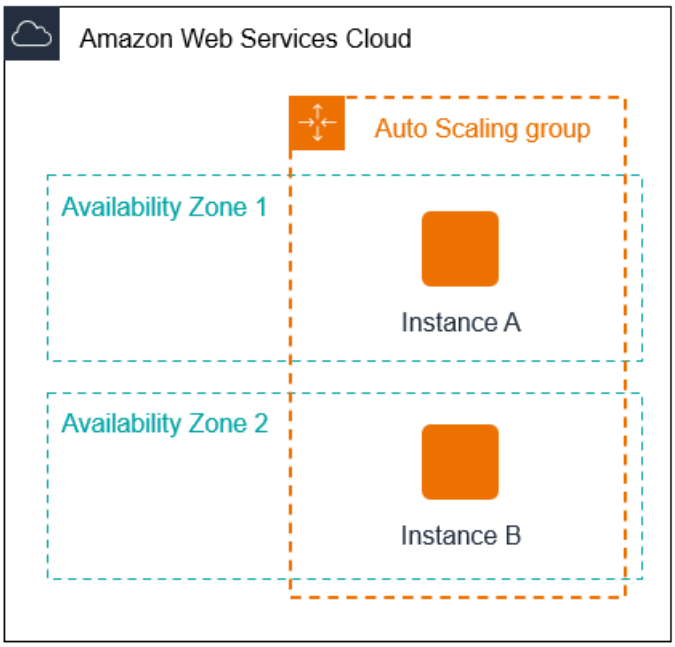
当自动扩缩组的所需容量有一个低于该组当前容量的新值时，就会发生横向缩减事件。

缩减事件发生在以下场景中：

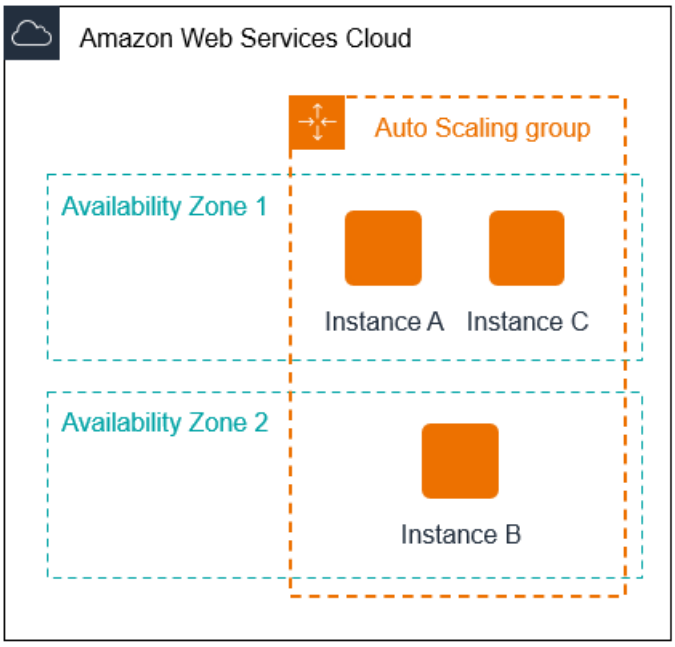
- 使用动态扩展策略时，组的大小会因指标值的更改而减小
- 当使用计划的扩展时，组的大小由于计划的操作而减小
- 当手动减小组的大小

以下示例显示在发生横向缩减事件时终止策略的工作原理。

1. 例如，假设您有一个包含一个实例类型、两个可用区和两个实例所需容量的 Auto Scaling group 组。它还具有一个动态扩展策略，可在资源利用率增加或减少时添加和删除实例。该组中的两个实例分布在两个可用区之间，如下图所示。

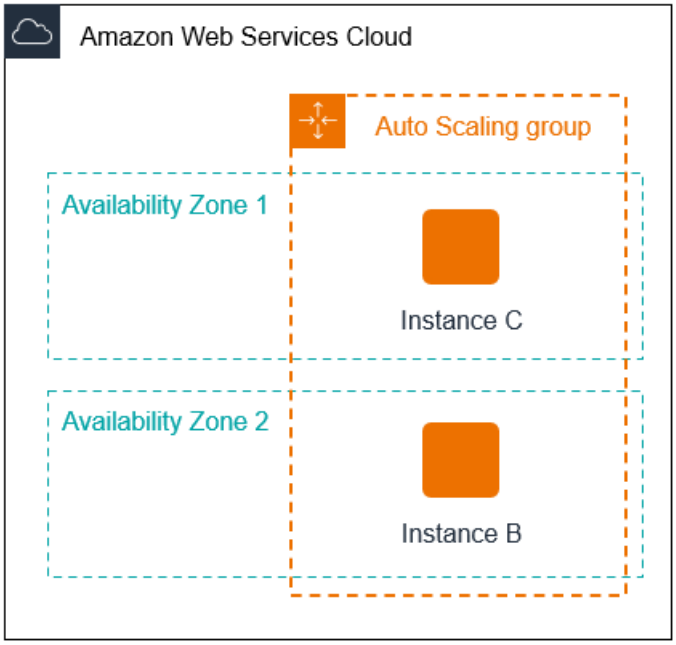


2. 当 Auto Scaling 组扩展时，Amazon A EC2 uto Scaling 会启动一个新实例。Auto Scaling 组现在有三个实例，分布在两个可用区，如下图所示。



3. 当 Auto Scaling 组缩减规模时，Amazon A EC2 uto Scaling 会终止其中一个实例。
4. 如果您没有为该群组分配特定的终止策略，Amazon A EC2 uto Scaling 将使用默认的终止策略。该策略会选择具有两个实例的可用区，并终止从启动配置、不同启动模板或最旧版本的当前启动模板

启动的实例。如果实例是从相同的启动模板和版本启动的，Amazon A EC2 uto Scaling 会选择最接近下一个账单时间的实例并将其终止。



实例刷新

您可以启动实例刷新以更新自动扩缩组中的实例。在实例刷新期间，Amazon A EC2 uto Scaling 会终止组中的实例，然后为已终止的实例启动替换实例。Auto Scaling 组的终止策略控制首先替换哪些实例。

重新平衡可用区

Amazon A EC2 uto Scaling 在为你的 Auto Scaling 组启用的可用区中均衡你的容量。这有助于减少可用区中断的影响。如果可用区间的容量分配失衡，Amazon A EC2 uto Scaling 会重新平衡 Auto Scaling 组，方法是在已启用的可用区中启动实例最少的实例，并在其他地方终止实例。终止策略控制哪些实例优先终止。

跨可用区的实例分布可能失去平衡的原因有很多。

移除实例

如果您从 Auto Scaling 组中分离实例，将实例置于备用状态或者明确终止实例并减少所需容量，从而阻止启动替换实例，则该组可能会变得不平衡。如果发生这种情况，Amazon A EC2 uto Scaling 会通过重新平衡可用区域来进行补偿。

使用不同于最初指定的可用区

如果您将 Auto Scaling 组扩展为包括其他可用区，或者您更改了使用的可用区，Amazon A EC2 uto Scaling 会在新的可用区域中启动实例，并终止其他区域中的实例，以帮助确保您的 Auto Scaling 组均匀地跨越可用区。

可用性中断

可用性中断的情况很少发生。但是，如果某个可用区变得不可用并稍后恢复，则您的 Auto Scaling 组可能会在可用区之间变得不均衡。Amazon A EC2 uto Scaling 会尝试逐步重新平衡该组，而重新平衡可能会终止其他区域中的实例。

例如，假设您有一个包含一个实例类型、两个可用区和两个实例所需容量的 Auto Scaling 组。在一个可用区出现故障的情况下，Amazon A EC2 uto Scaling 会自动在运行正常的可用区中启动一个新实例，以取代运行状况不佳的可用区中的实例。然后，当运行状况不佳的可用区稍后恢复到正常状态时，Amazon A EC2 uto Scaling 会自动在该区域启动一个新实例，这反过来又会终止未受影响区域中的实例。

> **Note**
>
> 重新平衡时，Amazon A EC2 uto Scaling 会在终止旧实例之前启动新实例，这样再平衡就不会影响应用程序的性能或可用性。
> 由于 Amazon A EC2 uto Scaling 会在终止旧实例之前尝试启动新实例，因此达到或接近指定的最大容量可能会阻碍或完全停止再平衡活动。为避免此问题，在再平衡活动期间，系统可以暂时超出某组的指定最大容量的 10%（或 1 个实例边缘，以较大者为准）。仅当该组达到或接近最大容量，并需要重新平衡时，才可超出容量限制；此类情况的原因是用户请求重新分区，或者是为了弥补区域可用性问题。该超出状态仅持续重新平衡该组所需的时间。

为 Amazon A EC2 uto Scaling 配置终止策略

终止政策提供了 Amazon A EC2 uto Scaling 在按特定顺序终止实例时所遵循的标准。默认情况下，Amazon A EC2 uto Scaling 使用终止策略，该策略旨在首先终止使用过时配置的实例。您可以更改终止策略以控制哪些实例最需要优先终止。

当 Amazon A EC2 uto Scaling 终止实例时，它会尝试在为你的 Auto Scaling 组启用的可用区之间保持平衡。保持区域平衡优先于终止策略。如果一个可用区的实例数多于其他可用区，则 Amazon A EC2 uto Scaling 会先将终止策略应用于不平衡的区域。如果可用区保持平衡，则会在所有区域中应用终止策略。

主题

默认终止策略的工作原理

当 Amazon A EC2 uto Scaling 需要终止一个实例时，它会首先确定哪个可用区（一个或多个区域）的实例最多，并且至少有一个实例不受缩容保护。然后，它继续评估已确定的可用区内未受保护的实例，如下所示：

使用过时配置的实例

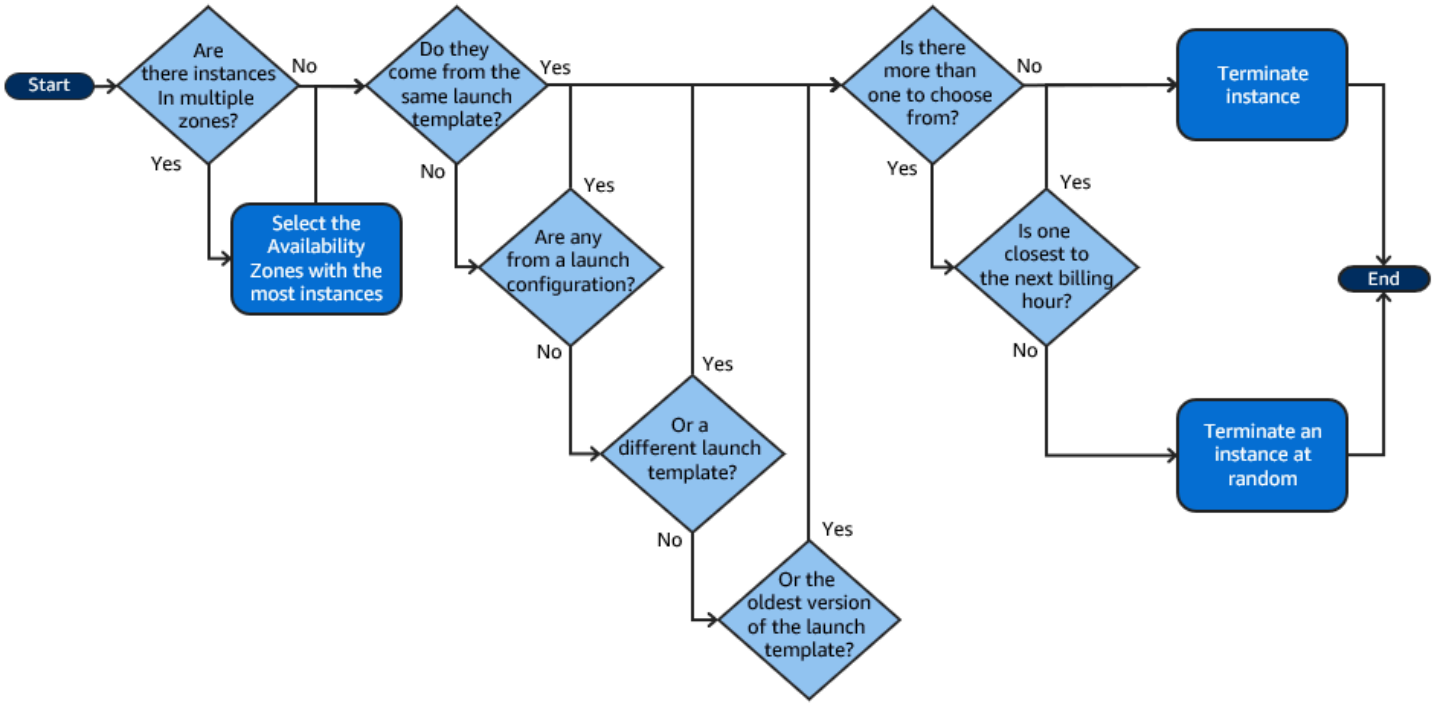
- 对于使用启动模板的组：确定任何实例是否使用过时的配置，按照以下顺序确定优先级：
  1. 首先，检查实例是否使用启动配置启动。
  2. 然后，检查实例是否使用其他启动模板而不是当前启动模板启动。
  3. 最后，检查实例是否使用当前启动模板的最旧版本。
- 对于使用启动配置的组：确定是否有任何实例使用最旧的启动配置。

如果找不到配置过时的实例，或者有多个实例可供选择，Amazon A EC2 uto Scaling 会考虑下一个标准，即实例接近下一个计费时间。

接近下一个计费小时的实例

确定符合先前标准的任何实例是否最接近下一个计费小时。如果多个实例同样接近，则随机终止一个实例。这有助于最大限度使用按小时计费的实例。但是，现在大多数 EC2 使用量按秒计费，因此这种优化提供的好处较少。有关更多信息，请参阅 Amazon EC2 定价。

以下流程图说明默认终止策略如何适用于使用启动模板的组。



默认终止策略和混合实例组

在终止混合实例组中的实例时，Amazon A EC2 uto Scaling 会应用额外的标准。

当 Amazon A EC2 uto Scaling 需要终止实例时，它会首先根据群组的设置确定应终止哪个购买选项（竞价或按需）。这样可以确保该组内的竞价型实例和按需型实例随着时间的推移倾向于指定的比率。

然后，其将在每个可用区内独立应用终止策略。其可确定终止哪个可用区中的哪个竞价型实例或按需型实例，以保持可用区平衡。同样的逻辑适用于为实例类型定义了权重的混合实例组。

在每个区域内，默认终止策略的工作原理如下，用于确定可以终止购买选项中哪些不受保护的实例：

1. 确定是否可以终止任何实例，以提高与自动扩缩组指定分配策略的一致性。如果没有确定要优化的实例，或者有多个实例可供选择，则会继续评估。
2. 确定任何实例是否使用过时的配置，按照以下顺序确定优先级：
   a. 首先，检查实例是否使用启动配置启动。
   b. 然后，检查实例是否使用其他启动模板而不是当前启动模板启动。
   c. 最后，检查实例是否使用当前启动模板的最旧版本。

   如果未找到有过时配置的实例，或者有多个实例可供选择，则会继续评估。
3. 确定是否有任何实例最接近下一个计费小时。如果多个实例同样接近，则随机选择一个实例。

预定义的终止策略

您可以从以下预定义的终止策略中进行选择：

- **`Default`**：根据默认终止策略终止实例。
- **`AllocationStrategy`**：终止自动扩缩组中的实例，使剩余实例与所终止实例类型（竞价型实例或按需型实例）的分配策略匹配。当您首选的实例类型发生变化时，可使用该策略。如果竞价分配策略是 `lowest-price`，则您可以逐渐在 N 个最低价竞价池中重新平衡分布竞价型 实例。如果竞价分配策略是 `capacity-optimized`，则您可以逐渐重新平衡具有更多可用竞价容量的竞价池之间 竞价型实例的分布。您也可以使用较高优先级类型的按需实例逐渐替代较低优先级类型的按需实例。
- **`OldestLaunchTemplate`**：终止采用最旧启动模板的实例。利用此策略，会首先终止使用非当前启动模板的实例，然后终止使用当前启动模板的最旧版本的实例。如果要更新某个组并且逐步淘汰先前配置中的实例时，此策略非常有用。
- **`OldestLaunchConfiguration`**：终止采用最旧启动配置的实例。如果要更新某个组并且逐步淘汰先前配置中的实例时，此策略非常有用。使用此策略，使用非当前启动配置的实例将首先终止。
- **`ClosestToNextInstanceHour`**：终止最接近下个计费小时的实例。此策略有助于最大限度使用按小时收费的实例。
- **`NewestInstance`**：终止组中最新的实例。如果要测试新的启动配置但不想在生产中保留它时，此策略非常有用。
- **`OldestInstance`**：终止组中最旧的实例。当您将 Auto Scaling 组中的实例升级到新的 EC2 实例类型时，此选项非常有用。您可以逐渐将较旧类型的实例替换为较新类型的实例。

> **Note**
>
> 无论使用哪种终止策略，Amazon A EC2 uto Scaling 始终首先在可用区域之间平衡实例。因此，您可能会遇到一些较新的实例在旧实例之前终止的情况。例如，当某个可用区的实例数多于该组使用的其他可用区时，或某个可用区的实例数多于该组使用的其他可用区。

更改自动扩缩组的终止策略

要更改自动扩缩组的终止策略，请使用以下方法之一。

Console

最初在 Amazon A EC2 uto Scaling 控制台中创建 Auto Scaling 组时，您无法更改终止策略。默认终止策略被自动使用。创建自动扩缩组后，您可以将默认策略替换为不同的终止策略或按其应有的应用顺序列出的多个终止策略。

更改自动扩缩组的终止策略

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
4. 对于 Termination policies（终止策略），请选择一个或多个终止策略。如果您选择多个策略，请按照您希望评估策略的顺序将其列出。

   您还可选择 Custom termination policy（自定义终止策略），然后选择一个满足您需求的 Lambda 函数。如果您已经为 Lambda 函数创建了版本和别名，则可以从 Version/Alias（版本/别名）下拉列表选择版本或别名。要使用您的 Lambda 函数的未发布版本，请保留 Version/Alias（版本/别名）设置为默认值。有关更多信息，请参阅 了解使用 Lambda 创建自定义终止策略。。

   > **Note**
   >
   > 使用多个策略时，必须正确设置它们的顺序：
   >
   > - 如果您使用 Default（默认）策略，它必须是列表中的最后一个策略。
   > - 如果您使用 Custom termination policy（自定义终止策略），它必须是列表中的第一个策略。
5. 选择 Update（更新）。

Amazon CLI

自动使用默认终止策略，除非指定了不同的策略。

更改自动扩缩组的终止策略

使用以下命令之一：

- create-auto-scaling-group
- update-auto-scaling-group

您可以单独使用终止策略，或者将它们合并到策略列表中。例如，使用以下命令更新 Auto Scaling 组以首先使用 `OldestLaunchConfiguration` 策略，然后使用 `ClosestToNextInstanceHour` 策略。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

如果您使用 `Default` 终止策略，请将该策略设为终止策略列表中的最后一项。例如，`--termination-policies "OldestLaunchConfiguration" "Default"`。

要使用自定义终止政策，必须先使用创建终止政策 Amazon Lambda。要指定要用作终止策略的 Lambda 函数，请将其设为终止策略列表中的第一个函数。例如，`--termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration"`。有关更多信息，请参阅 了解使用 Lambda 创建自定义终止策略。。

了解使用 Lambda 创建自定义终止策略。

在缩小 A EC2 uto Scaling 组的大小（称为缩小）时，Amazon Auto Scaling 使用终止策略来优先终止哪些实例。Auto Scaling 组使用默认终止策略，但您可以选择或创建您自己的终止策略。有关选择预定义终止策略的更多信息，请参阅 为 Amazon A EC2 uto Scaling 配置终止策略。

在本主题中，您将学习如何使用 Amazon A EC2 uto Scaling 为响应某些事件而调用的 Amazon Lambda 函数来创建自定义终止策略。您创建的 Lambda 函数会处理 Amazon A EC2 uto Scaling 发送的输入数据中的信息，并返回准备终止的实例列表。

自定义终止策略可以更好地控制终止哪些实例以及何时终止。例如，当您的 Auto Scaling 组缩小规模时，Amazon A EC2 uto Scaling 无法确定是否有不应中断的工作负载在运行。使用 Lambda 函数，您可以验证终止请求，等到工作负载完成后再将实例 ID 返回给 Amazon A EC2 uto Scaling 进行终止。

内容

- 注意事项
- 创建 Lambda 函数
- 限制

输入数据

Amazon A EC2 uto Scaling 会为缩放事件生成 JSON 有效负载，当实例由于最长实例生命周期或实例刷新功能而即将终止时，也会生成一个 JSON 有效负载。它还为在跨可用区重新平衡组时可以启动的横向缩减事件生成 JSON 有效载荷。

此有效负载包含有关 Amazon A EC2 uto Scaling 需要终止的容量、它建议终止的实例列表以及启动终止的事件的信息。

以下是示例负载：

```
{
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-
id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 2,
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 1,
      "InstanceMarketOption": "spot"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "Capacity": 3,
      "InstanceMarketOption": "on-demand"
    }
  ],
  "Instances": [
    {
      "AvailabilityZone": "us-east-1b",
      "InstanceId": "i-0056faf8da3e1f75d",
      "InstanceType": "t2.nano",
```

```
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-02e1c69383a3ed501",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-036bc44b6092c01c7",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    ...
  ],
  "Cause": "SCALE_IN"
}
```

有效负载包括 Auto Scaling 组的名称、其 Amazon Resource Name (ARN) 以及以下元素：

- `CapacityToTerminate` 描述了在给定可用区域内设置为终止的竞价或按需容量的大小。
- `Instances`表示 Amazon A EC2 uto Scaling 根据中的信息建议终止的实例`CapacityToTerminate`。
- `Cause` 描述了导致终止的事件：`SCALE_IN`、`INSTANCE_REFRESH`、`MAX_INSTANCE_LIFETIME` 或者 `REBALANCE`。

以下信息概述了 Amazon A EC2 uto Scaling 如何生成输入数据的最重要因素：`Instances`

- 当实例由于横向缩减事件和基于实例刷新的终止而终止时，优先保持可用区间的平衡。因此，如果某个可用区的实例数多于该组使用的其他可用区，则您的终止策略将应用于不均衡可用区中的实例。如果组使用的可用区是均衡的，则输入数据将包含组的所有可用区中的实例。
- 使用混合实例策略，则根据您对每个购买选项的所需百分比，维持您的竞价和按需容量的平衡也将优先考虑。我们首先确定应终止两种类型中的哪种类型（竞价或按需）。然后，我们确定哪些实例（在已确定的购买选项内），我们可以终止哪些可用区，这些实例将导致可用区域最平衡。

响应数据

输入数据和响应数据协同工作，以缩小要终止的实例列表。

对于给定的输入，Lambda 函数的响应应类似于以下示例：

```
{
  "InstanceIDs": [
    "i-02e1c69383a3ed501",
    "i-036bc44b6092c01c7",
    ...
  ]
}
```

这些区域有：InstanceIDs 表示准备终止的实例。

或者，您可以返回准备终止的一组不同的实例，这些实例将覆盖输入数据中的实例。如果在您的 Lambda 函数被调用时没有任何实例可以终止，您也可以选择不返回任何实例。

没有准备好终止的实例时，Lambda 函数的响应应类似于以下示例：

```
{
  "InstanceIDs": [ ]
}
```

注意事项

注意以下使用自定义终止策略时的注意事项：

- 首先在响应数据中返回实例并不能保证其终止。如果在调用 Lambda 函数时返回的实例数量超过所需数量，Amazon A EC2 uto Scaling 会根据您为 Auto Scaling 组指定的其他终止策略评估每个实例。当存在多个终止策略时，它会尝试应用列表中的下一个终止策略，如果实例数量超过了终止所需的实例，则会转到下一个终止策略，依此类推。如果未指定其他终止策略，则使用默认终止策略来确定要终止的实例。
- 如果未返回任何实例，或者您的 Lambda 函数超时，Amazon A EC2 uto Scaling 会稍等片刻，然后再再次调用您的函数。对于任何横向缩减事件，只要组的所需容量小于其当前容量，它就会继续尝试。例如，基于刷新的终止，它会持续尝试一个小时。之后，如果它继续无法终止任何实例，则实例刷新操作将失败。在最长实例寿命的情况下，Amazon A EC2 uto Scaling 会不断尝试终止被确定为超过其最长生命周期的实例。
- 由于您的函数会重复重试，因此请确保在使用 Lambda 函数作为自定义终止策略之前测试并修复代码中的任何永久性错误。
- 如果您使用自己的待终止实例列表覆盖输入数据，而终止这些实例会使可用区失去平衡，那么 Amazon A EC2 uto Scaling 会逐渐重新平衡可用区间的容量分配。首先，它调用你的 Lambda 函

数，查看是否有准备终止的实例，以便它可以确定是否开始重新平衡。如果存在可以终止的实例，它会首先启动新实例。当实例完成启动后，它会检测到组的当前容量高于其所需容量，并启动横向缩减事件。

- 自定义终止策略不会影响您同时使用横向缩减保护功能来保护某些实例不被终止的能力。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

创建 Lambda 函数

首先创建 Lambda 函数，以便您可以在 Auto Scaling 组的终止策略中指定 Amazon Resource Name (ARN)。

要创建 Lambda 函数（控制台）

1. 打开 Lambda 控制台的 Functions（函数）页面。
2. 在屏幕顶部的导航栏中，选择您在创建 Auto Scaling 组时使用的同一区域。
3. 选择 Create function（创建函数），然后选择 Author from scratch（从头开始创作）。
4. 在基本信息下的函数名称中输入函数的名称。
5. 选择 Create function (创建函数)。返回到函数的代码和配置。
6. 让函数在控制台中保持打开状态，在函数代码，请将代码粘贴到编辑器中。
7. 选择部署。
8. 或者，通过选择版本选项卡，然后选择发布新版本，创建 Lambda 函数的发布版本。要了解有关 Lambda 中的版本控制的详细信息，请参阅 Amazon Lambda 开发人员指南中的 Lambda 函数版本。
9. 如果您选择发布版本，在别名选项卡选择是否要将别名与此版本的 Lambda 函数关联。要了解有关 Lambda 中的别名的详细信息，请参阅 Amazon Lambda 开发人员指南中的 Lambda 函数别名。
10. 接下来，选择配置选项卡，然后 Permissions（权限）。
11. 向下滚动到基于资源的策略，然后选择添加权限。基于资源的策略用于向在策略中指定的委托人授予调用函数的权限。在这种情况下，委托人将是与 A EC2 uto Scaling 组关联的 Amazon Auto Scaling 服务相关角色。
12. 在策略语句部分中，配置您的权限：

    a. 选择 Amazon Web Services 账户。

b. 适用于委托人中，输入调用服务链接角色的 ARN，例如 **arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling**。

c. 在 “操作” 中，选择 lambda: InvokeFunction。

d. 对于声明 ID，输入唯一的声明 ID，如 **AllowInvokeByAutoScaling**。

e. 选择保存。

13. 在遵循以上说明操作后，继续作为下一步，在 Auto Scaling 组的终止策略中指定函数的 ARN。有关更多信息，请参阅 更改自动扩缩组的终止策略。

> **Note**
>
> 有关可用作开发 Lambda 函数的参考的示例，请参阅 Amazon A EC2 uto Scal GitHub ing 的存储库。

限制

- 您只能在 Auto Scaling 组的终止策略中指定一个 Lambda 函数。如果指定了多个终止策略，则必须先指定 Lambda 函数。
- 您可以使用非限定 ARN（不带后缀）或具有版本或别名作为后缀的限定 ARN 来引用 Lambda 函数。如果使用了不合格的 ARN（例如 `function:my-function`），您的基于资源的策略必须在函数的未发布版本上创建。如果使用了合格的 ARN（例如 `function:my-function:1` 或者 `function:my-function:prod`），您的基于资源的策略必须在函数的未发布版本上创建。
- 您不能使用后缀为 $LATEST 的合格 ARN。如果您尝试添加引用后缀为 $LATEST 的限定 ARN 的自定义终止策略，它将导致出现错误。
- 输入数据中提供的实例数量限制为 30,000 个实例。如果可以终止的实例超过 30,000 个，则输入数据将包含 `"HasMoreInstances": true` 以指示返回的实例的最大数目。
- Lambda 函数的最长运行时间为两秒（2000 毫秒）。作为最佳实践，您应该根据预期运行时间设置 Lambda 函数的超时值。Lambda 函数的默认超时时间为三秒，但这可以减少。
- 如果您的运行时间超出 2 秒的限制，则任何横向缩减操作都将处于暂停状态，直到运行时间降至该阈值以下。对于运行时间始终较长的 Lambda 函数，请找到一种缩短运行时间的方法，例如将结果缓存到其可在后续 Lambda 调用期间进行检索的位置。

使用实例横向缩减保护以控制实例终止

实例缩减保护使您可以控制 Amazon A EC2 uto Scaling 可以终止哪些实例。此功能的一个常见用例是扩缩基于容器的工作负载。有关更多信息，请参阅 设计您的应用程序以妥善处理实例终止。

默认情况下，创建自动扩缩组时将禁用实例横向缩减保护。这意味着 Amazon A EC2 uto Scaling 可以终止该组中的任何实例。

您可以在自动扩缩组上启用实例横向缩减保护设置，在实例启动后立即对其进行保护。当实例状态为 `InService` 时，实例缩减保护启动。然后，如需控制哪些实例可以终止，请禁用自动扩缩组中单个实例的横向缩减保护设置。这样做可以继续保护某些实例免遭意外终止。

主题

注意事项

使用实例横向缩减保护时应注意以下事项：

- 如果自动扩缩组中的所有实例都受横向缩减保护并且发生横向缩减事件，则该组的所需容量会递减。不过，自动扩缩组不能终止所需数量的实例，直到其实例横向缩减保护设置被禁用。在 Auto Scaling 组的活动历史记录中，如果 Auto Scaling 组中的所有实例都受到保护，在发生缩减事件时不会缩小，则会显示以下消息： Amazon Web Services Management ConsoleCould not scale to desired capacity because all remaining instances are protected from scale in.
- 当您分离受横向缩减保护的实例时，其实例横向缩减保护设置就会失效。再次将实例附加到组时，它会继承组的当前实例横向缩减保护设置。当 Amazon A EC2 uto Scaling 启动新实例或将实例从温池移至 Auto Scaling 组时，该实例将继承 Auto Scaling 组的实例规模保护设置。
- 实例缩减保护并不能针对以下情况保护 Auto Scaling 实例：
  - 实例未通过运行状况检查的情况下的运行状况检查更换。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。
  - 竞价型实例中断。当竞价型实例的容量不再可用或 Spot 价格超过您的最高价时，将终止该实例。
  - 容量块预留结束。Amazon EC2 会收回容量块实例，即使这些实例受到保护以免缩容。

- 通过 `terminate-instance-in-auto-scaling-group` 命令手动终止。有关更多信息，请参阅 终止您自动扩缩组中的实例（Amazon CLI）。
- 通过 Amazon EC2 控制台、CLI 命令和 API 操作手动终止。要保护 Auto Scaling 实例免遭手动终止，请启用亚马逊 EC2 终止保护。（这不会阻止 Amazon A EC2 uto Scaling 通过该`terminate-instance-in-auto-scaling-group`命令终止实例或手动终止。）有关在启动模板中启用 Amazon EC2 终止保护的信息，请参阅使用高级设置创建启动模板。

更改自动扩缩组的横向缩减保护

您可以启用或禁用 Auto Scaling 组的实例缩减保护设置。启用该功能后，该组启动的所有新实例都将启用实例横向缩减保护。

为自动扩缩组启用或禁用此设置不会影响现有的实例。

Console

启用新自动扩缩组的横向缩减保护

创建自动扩缩组时，在配置组大小和扩展策略页面的实例横向缩减保护下，选中启用实例的横向缩减保护复选框。

启用或禁用现有组的横向缩减保护

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
4. 对于实例横向缩减保护，请选中或清除启用实例扩缩保护复选框，以根据需要启用或禁用此选项。
5. 选择更新。

Amazon CLI

启用新自动扩缩组的横向缩减保护

使用以下 create-auto-scaling-group 命令启用实例缩减保护。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in ...
```

启用现有组的横向缩减保护

使用以下update-auto-scaling-group命令为指定的 Auto Scaling 组启用实例缩减保护。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in
```

禁用现有组的横向缩减保护

使用以下命令为指定组禁用实例缩减保护。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-instances-protected-from-scale-in
```

更改实例的横向缩减保护

默认情况下，实例从其 Auto Scaling 组获取其实例缩减保护设置。但是，您可以在实例启动后对单个实例启用或禁用实例横向缩减保护。

Console

启用或禁用实例的横向缩减保护

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Instance management（实例管理）选项卡的 Instances（实例）中，选择实例。
4. 要启用实例缩减保护，请依次选择 Actions（操作）和 Set scale-in protection（设置缩减保护）。系统提示时，选择 Set scale-in protection（设置缩减保护）。
5. 要禁用实例缩减保护，请依次选择 Actions（操作）和 Remove scale-in protection（删除缩减保护）。系统提示时，选择 Remove scale-in protection（删除缩减保护）。

Amazon CLI

启用实例的横向缩减保护

使用以下 set-instance-protection 命令为指定的实例启用实例缩减保护。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-
group-name my-asg --protected-from-scale-in
```

禁用实例的横向缩减保护

使用以下命令为指定实例禁用实例缩减保护。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-
group-name my-asg --no-protected-from-scale-in
```

> **Note**
>
> 请记住，实例缩减保护并不能保证在发生人为错误时不会终止实例，例如，有人使用 Amazon 控制台或手动终止实例。 EC2 Amazon CLI为了保护您的实例免受意外终止，您可以使用 Amazon EC2 终止保护。但是，即使启用了终止保护和实例扩展保护，如果运行状况检查确定实例运行状况不佳或组本身被意外删除，保存到实例存储的数据也可能会丢失。与任何环境一样，最佳做法是频繁备份您的数据，或者在适合您的业务连续性要求时备份数据。

设计您的应用程序以妥善处理实例终止

本主题介绍以下功能：您可以使用这些功能防止自动扩缩组终止那些尚未准备好终止的实例，或者防止过快地终止实例以至于它们无法完成分配的任务。您可以将所有这三项功能组合使用，也可以单独使用来设计应用程序，以妥善处理实例终止。

例如，假设您有一个 Amazon SQS 队列，用于收集长时间运行的任务的传入消息。当新消息到达时，自动扩缩组中的一个实例会检索该消息并开始对其进行处理。处理每个消息需要 3 小时。随着消息数量的增加，新的实例会自动添加到自动扩缩组中。随着消息数量的减少，现有实例会自动终止。在这种情况下，Amazon A EC2 uto Scaling 必须决定终止哪个实例。默认情况下，Amazon A EC2 uto Scaling 可能会终止处理长达 3 小时的任务后已有 2.9 小时的实例，而不是当前处于空闲状态的实例。为了避免在使用 Amazon A EC2 uto Scaling 时出现意外终止问题，您必须设计应用程序以应对这种情况。

内容

> **Important**
>
> 在 Amazon A EC2 uto Scaling 上设计应用程序以优雅地处理实例终止时，请记住这些要点。
>
> - 如果实例运行状况不佳，无论您使用哪种功能，Amazon A EC2 uto Scaling 都会将其替换（除非您暂停该ReplaceUnhealthy进程）。您可以使用生命周期挂钩来允许应用程序正常关闭，或者在实例终止之前复制需要恢复的任何数据。
> - 无法保证终止生命周期挂钩是否会在实例终止之前运行或完成。如果出现故障，Amazon A EC2 uto Scaling 仍会终止该实例。

横向缩减保护实例

在许多情况下，您可以使用实例横向缩减保护，此时的终止实例是一项关键操作，默认应拒绝该操作，并且仅明确允许供指定实例使用。例如，在运行容器化工作负载时，通常希望保护所有实例，仅对没有当前任务或计划任务的实例取消保护。诸如 Amazon ECS 之类的服务已在其产品中内置集成了实例横向缩减保护。

您可以在自动扩缩组上启用横向缩减保护，以便在创建实例时对其应用横向缩减保护，并对现有实例启用横向缩减保护。当实例没有更多作业要做时，它可以关闭保护。该实例可以继续轮询新作业，并在分配了新作业时重新启用保护。

应用程序可以从用于管理实例是否可终止的集中控制面板设置保护，也可以从实例本身设置保护。但是，如果大量实例不断切换横向缩减保护，那么大的实例集可能会遇到节流问题。

有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。

自定义终止策略

与实例横向缩减保护一样，自定义终止策略可帮助您防止自动扩缩组终止指定的实例。

默认情况下，自动扩缩组使用默认终止策略来确定首先终止哪个实例。如果您想更好地控制首先终止哪些实例，则可以使用 Lambda 函数实现您自己的自定义终止策略。每当 Amazon A EC2 uto Scaling 必

须决定终止哪个实例时，它都会调用该函数。它只会终止该函数返回的实例。如果函数出错、超时或生成空列表，Amazon A EC2 uto Scaling 不会终止实例。

如果知道某个实例何时足够冗余或利用率不足，因此可以终止该实例，那么自定义终止策略会非常有用。为了支持这一观点，您需要使用控制面板来实现应用程序，由该平面监控整个组的工作负载。这样，如果一个实例仍在处理作业，那么 Lambda 函数就会知道不将它包含在内。

有关更多信息，请参阅 了解使用 Lambda 创建自定义终止策略。。

终止生命周期挂钩

终止生命周期挂钩可延长已选择终止的实例的使用寿命。它提供了额外的时间来完成当前分配给实例的所有消息或请求，或者保存进度并将工作转移给另一个实例。

对于许多工作负载，生命周期挂钩可能足以正常地关闭运行着已选择终止的实例的应用程序。这是一种尽最大努力的方法，不能用于在出现故障时防止终止。

要使用生命周期挂钩，您需要知道何时选择终止实例。有两种方法可以知道这一点：

选项	描述	最适合用于	指向文档的链接
在实例之内	实例元数据服务 (IMDS) 是一个安全端点，您可以通过它来直接从实例中轮询实例的状态。如果元数据随之返回 `Terminated` ，则您的实例将按计划终止。	在实例终止之前必须对实例执行操作的应用程序。	检索目标生命周期状态
在实例之外	当实例终止时，会生成事件通知。您可以使用亚马逊 EventBridge、亚马逊 SQS 或 Amazon SNS 创建规则来捕获这些事件，并调用响应，例如使用 Lambda 函数。	需要在实例之外进行操作的应用程序。	创建通知目标

要使用生命周期挂钩，您还需要知道您的实例何时准备好完全被终止。Amazon A EC2 uto Scaling 不会告诉亚马逊终止该实例，直到它接 EC2 到CompleteLifecycleAction呼叫或超时过去（以先发生者为准）。

默认情况下，由于终止生命周期挂钩的存在，实例可以继续运行一小时（检测信号超时）。如果一小时的时间不足以完成生命周期操作，则可以配置默认超时。当生命周期操作实际正在进行时，您可以通过RecordLifecycleActionHeartbeatAPI 调用延长超时时间。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

暂停和恢复 Amazon A EC2 uto Scaling 流程

本主题描述了如何暂停然后恢复自动扩缩组的一个或多个进程，以暂时禁用某些操作。

当您需要在不受扩缩策略或计划操作干扰的情况下调查或排查问题时，暂停进程非常有用。它还有助于防止 Amazon A EC2 uto Scaling 在您对 Auto Scaling 组进行更改时将实例标记为运行状况不佳并替换它们。

主题

> **Note**
>
> 除了您启动的暂停之外，Amazon A EC2 uto Scaling 还可以暂停反复无法启动实例的 Auto Scaling 群组的进程。这称为管理暂停。管理暂停最常用于符合以下条件的 Auto Scaling 组：连续尝试启动实例的时间超过 24 小时，但是未成功启动任何实例。您可以恢复 Amazon A EC2 uto Scaling 出于管理原因暂停的流程。

进程的类型

暂停-恢复功能支持以下进程：

- Launch— 在 Auto Scaling 组扩展时，或者当 Amazon A EC2 uto Scaling 出于其他原因（例如向温池中添加实例时）选择启动实例时，将实例添加到 Auto Scaling 组。

- `Terminate`— 当 Auto Scaling 组缩小规模，或者 Amazon A EC2 uto Scaling 出于其他原因选择终止实例时，例如当实例因超过其最大生命周期或未通过运行状况检查而终止实例时，从 Auto Scaling 组中移除实例。
- `AddToLoadBalancer`：在实例启动时，将其添加到附加的负载均衡器目标组或者经典负载均衡器。有关更多信息，请参阅 使用 Elastic Load Balancing 在 Auto Scaling 组中分配传入的应用程序流量 。
- `AlarmNotification`— 接受来自与动态扩展策略关联的 CloudWatch警报的通知。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。
- `AZRebalance`— 当组变得不平衡时（例如，当以前不可用的可用区恢复正常状态时），在所有指定的可用区之间均匀地平衡组中的 EC2 实例数量。有关更多信息，请参阅 再平衡活动。
- `HealthCheck`— 如果亚马逊 EC2 或 Elastic Load Balancing 告诉 Amazon A EC2 uto Scaling 该实例运行状况不佳，则检查该实例的运行状况并将该实例标记为运行状况不佳。此流程可覆盖您手动设置的实例运行状况状态。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。
- `InstanceRefresh`：使用实例刷新功能终止并替换实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。
- `ReplaceUnhealthy`：终止被标记为运行状况不佳的实例，然后创建新实例以替换它们。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。
- `ScheduledActions`：执行您创建的计划扩缩操作，或您在创建 Amazon Auto Scaling 扩缩计划并开启预测扩缩时创建的计划扩缩操作。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。

暂停进程的注意事项

暂停进程之前，请注意以下事项：

- 暂停`AlarmNotification`允许您暂时停止群组的目标跟踪、步骤和简单扩展策略，而无需删除扩展策略或其关联 CloudWatch 警报。要暂时停止单个扩缩策略，请参阅禁用 Auto Scaling 组的扩缩策略。
- 您可以选择暂停`HealthCheck`和`ReplaceUnhealthy`进程以重启实例，而不会让 Amazon A EC2 uto Scaling 根据其运行状况检查终止实例。但是，如果您需要 Amazon A EC2 uto Scaling 继续对剩余的实例执行运行状况检查，请改用备用功能。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。
- 如果您暂停了 `Launch` 和 `Terminate` 进程或者 `AZRebalance`，并且随后对自动扩缩组进行了更改（例如，分离实例或更改指定的可用区），则您的组可能会在可用区之间失去均衡。如果发生这种情

况，在您恢复暂停的流程后，Amazon A EC2 uto Scaling 会逐渐在可用区域之间均匀地重新分配实例。

- 如果您暂停该Terminate进程，您仍然可以使用带有强制删除选项的delete-auto-scaling-group命令来强制终止实例。
- 暂停 Terminate 进程仅适用于当前处于 InService 状态的实例。它不会阻止对处于其他状态（例如 Pending）的实例，或无法从备用状态正常恢复的实例的终止。
- 如果该RemoveFromLoadBalancerLowPriority过程出现在使用 Amazon CLI 或描述 Auto Scaling 组的调用中，则可以忽略该过程 SDKs。此过程已过时，保留此过程仅是为了向后兼容。

暂停进程

要暂停自动扩缩组的进程，请使用以下方法之一：

Console

暂停一个流程

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
4. 对于 Suspended processes（暂停的进程），选择要暂停的进程。
5. 选择更新。

Amazon CLI

使用下面的 suspend-processes 命令暂停单个进程。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck ReplaceUnhealthy
```

要暂停所有进程，请忽略 --scaling-processes 选项，如下所示。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

恢复进程

要恢复自动扩缩组的暂停进程，请使用以下方法之一：

Console

恢复一个暂停的流程

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，选择 Advanced configurations（高级配置）、Edit（编辑）。
4. 对于 Suspended processes（已暂停的进程），移除该已暂停的进程。
5. 选择更新。

Amazon CLI

要恢复已暂停的进程，请使用以下 resume-processes 命令。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck
```

要恢复所有已暂停的进程，请忽略 `--scaling-processes` 选项，如下所示。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

暂停进程如何影响其他进程

以下各节描述了单独暂停不同进程时发生的情况。

主题

- Launch 已暂停
- Terminate 已暂停

Launch 已暂停

- AlarmNotification 仍处于活动状态，但是您的自动扩缩组无法为超限警报启动横向扩展活动。
- ScheduledActions 处于活动状态，但是您的自动扩缩组无法为出现的任何计划操作启动横向扩展活动。
- AZRebalance 停止对组进行重新平衡。
- ReplaceUnhealthy 继续终止运行不正常的实例，但不启动替换实例。当您恢复该Launch流程时，Amazon A EC2 uto Scaling 会立即替换在暂停期间终止的所有实例。Launch
- InstanceRefresh 不会替换实例。

Terminate 已暂停

- AlarmNotification 仍处于活动状态，但是您的自动扩缩组无法为超限警报启动横向缩减活动。
- ScheduledActions 处于活动状态，但是您的自动扩缩组无法为出现的任何计划操作启动横向缩减活动。
- AZRebalance 仍处于活动状态，但不能正常运行。它可以启动新实例而不终止旧实例。这可能导致您的 Auto Scaling 组增加到比最大大小超出百分之十，因为在重新平衡活动期间允许短时间内发生这种情况。您的 Auto Scaling 组可以保持超出其最大大小，直到您恢复 Terminate 进程。
- ReplaceUnhealthy 处于非活动状态但未 HealthCheck。当 Terminate 恢复后，ReplaceUnhealthy 进程将立即开始运行。如果任何实例在 Terminate 暂停期间被标记为运行状况不佳，将立即替换它们。
- InstanceRefresh 不会替换实例。

AddToLoadBalancer 已暂停

- Amazon A EC2 uto Scaling 会启动实例，但不会将其添加到负载均衡器目标组或 Classic Load Balancer。在您恢复 AddToLoadBalancer 进程后，该进程也会在启动实例时将其添加到负载均衡器。不过，它不会添加在此流程暂停时启动的实例。您必须手动注册这些实例。

AlarmNotification 已暂停

- 当 CloudWatch 警报阈值被违反时，Amazon A EC2 uto Scaling 不会调用扩展策略。当您恢复时AlarmNotification，Amazon A EC2 uto Scaling 会考虑当前存在违规警报阈值的策略。

AZRebalance 已暂停

- 在某些事件发生后，Amazon A EC2 uto Scaling 不会尝试重新分配实例。不过，如果发生横向扩展或横向缩减事件，扩缩进程仍会尝试平衡可用区。例如，在扩展期间，它会在可用区中启动实例最少的实例。如果群组在暂停期间变得不平衡，而AZRebalance您又将其恢复，Amazon A EC2 uto Scaling 会尝试重新平衡该群组。它先调用 Launch，然后调用 Terminate。
- 暂停使用时AZRebalance，温水池不会受到影响。

HealthCheck 已暂停

- 由于 EC2 和 Elastic Load Balancing 运行状况检查，Amazon A EC2 uto Scaling 停止将实例标记为运行状况不佳。您的自定义运行状况检查会继续正常运行。当您暂停 HealthCheck 后，在需要时可以手动设置组中实例的运行状况，并由 ReplaceUnhealthy 替换它们。

InstanceRefresh 已暂停

- 由于实例刷新，Amazon A EC2 uto Scaling 会停止替换实例。如果正在刷新实例，则会暂停操作而不将其取消。

ReplaceUnhealthy 已暂停

- Amazon A EC2 uto Scaling 会停止替换标记为运行状况不佳的实例。失败 EC2 或 Elastic Load Balancing 运行状况检查失败的实例仍被标记为运行状况不佳。一旦您恢复

该ReplaceUnhealthy流程，Amazon A EC2 uto Scaling 就会替换在该流程暂停期间被标记为运行状况不佳的实例。ReplaceUnhealthy 进程会首先调用 Terminate，然后调用 Launch。

ScheduledActions 已暂停

- Amazon A EC2 uto Scaling 不会运行计划在暂停期间运行的计划操作。恢复时ScheduledActions，Amazon A EC2 uto Scaling 仅考虑计划时间尚未过的计划操作。

其它注意事项

此外，在暂停 Launch 或 Terminate 时，以下功能可能无法正常运行：

- 最大实例生命周期：暂停 Launch 或 Terminate 时，最大实例生命周期功能无法替换任何实例。
- 竞价型实例中断：如果 Terminate 已暂停且您的自动扩缩组包含竞价型实例，这些实例在竞价型容量不再可用的情况下仍可终止。暂停期间Launch，Amazon A EC2 uto Scaling 无法从另一个竞价型实例池或同一竞价型实例池中启动替代实例（当该实例池再次可用时）。
- 容量再平衡 — 如果已暂停Terminate并且您使用容量再平衡来处理竞价型实例中断，则在竞价型容量不再可用时，Amazon EC2 Spot 服务仍可以终止实例。如果Launch已暂停，Amazon A EC2 uto Scaling 将无法从另一个竞价型实例池启动替换实例，也无法在同一竞价型实例池再次可用时从该池启动替换实例。
- 附加和分离实例：暂停 Launch 和 Terminate 时，您可以分离附加到自动扩缩组的实例，但在暂停 Launch 期间，您无法将新实例附加到该组。
- 备用实例：暂停 Launch 和 Terminate 时，您可以将实例置于 Standby 状态，但在暂停 Launch 期间，您无法将处于 Standby 状态的实例恢复为服务状态。

监控您的 Amazon A EC2 uto Scaling 群组

监控是维护 Amazon A EC2 uto Scaling 和您的 Amazon Web Services 云 解决方案的可靠性、可用性和性能的重要组成部分。 Amazon 提供了以下监控工具，用于监视 Amazon A EC2 uto Scaling，在出现问题时进行报告，并在适当时自动采取措施：

运行状况检查

Amazon A EC2 uto Scaling 会定期对您的 Auto Scaling 组中的实例执行运行状况检查。如果实例未通过运行状况检查，则该实例将被标记为运行状况不佳，并将在Amazon A EC2 uto Scaling启动新实例来替换该实例时终止。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

Amazon Health Dashboard

Amazon Health Dashboard 显示信息，还提供 Amazon 资源运行状况变化时调用的通知。信息会以两种方式显示：在显示按类别组织的最近和未来事件的控制面板上，以及在显示过去 90 天内所有事件的完整事件日志中。有关更多信息，请参阅 Amazon Health Dashboard 亚马逊 A EC2 uto Scaling 的通知。

CloudTrail

使用 Amazon CloudTrail，您可以跟踪您的个人或代表您向 Amazon A EC2 uto Scaling API 发出的调用 Amazon Web Services 账户。 CloudTrail 将信息存储在您指定的 Amazon S3 存储桶中的日志文件中。您可以使用这些日志文件监控 Auto Scaling 组的活动。日志包括发出的请求、请求来自的源 IP 地址、发出请求的用户、发出请求的时间，等等。有关更多信息，请参阅 使用记录 Amazon A EC2 uto Scaling API 调用 Amazon CloudTrail。

> **为您的 Amazon EC2 实例收集日志**
>
> 您可以使用 CloudWatch 从操作系统收集 EC2实例的日志。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器收集指标和日志以及查看发送到 CloudWatch 日志的日志数据。
> 有关可帮助您记录和收集工作负载数据的其他 Amazon 服务的信息，请参阅Amazon 规范性指南中的应用程序所有者日志记录和监控指南。

亚马逊 CloudWatch

Amazon CloudWatch 可帮助您分析日志，并实时监控您的 Amazon 资源和托管应用程序的指标。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值

时通知您或采取措施。例如，您可以在网络活动突然高于或低于指标的预期值时收到通知。有关使用此服务监控自动扩缩组和实例指标的更多信息，请参阅 CloudWatch 监控您的 Auto Scaling 组和实例的指标。

CloudWatch 还会跟踪 Amazon A EC2 uto Scaling 的 Amazon API 使用率指标。您可以使用这些指标来配置警报，以在 API 调用量超过您定义的阈值时提醒您。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的Amazon 使用量指标。

Amazon Compute Optimizer

Compute Optimizer 提供亚马逊 EC2 实例建议，可以帮助您决定是否迁移到新的实例类型。它可以监控自动扩缩组的实例类型是否最佳，并生成降低成本和提高工作负载性能的建议。有关更多信息，请参阅 通过获取实例类型建议 Amazon Compute Optimizer。

亚马逊 EventBridge

Amazon EventBridge 是一项无服务器事件总线服务，可以轻松地将您的应用程序与来自各种来源的数据连接起来。 EventBridge 提供来自您自己的应用程序、 Software-as-a-Service (SaaS) 应用程序和 Amazon 服务的实时数据流，并将这些数据路由到 Lambda 等目标。这让您可以监控服务中发生的事件，并构建事件驱动型架构。有关更多信息，请参阅 用于处理 EventBridge Auto Scaling 事件。

Amazon Security Hub

Amazon Security Hub用于监控您对 Amazon A EC2 uto Scaling 的使用情况，因为它与安全最佳实践有关。Security Hub 使用侦测性安全控件来评估资源配置和安全标准，以帮助您遵守各种合规框架。有关使用 Security Hub 评估亚马逊 A EC2 uto Scaling 资源的更多信息，请参阅Amazon Security Hub 用户指南中的 Amazon A EC2 uto Scaling 控件。

Amazon Simple Notification Service

您可以将 Auto Scaling 群组配置为在 Amazon A EC2 uto Scaling 启动或终止实例时发送 Amazon SNS 通知。有关更多信息，请参阅 亚马逊 Auto Scaling 的亚马 EC2 逊 SNS 通知选项。

自动扩缩组中实例的运行状况检查

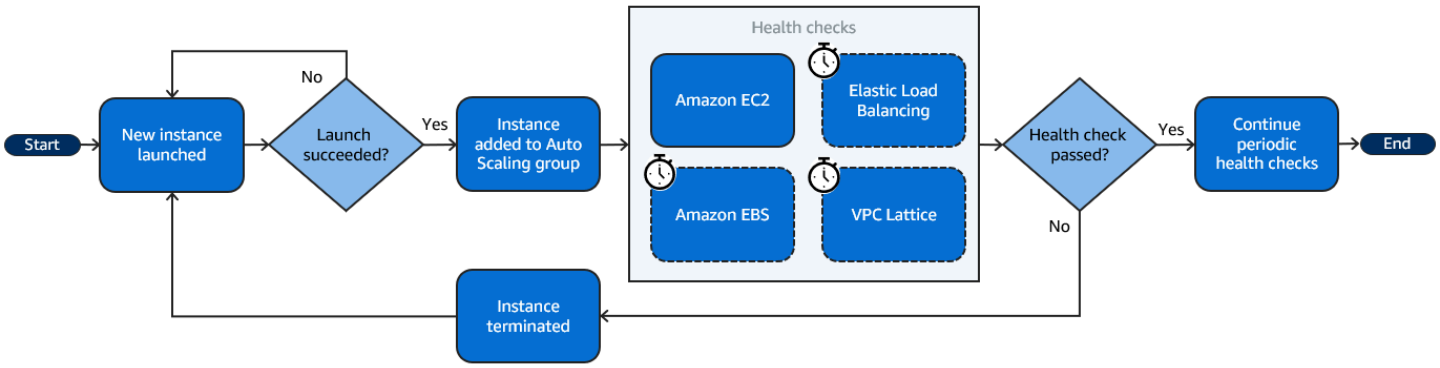
Amazon A EC2 uto Scaling 会持续监控 Auto Scaling 组中实例的运行状况以保持所需的容量。

自动扩缩组中的所有实例均以 `Healthy` 状态开始。除非 Amazon A EC2 uto Scaling 收到实例运行状况不佳的通知，否则这些实例将被视为运行状况良好。当实例运行状况不佳且必须更换时，它可以接收来自各种来源的通知。这些源包括以下内容：

- Amazon EC2
- Elastic Load Balancing
- VPC Lattice
- Amazon EBS
- 您定义的自定义运行状况检查

当 Amazon A EC2 uto Scaling 确定某个`InService`实例运行状况不佳时，它会将其替换为新实例，以保持该组的所需容量。新实例使用自动扩缩组的当前设置及其关联的启动模板或启动配置启动。

以下流程图说明在自动扩缩组中启动新实例的过程。首先启动实例。如果启动成功，则该实例将添加到自动扩缩组。然后，Amazon A EC2 uto Scaling 使用内置的 Amazon EC2 状态检查对实例执行运行状况检查，并在宽限期过后，使用您为该组启用的任何可选运行状况检查。这些运行状况检查会定期持续进行。如果任何运行状况检查失败，则该实例将被替换。



当实例意外终止（例如竞价型实例中断或用户手动终止）时，也可能出现运行状况不佳的实例。同样，在这种情况下，Amazon A EC2 uto Scaling 将自动启动替换实例，以保持所需的容量。

内容

- 关于自动扩缩组的运行状况检查
- 设置自动扩缩组的运行状况检查宽限期
- 使用运行状况检查监控具有受损 Amazon EBS 卷的 Auto Scaling 实例
- 为您的自动扩缩组设置自定义运行状况检查
- 查看运行状况检查失败原因
- 对 Amazon A EC2 uto Scaling 中运行不正常的实例进行故障排除

关于自动扩缩组的运行状况检查

本主题概述了可用的运行状况检查类型，并介绍了将 Amazon A EC2 uto Scaling 运行状况检查与您的应用程序集成的关键注意事项。

内容

运行状况检查类型

Amazon A EC2 uto Scaling 可以使用以下一项或多项运行状况检查来确定`InService`实例的运行状况：

运行状况检查类型	检查内容
Amazon EC2 状态检查和预定活动	• 检查实例是否正在运行。 • 检查可能会影响实例的底层硬件或软件问题。 这是自动扩缩组的默认运行状况检查类型。
Elastic Load Balancing 运行状况检查	• 检查负载均衡器是否将实例报告为运行正常，以确认实例是否可用于处理请求。 要运行此运行状况检查类型，您必须为自动扩缩组开启此运行状况检查。
VPC Lattice 运行状况检查	• 检查 VPC Lattice 是否将实例报告为运行正常，以确认实例是否可用于处理请求。

运行状况检查类型	检查内容
	要运行此运行状况检查类型，您必须为自动扩缩组开启此运行状况检查。
Amazon EBS 运行状况检查	• 检查 EBS 卷是否可访问以及是否通过 I/O 状态检查。 要运行此运行状况检查类型，您必须为自动扩缩组开启此运行状况检查。
自定义运行状况检查	• 根据自定义运行状况检查的结果，检查可能表明实例存在运行状况问题的任何其他问题。

Amazon EC2 健康检查

实例启动后，它会附加到自动扩缩组并进入 `InService` 状态。要详细了解 Auto Scaling 组中实例的不同生命周期状态，请参阅 Amazon A EC2 uto Scaling 实例生命周期。

Amazon A EC2 uto Scaling 会定期检查 Auto Scaling 组内所有实例的运行状况，以确保它们正在运行且状态良好。

状态检查

Amazon A EC2 uto Scaling 使用亚马逊 EC2 实例状态检查和系统状态检查的结果来确定实例的运行状况。如果实例处于除之外的任何 Amazon EC2 状态`running`，或者其状态检查状态变为`impaired`，则 Amazon A EC2 uto Scaling 会认为该实例运行状况不佳并替换它。这包括当实例为任何以下状态时：

- `stopping`
- `stopped`
- `shutting-down`
- `terminated`

Amazon EC2 状态检查不需要任何特殊配置，并且始终处于启用状态。有关更多信息，请参阅 Amazon EC2 用户指南中的状态检查类型。

> **Important**
>
> Amazon A EC2 uto Scaling 允许状态检查偶尔失败，无需采取任何措施。当状态检查失败时，Amazon A EC2 uto Scaling 会等待几分钟 Amazon 才能修复问题。它不会在状态检查的状态变为 `impaired` 时立即将实例标记为 `Unhealthy`。
> 但是，如果 Amazon A EC2 uto Scaling 检测到某个实例不再处于该`running`状态，则这种情况将被视为立即出现故障。在这种情况下，它会立即将实例标记为 `Unhealthy` 并予以替换。

计划的事件

Amazon 偶尔 EC2 会将您的实例上的事件安排在特定时间戳之后运行。有关更多信息，请参阅 Amazon EC2 用户指南中的实例计划事件。

如果您的一个实例受到计划事件的影响，Amazon A EC2 uto Scaling 会认为该实例运行状况不佳并替换它。在到达时间戳中指定的日期和时间之前，实例不会开始关闭。

Elastic Load Balancing 运行状况检查

当您为 Auto Scaling 组启用 Elastic Load Balancing 运行状况检查时，Amazon A EC2 uto Scaling 可以使用这些运行状况检查的结果来确定实例的运行状况。

您必须先配置一个 Elastic Load Balancing 负载均衡器并为其配置运行状况检查以确定实例是否运行正常，然后才能为自动扩缩组开启 Elastic Load Balancing 运行状况检查。有关更多信息，请参阅 准备附加 Elastic Load Balancing 负载均衡器。

将负载均衡器附加到您的自动扩缩组后，会发生以下情况：

- Amazon A EC2 uto Scaling 将 Auto Scaling 组中的实例注册到负载均衡器。
- 实例完成注册后，它会进入 `InService` 状态并可与该负载均衡器一起使用。

默认情况下，Amazon A EC2 uto Scaling 会忽略 Elastic Load Balancing 运行状况检查的结果。
在您为 Auto Scaling 组开启这些运行状况检查后，当 Elastic Load Balancing 将注册的实例报告为时`Unhealthy`，Amazon A EC2 uto Scaling 会将该实例`Unhealthy`标记为下一次定期运行状况检查并替换该实例。

如果您的负载均衡器启用了连接耗尽（取消注册延迟），Amazon A EC2 uto Scaling 会等待运行中请求完成或最大超时到期，然后才会终止运行状况不佳的实例。

> **Note**
>
> 有关如何附加负载均衡器以及如何为自动扩缩组开启 Elastic Load Balancing 运行状况检查的说明，请参阅将 Elastic Load Balancing 负载均衡器附加到自动扩缩组。
> 当您为组启用 Elastic Load Balancing 运行状况检查时，Amazon A EC2 uto Scaling 可以替换 Elastic Load Balancing 报告为运行状况不佳的实例，但前提是负载均衡器处于该`InService`状态。有关更多信息，请参阅 验证负载均衡器的附加状态。

VPC Lattice 运行状况检查

默认情况下，Amazon A EC2 uto Scaling 会忽略 VPC 莱迪思运行状况检查的结果。您可以选择为自动扩缩组开启这些运行状况检查。执行此操作后，当 VPC Lattice 将注册实例报告为时`Unhealthy`，Amazon A EC2 uto Scaling 会在下一次定期运行状况检查中标记该实例`Unhealthy`并将其替换。注册实例然后检查其运行状况的过程与 Elastic Load Balancing 运行状况检查的过程相同。

> **Note**
>
> 有关如何附加 VPC Lattice 目标组以及如何为自动扩缩组开启 VPC Lattice 运行状况检查的说明，请参阅将 VPC Lattice 目标组附加到您的自动扩缩组。
> 当您为群组启用 VPC Lattice 运行状况检查时，Amazon A EC2 uto Scaling 可以替换 VPC Lattice 报告为运行状况不佳的实例，但前提是目标组处于该状态。`InService`有关更多信息，请参阅 验证您的 VPC Lattice 目标组的附件状态。

Amazon A EC2 uto Scaling 如何最大限度地减少停机时间

默认情况下，在终止现有实例的同时会预置新实例，这可能导致在新实例完全运行之前无法接受新请求。

如果 Amazon A EC2 uto Scaling 确定任何实例已停止运行（或者已`Unhealthy`使用set-instance-health命令标记这些实例），它将立即替换它们。但是，如果发现其他实例运行状况不佳，Amazon A EC2 uto Scaling 将使用以下方法从故障中恢复。这种方法可尽可能减少可能因临时问题或运行状况检查配置错误导致的任何停机时间。

- 如果扩展活动正在进行中，并且您的 Auto Scaling 组的容量少于其所需容量 10% 或更多，则 Amazon A EC2 uto Scaling 会等待正在进行的扩展活动，然后再替换运行状况不佳的实例。

- 扩展时，Amazon A EC2 uto Scaling 会等待实例通过初始运行状况检查。它还会等待默认实例预热完成，以确保新实例准备就绪。
- 在实例完成预热并且该组已达到所需容量的 90% 以上之后，Amazon A EC2 uto Scaling 会按如下方式替换运行状况不佳的实例：
  - Amazon A EC2 uto Scaling 一次最多只能替换该群组所需容量的10%。它将在所有运行不正常的实例都被替换之前持续这样操作。
  - 在替换实例时，它会等待新实例通过初始运行状况检查。它还会等待默认的实例预热期结束，然后再继续操作。

> **Note**
>
> 如果 Auto Scaling 组的大小足够小，结果值 10% 小于 1，则 Amazon A EC2 uto Scaling 会逐一替换运行状况不佳的实例。这可能会导致该组出现短暂停机。
> 此外，如果 Elastic Load Balancing 运行状况检查报告某个 Auto Scaling 组中的所有实例都运行状况不佳，并且负载均衡器处于该`InService`状态，那么 Amazon A EC2 uto Scaling 一次可能会将较少的实例标记为运行状况不佳。这可能会导致一次替换的实例数量少于在其他场景中执行的 10% 这一比例。这使您有时间修复问题，而不会让 Amazon A EC2 uto Scaling 自动终止整个群组。

暖池中实例的运行状况检查

Amazon A EC2 uto Scaling 还会对温池中的实例执行运行状况检查。有关更多信息，请参阅 查看运行状况检查状态以及运行状况检查失败的原因。

运行状况检查注意事项

以下是使用 Amazon A EC2 uto Scaling 运行状况检查时的注意事项。

- 如果您需要在正在终止的实例上或正在启动的实例上执行某些操作，则可以使用生命周期钩子。这些挂钩允许您在 Amazon A EC2 uto Scaling 启动或终止实例时执行自定义操作。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。
- Amazon A EC2 uto Scaling 不提供从其运行 EC2 状况检查中删除亚马逊状态检查和计划事件的方法。如果您不想替换实例，则建议暂停任何单个 Auto Scaling 组的 `ReplaceUnhealthy` 和 `HealthCheck` 进程。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 要手动将运行状况不佳的实例的健康状态设置回原为`Healthy`，您可以尝试使用set-instance-health命令。如果您收到错误消息，则可能是因为该实例已经开始终止。通常，只有

在ReplaceUnhealthy进程或进程暂停的情况下，Healthy使用set-instance-health命令将实例的运行状况重新设置为才有用。Terminate

- 如果您需要在不受运行状况检查干扰的情况下对实例进行问题排查，则可以将该实例置于 Standby 状态。在您恢复使用Standby状态之前，Amazon A EC2 uto Scaling 不会对处于该状态的实例执行运行状况检查。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。
- 实例终止后，任何关联的弹性 IP 地址都会取消关联，并且不会自动与新实例关联。必须手动将弹性 IP 地址关联到新实例，或者使用基于生命周期挂钩的解决方案自动完成关联。有关更多信息，请参阅 Amazon EC2 用户指南中的弹性 IP 地址。
- 同样，实例终止后，其挂载的 EBS 卷也将被分离（或删除，具体取决于卷的 DeleteOnTermination 属性）。必须手动将这些 EBS 卷挂载到新实例，或者使用基于生命周期挂钩的解决方案自行完成挂载。有关更多信息，请参阅《Amazon EBS 用户指南》中的将 Amazon EBS 卷挂载到实例。

设置自动扩缩组的运行状况检查宽限期

当 Amazon A EC2 uto Scaling 运行状况检查确定某个InService实例运行状况不佳时，它会将其替换为新实例。运行状况检查宽限期指定了因新实例运行状况不正常而将其终止之前继续运行的最短时间（以秒为单位）。

例如，如果 Elastic Load Balancing 运行状况检查失败，且原因是实例仍在初始化，Amazon A EC2 uto Scaling 可能需要避免采取行动。Elastic Load Balancing 运行状况检查并行运行，从实例向负载均衡器注册时开始。宽限期可防止 Amazon A EC2 uto Scaling 标记您新启动的实例，如果这些实例Unhealthy在进入状态后没有立即通过这些运行状况检查，则不必要地将其终止。InService

在控制台中，创建自动扩缩组时的运行状况检查宽限期默认为 300 秒。使用 Amazon CLI 或 SDK 创建 Auto Scaling 组时，其默认值为 0 秒。值为 0 将关闭运行状况检查宽限期。

将此值设置得过高会降低 Amazon A EC2 uto Scaling 运行状况检查的有效性。如果您为实例启动使用了生命周期挂钩，则可以将运行状况检查宽限期设置为 0。借助生命周期挂钩，Amazon A EC2 uto Scaling 提供了一种方法来确保实例在进入状态之前始终处于初始化InService状态。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

宽限期适用于以下实例：

- 新启动的实例
- 处于待机状态后重新投入运行的实例
- 您手动附加到组的实例

> **Important**
>
> 在运行状况检查宽限期内，如果 Amazon A EC2 uto Scaling 检测到某个实例不再处于亚马逊 EC2 `running`状态，它会立即标记该实例`Unhealthy`并替换它。例如，假设您停止了自动扩缩组中的某个实例，则该实例会被标记为`Unhealthy`并被替换。

设置组的运行状况检查宽限期

您可以为新的和现有的自动扩缩组设置运行状况检查宽限期。

Console

修改新组的运行状况检查宽限期

创建自动扩缩组时，在配置高级选项页面的运行状况检查、运行状况检查宽限期中输入相应的时间（单位为秒）。这是 Amazon A EC2 uto Scaling 在实例进入状态后必须等待多长时间才能检查其运行`InService`状况。

Amazon CLI

修改新组的运行状况检查宽限期

将该`--health-check-grace-period`选项添加到create-auto-scaling-group命令中。以下示例将为一个名为 *my-asg* 的新自动扩缩组配置运行状况检查宽限期，其值为 *60* 秒。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 60 ...
```

Console

修改现有组的运行状况检查宽限期

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。

4. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
5. 在 Health check grace period（运行状况检查宽限期）下，输入时间长短，单位为秒。这是 Amazon A EC2 uto Scaling 在实例进入状态后必须等待多长时间才能检查其运行`InService`状况。
6. 选择更新。

Amazon CLI

修改现有组的运行状况检查宽限期

将该`--health-check-grace-period`选项添加到update-auto-scaling-group命令中。以下示例将为一个名为 *my-asg* 的现有自动扩缩组配置运行状况检查宽限期，其值为 *120* 秒。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 120
```

> **Note**
>
> 我们强烈建议同时为您的自动扩缩组设置默认实例预热时间。有关更多信息，请参阅 为 Auto Scaling 组设置原定设置实例预热。

使用运行状况检查监控具有受损 Amazon EBS 卷的 Auto Scaling 实例

您可以为您的 Auto Scaling 组启用 Amazon EBS 运行状况检查，以确保 Amazon A EC2 uto Scaling 监控运行应用程序的整个系统。

在您开启这些运行状况检查后，Amazon A EC2 uto Scaling 会收到对实例所连接的 EBS 卷执行的亚马逊 EC2状态检查的结果。如果卷无法访问或未通过 I/O 状态检查，则运行状况检查将失败，相应的实例将被视为运行状况不佳。当 Amazon A EC2 uto Scaling 检测到运行状况不佳的实例时，它会将其替换。

本主题假设您熟悉所附加的 EBS 状态检查。如果您不是，请参阅《亚马逊 EC2 用户指南》的 “附加 EBS 状态检查” 部分，了解详情。以下主题介绍如何开启依赖于所附的 EBS 状态检查的 Amazon A EC2 uto Scaling 运行状况检查。

> **Note**
>
> 您可以为所有自动扩缩组开启 Amazon EBS 运行状况检查。但是，这些运行状况检查仅适用于在 Amazon Nitro System 上构建的实例。

为组开启 Amazon EBS 运行状况检查

您可以为新的以及现有的自动扩缩组开启 Amazon EBS 运行状况检查。

Console

为新组开启 Amazon EBS 运行状况检查

创建自动扩缩组时，在配置高级选项页面上，对于运行状况检查、其他运行状况检查类型，请选择开启 Amazon EBS 运行状况检查。然后，在 运行状况检查宽限期下，输入时间长短，单位为秒。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后必须等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

Amazon CLI

为新组开启 Amazon EBS 运行状况检查

将该`--health-check-type`选项添加到create-auto-scaling-group命令中。以下示例将名为 *my-asg* 的新自动扩缩组的 `--health-check-type` 选项指定为 **EBS**。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60 ...
```

您可以为 `--health-check-type` 选项指定多个值。例如，要同时添加 Amazon EBS 和 Elastic Load Balancing 运行状况检查类型，请使用以下命令。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS,ELB" --health-check-grace-period 60 ...
```

值的名称区分大小写。

Console

为现有组开启 Amazon EBS 运行状况检查

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
4. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
5. 对于运行状况检查、其他运行状况检查类型，选择开启 Amazon EBS 运行状况检查。
6. 对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后必须等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
7. 选择更新。

Amazon CLI

为现有组开启 Amazon EBS 运行状况检查

将该`--health-check-type`选项添加到update-auto-scaling-group命令中。以下示例将名为 *my-asg* 的现有自动扩缩组的 `--health-check-type` 选项指定为 **EBS**。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60
```

要使用多种运行状况检查类型，您可以为 `--health-check-type` 选项指定多个值（例如 **EBS,ELB**）。

值的名称区分大小写。

关闭自动扩缩组的 Amazon EBS 运行状况检查

以下主题描述了如何关闭自动扩缩组的 Amazon EBS 运行状况检查。如果您不再需要 Amazon EBS 运行状况检查，则请按照以下步骤将其关闭。

Console

关闭组的 Amazon EBS 运行状况检查

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
4. 对于运行状况检查、其他运行状况检查类型，取消选择开启 Amazon EBS 运行状况检查。
5. 选择更新。

Amazon CLI

关闭组的 Amazon EBS 运行状况检查

要更新 Auto Scaling 组的运行状况检查，使其不再使用 Amazon EBS 运行状况检查，请使用update-auto-scaling-group命令。包括 `--health-check-type` 选项和 **`EC2`** 的值。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

要在不关闭其他运行状况检查类型（例如 Elastic Load Balancing）的情况下关闭 Amazon EBS 运行状况检查，则必须指定类型而不是使用 **`EC2`**。例如，对于 Elastic Load Balancing 运行状况检查，请为 `--health-check-type` 选项指定 **`ELB`**。

值的名称区分大小写。

为您的自动扩缩组设置自定义运行状况检查

您可以使用自定义运行状况检查来补充 Amazon A EC2 uto Scaling 提供的现有运行状况检查选项。通过将自定义运行状况检查与其他运行状况检查类型相结合，您可以创建针对应用程序需求量身定制的全面运行状况监控系统。

首先，创建自定义测试，以验证自动扩缩组中的实例是否正常运行并且可以处理传入流量。如果您配置的运行状况检查检测到某个实例没有响应，请将该特定实例标记为Unhealthy，这会导致 Amazon A EC2 uto Scaling 立即将其替换。

您可以使用 Amazon CLI 或软件开发工具包将实例的运行状况直接发送到 Amazon A EC2 uto Scaling。以下示例向您展示了如何使用 Amazon CLI 来配置实例的运行状况，然后验证实例的运行状况。

使用以下set-instance-health命令将指定实例的运行状况设置为**Unhealthy**。

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-
status Unhealthy
```

默认情况下，此命令会执行运行状况检查宽限期。不过，您可以通过包括 `--no-should-respect-grace-period` 选项来覆盖此行为，不执行此宽限期。

使用以下describe-auto-scaling-groups命令验证实例的运行状况是否为Unhealthy。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

下面是一个示例响应，向您演示实例的运行状况为 Unhealthy 并且实例正在终止。

```
{
    "AutoScalingGroups": [
        {
            ....
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcdef0"
                    },
                    "InstanceId": "i-1234567890abcdef0",
                    "InstanceType": "t2.micro",
                    "HealthStatus": "Unhealthy",
                    "LifecycleState": "Terminating"
                },
                ...
            ]
        }
    ]
}
```

查看运行状况检查失败原因

您可以使用以下过程来查看有关因运行状况检查而被替换的任何实例的信息。

默认情况下，Amazon A EC2 uto Scaling 会创建一个新的扩展活动来终止运行状况不佳的实例，然后将其终止。在实例终止期间，另一个扩缩活动将会启动一个新实例。您可以使用实例维护策略将此行为更改为尽快开始启动新实例。有关更多信息，请参阅 实例维护策略。

Console

查看运行状况检查失败原因

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中 Auto Scaling 组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在活动选项卡的活动历史记录下，状态列显示您的 Auto Scaling 组是否已成功启动或终止实例。

   如果它终止了任何运行状况不佳的实例，原因列显示终止的日期和时间以及运行状况检查失败的原因。例如，`At 2022-05-14T20:11:53Z an instance was taken out of service in response to a user health-check`。此消息表示自定义运行状况检查将实例标记为运行状况不佳。

   有关运行状况检查失败的帮助，请参阅对 Amazon A EC2 uto Scaling 中运行不正常的实例进行故障排除。

Amazon CLI

查看运行状况检查失败原因

使用以下 describe-scaling-activities 命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下为示例响应，其中 `Cause` 包含运行状况检查失败的原因。

```
{
  "Activities": [
```

```
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to a user health-check.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

有关输出中字段的描述，请参阅《Amazon A EC2 uto Scaling API 参考》中的 “活动”。

要描述删除 Auto Scaling 组后的伸缩活动，describe-scaling-activities请在命令中添加`--include-deleted-groups`选项。

对 Amazon A EC2 uto Scaling 中运行不正常的实例进行故障排除

以下是 Amazon A EC2 uto Scaling 返回的错误消息、潜在原因以及您可以采取的解决问题的步骤。

要检索错误消息，请参阅查看运行状况检查失败原因。

错误消息

- 由于实例状态检查失败，EC2 实例已停止服务
- 由于运行 EC2 状况检查显示该实例已终止或停止，该实例已停止服务
- 实例因 ELB 系统运行状况检查失败而停止服务
- 其他资源

由于实例状态检查失败，EC2 实例已停止服务

问题：Auto Scaling 实例未通过亚马逊 EC2 状态检查。

原因 1：如果存在导致亚马逊 EC2 认为您的 Auto Scaling 组中的实例受损的问题，Amazon A EC2 uto Scaling 将在运行状况检查中自动替换这些实例。

解决方案 1：当实例状态检查失败时，通常必须通过更改实例配置来自行解决问题，直到应用程序不再出现任何问题。要解决该问题，请完成以下步骤：

1. 手动创建不属于 Auto Scaling 组的 Amazon EC2 实例，然后调查问题。有关调查受损实例的一般帮助，请参阅 Amazon EC2 用户指南中的对状态检查失败的实例进行故障排除。
2. 在确认实例已成功启动且运行状况良好后，请将新的、无错误的实例配置部署到 Auto Scaling 组。
3. 删除您创建的实例以免对 Amazon 账户继续产生费用。

由于运行 EC2 状况检查显示该实例已终止或停止，该实例已停止服务

问题：已停止、重新启动或终止的 Auto Scaling 实例将被替换。

原因 1：用户已手动停止、重新启动或终止实例。

解决方案 1：如果您需要停止或重启自动扩缩组中的实例，则建议您首先将实例置于备用状态。有关更多信息，请参阅 临时从 Auto Scaling 组中移除实例。

原因 2：在亚马逊竞价服务中断竞价型实例后，Amazon A EC2 uto Scaling 会尝试替换 EC2 竞价型实例，因为竞价价格上涨幅度超过您的最高价格或容量不再可用。

解决方案 2：无法保证在任何给定时间点存在竞价型实例来满足请求。但是，您可以尝试以下操作：

- 使用更高的 Spot 最高价（可能是按需价格）。通过将最高价格设置得更高，Amazon EC2 Spot 服务有更好的机会启动和维持所需的容量。
- 通过在多个可用区中运行多个实例类型，增加您可以从中启动实例的不同容量池的数量。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。
- 如果您使用多个实例类型，请考虑启用容量再平衡功能。如果您希望 Amazon EC2 Spot 服务在正在运行的实例终止之前尝试启动新的竞价型实例，这将非常有用。有关更多信息，请参阅 在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例。

原因 3：对于容量块，Amazon 会在容量块结束前 30 分钟 EC2 终止所有仍在运行的实例。这种突然终止会导致您的自动扩缩组尝试启动新实例以维持其所需容量，即使容量块即将结束。

解决方案 3：要解决此问题，请尝试以下操作：

- 减少自动扩缩组的所需容量，以防止其尝试启动新实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的手动扩展。
- 请务必在容量块结束时间前 30 分钟横向缩减您的自动扩缩组，这样就不会频繁遇到此错误。确保所有生命周期挂钩在容量块结束时间前 30 分钟完成。有关更多信息，请参阅 使用 Capacity Blocks 适用于机器学习工作负载。

实例因 ELB 系统运行状况检查失败而停止服务

问题：Auto Scaling 实例可能通过 EC2 状态检查。但是，它们可能无法对已注册 Auto Scaling 组的目标组或经典负载均衡器进行 Elastic Load Balancing 运行状况检查。

原因 1：如果您的 Auto Scaling 组依赖于 Elastic Load Balancing 提供的运行状况检查，则 Amazon A EC2 uto Scaling 会通过检查状态检查和 Elastic Load Balancing 运行 EC2 状况检查的结果来确定您的实例的运行状况。负载均衡器通过向每个实例发送请求并等待正确响应或与实例建立连接来执行运行状况检查。实例未能通过 Elastic Load Balancing 运行状况检查，可能是因为实例中运行的应用程序发生问题，导致负载均衡器将实例视为停止服务。

解决方案 1：要通过 Elastic Load Balancing 运行状况检查，请执行以下操作：

- 验证目标组的运行状况检查设置是否已正确配置。您定义每个目标组的负载均衡器的运行状况检查设置。有关更多信息，请参阅 配置目标的运行状况检查。
- 记住负载均衡器所需的成功代码，并且验证应用程序已正确配置为在成功时返回这些代码。
- 验证负载均衡器和 Auto Scaling 组的安全组是否已正确配置。
- 验证负载均衡器是否配置在与 Auto Scaling 组相同的可用区中。

解决方案 2：更新 Auto Scaling 组以禁用 Elastic Load Balancing 运行状况检查。有关如何禁用这些运行状况检查的说明，请参阅将 Elastic Load Balancing 负载均衡器附加到自动扩缩组。

原因 2：运行状况检查宽限期与实例启动时间不匹配。

解决方案 3：编辑自动扩缩组的运行状况检查宽限期。将宽限期设置为足够长的时间段，以支持 Elastic Load Balancing 认为新启动的实例正常之前运行状况检查所需的连续成功次数。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

其他资源

如果您遇到其他问题，请参阅以下 Amazon Web Services re:Post 文章以获取更多疑难解答帮助：

- 为什么 Amazon A EC2 uto Scaling 终止了一个实例？

- 为什么 Amazon A EC2 uto Scaling 没有终止运行状况不佳的实例？

Amazon Health Dashboard 亚马逊 A EC2 uto Scaling 的通知

您的 Amazon Health Dashboard 支持来自 Amazon A EC2 uto Scaling 的通知。针对可能影响应用程序的资源性能或者可用性问题，这些通知可以让您注意相关情况并提供修正指导。当前只有特定于缺少安全组和启动模板的事件可用。

Amazon Health Dashboard 是 Amazon Health 服务的一部分。它不需要设置，您的账户中通过身份验证的任何用户都可以查看。有关更多信息，请参阅Amazon Health 控制面板入门。

如果您收到类似于下文的消息，则应将其视为警报并采取措施。

示例：由于缺少安全组，Auto Scaling 组未向外扩展

```
 Hello,

 At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
Amazon Web Services ## 123456789012.

 A security group associated with this Auto Scaling group cannot be found. Each time
a
 scale out operation is performed, it will be prevented until you make a change that
 fixes the issue.

 We recommend that you review and update your Auto Scaling group configuration to
change
 the launch template or launch configuration that depends on the unavailable security
 group.

 Sincerely,
 Amazon Web Services
```

示例：由于缺少启动模板，Auto Scaling 组未向外扩展

```
 Hello,

 At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
Amazon Web Services ## 123456789012.

 The launch template associated with this Auto Scaling group cannot be found. Each
time
```

```
 a scale out operation is performed, it will be prevented until you make a change
that
 fixes the issue.

 We recommend that you review and update your Auto Scaling group configuration and
 specify an existing launch template to use.

 Sincerely,
 Amazon Web Services
```

CloudWatch 监控您的 Auto Scaling 组和实例的指标

指标是 Amazon 的基本概念 CloudWatch。指标表示发布到的一组按时间顺序排列的数据点。CloudWatch可将指标视为要监控的变量，而数据点代表该变量随时间变化的值。您可使用这些指标来验证您的系统是否按预期运行。

收集有关 A EC2 uto Scaling 群组信息的 Amazon Auto Scaling 指标位于`AWS/AutoScaling`命名空间中。从 Auto Scaling EC2 实例中收集 CPU 和其他使用率数据的亚马逊实例指标位于`AWS/EC2`命名空间中。

Amazon A EC2 uto Scaling 控制台显示了一系列图表，分别显示组指标和该组的汇总实例指标。根据您的需求，您可能更愿意从亚马逊访问您的 Auto Scaling 组和实例的数据， CloudWatch 而不是 Amazon A EC2 uto Scaling 控制台。

有关更多信息，请参阅 Amazon CloudWatch 用户指南。

内容

- 在 Amazon A EC2 uto Scaling 控制台中查看监控图表
- 亚马逊 A EC2 uto Scaling 的亚马逊 CloudWatch 指标
- 配置 Auto Scaling 实例的监控

在 Amazon A EC2 uto Scaling 控制台中查看监控图表

在亚马逊 EC2 控制台的 Amazon A EC2 uto Scaling 部分，您可以使用 CloudWatch 指标监控各个 Auto Scaling 组的 minute-by-minute进度。

您可以监控如下指标：

- Auto Scaling 指标 – 只有在启用的情况下才会开启这些 Auto Scaling 指标。有关更多信息，请参阅启用 Auto Scaling 组指标（控制台）。启用 Auto Scaling 指标后，监控图表显示以一分钟粒度发布的 Auto Scaling 指标数据。
- EC2 指标 — Amazon EC2 实例指标始终处于启用状态。如果启用了详细监控，监控图表将显示以一分钟精度发布的实例指标数据。有关更多信息，请参阅 配置 Auto Scaling 实例的监控。

使用 Amazon A EC2 uto Scaling 控制台查看监控图表

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中要查看其指标的 Auto Scaling 组旁边的复选框。

   将在 Auto Scaling 组页面底部打开一个拆分窗格。
3. 选择监控选项卡。

   Amazon A EC2 uto Scaling 会显示 A uto Scaling 指标的监控图表。
4. 要查看该组的聚合实例指标的监控图表，请选择EC2。

图表操作

- 将鼠标悬停在数据点上，可以查看 UTC 中特定时间的数据弹出窗口。
- 要放大图表，请从图表右上角的菜单工具选择 Enlarge (放大)（三个垂直点）。或者，选择图表顶部的最大化图标。
- 通过选择一个预定义的时间段值，调整图表中所显示数据的时间段。如果图表已放大，您可以选择 Custom (自定义) 来定义自己的时间段。
- 从菜单工具中选择 Refresh (刷新) 以更新图表中的数据。
- 将光标拖动到图表数据上以选择特定范围。然后，您可以选择工具菜单中的 Apply time range (应用时间范围)。
- 从菜单工具中选择 “查看日志”，在 CloudWatch 控制台中查看关联的日志流（如果有）。
- 要在中查看图表 CloudWatch，请从菜单工具中选择在指标中查看。这会将您带到该图表的 CloudWatch 页面。从中可以查看更多信息或访问历史信息，以更好地了解 Auto Scaling 组长时间内的变化情况。

适用于 Auto Scaling 组的图表指标

创建 Auto Scaling 组后，您可以打开 Amazon A EC2 uto Scaling 控制台，然后在 “监控” 选项卡上查看该组的监控图表。

在 Auto Scaling 章节中，图表指标包括以下指标。这些指标提供的测量值可以指示潜在的问题，例如终止实例的数量或挂起实例的数量。您可以在 亚马逊 A EC2 uto Scaling 的亚马逊 CloudWatch 指标中找到这些指标的定义。

显示名称	CloudWatch 指标名称
最小组大小	GroupMinSize
最大组大小	GroupMaxSize
所需容量	GroupDesiredCapacity
在服务实例中	GroupInServiceInstances
待处理实例	GroupPendingInstances
备用实例	GroupStandbyInstances
正在终止实例	GroupTerminatingInstances
实例总数	GroupTotalInstances

EC2在本节中，您可以根据您的 Amazon EC2 实例的关键性能指标找到以下图表指标。这些 EC2 指标是该组中所有实例的指标的汇总。您可以在亚马逊 EC2 用户指南中的列出您的实例的可用 CloudWatch 指标中找到这些指标的定义。

显示名称	CloudWatch 指标名称
CPU 利用率	CPUUtilization
磁盘读取	DiskReadBytes
读磁盘操作	DiskReadOps

显示名称	CloudWatch 指标名称
磁盘写入	DiskWriteBytes
磁盘写入操作	DiskWriteOps
网络输入	NetworkIn
网络输出	NetworkOut
Status Check Failed (Any) (状态检查失败(任意))	StatusCheckFailed
Status Check Failed (Instance) (状态检查失败(实例))	StatusCheckFailed_Instance
Status Check Failed (System) (状态检查失败(系统))	StatusCheckFailed_System

此外，自动扩缩图表指标中还有一些指标可用于特定使用案例。

如果您的组使用权重来定义每个实例为组的所需容量贡献多少单位，则以下图表指标非常有用。您可以在 亚马逊 A EC2 uto Scaling 的亚马逊 CloudWatch 指标 中找到这些指标的定义。

显示名称	CloudWatch 指标名称
在服务容量单位中	GroupInServiceCapacity
待处理的容量单位	GroupPendingCapacity
备用容量单位	GroupStandbyCapacity
终止容量单位	GroupTerminatingCapacity
总容量单位	GroupTotalCapacity

如果您的组使用暖池功能，则以下指标非常有用。您可以在 亚马逊 A EC2 uto Scaling 的亚马逊 CloudWatch 指标 中找到这些指标的定义。

显示名称	CloudWatch 指标名称
暖池最小尺寸	`WarmPoolMinSize`
暖池所需容量	`WarmPoolDesiredCapacity`
暖池挂起容量单位	`WarmPoolPendingCapacity`
暖池终止容量单位	`WarmPoolTerminatingCapacity`
暖池预热容量单位	`WarmPoolWarmedCapacity`
已启动的暖池总容量单位	`WarmPoolTotalCapacity`
组和暖池所需容量	`GroupAndWarmPoolDesiredCapacity`
已启动的组和暖池总容量单位	`GroupAndWarmPoolTotalCapacity`

相关资源

- 要监控每个实例的指标，请参阅 Amazon EC2 用户指南中的实例图表指标。
- CloudWatch 仪表板是 CloudWatch 控制台中可自定义的主页。您可以使用这些页面在单个视图中监控您的资源，甚至包括分布在不同区域的资源。您可以使用 CloudWatch 仪表板为您的 Amazon 资源创建指标和警报的自定义视图。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

亚马逊 A EC2 uto Scaling 的亚马逊 CloudWatch 指标

Amazon A EC2 uto Scaling 在`AWS/AutoScaling`命名空间中发布以下指标。实际可用的自动扩缩组指标将取决于您是否启用了组指标以及启用了哪些组指标。组指标以一分钟的粒度提供，无需额外付费，但您必须启用它们。

当您启用 Auto Scaling 组指标时，Amazon A EC2 uto Scaling 会尽力向 CloudWatch 每分钟发送一次采样数据。在极少数情况下， CloudWatch 当服务中断时，数据不会被回填以填补组指标历史记录中的空白。

内容

- 预测性扩缩指标和维度
- 启用 Auto Scaling 组指标（控制台）
- 启用 Auto Scaling 组指标 (Amazon CLI)

自动扩缩组指标

借助这些指标，您可以更清楚地了解自动扩缩组的历史记录，例如组大小随时间推移的变化。

指标	描述
GroupMinSize	自动扩缩组的最小大小。 报告条件：如果启用了指标收集，则报告。
GroupMaxSize	自动扩缩组的最大大小。 报告条件：如果启用了指标收集，则报告。
GroupDesiredCapacity	Auto Scaling 组试图维护的实例数量。 报告条件：如果启用了指标收集，则报告。
GroupInServiceInstances	作为 Auto Scaling 组的一部分运行的实例数量。该指标不包括处于挂起或终止状态的实例。 报告条件：如果启用了指标收集，则报告。
GroupPendingInstances	处于挂起状态的实例数量。挂起的实例尚不可用。该指标不包括处于可用状态或终止状态的实例。 报告条件：如果启用了指标收集，则报告。
GroupStandbyInstances	处于 Standby 状态的实例数。处于此状态的实例仍在运行，但不能有效使用。 报告条件：如果启用了指标收集，则报告。
GroupTerminatingInstances	正处于终止过程中的实例的数量。此指标不包括处于运行状态、待处理状态或在 Auto Scaling 组缩减后返回到温池的实例。

指标	描述
	报告条件：如果启用了指标收集，则报告。
GroupTotalInstances	Auto Scaling 组中的实例总数。该指标用于标识处于可用状态、挂起状态和终止状态的实例的数量。 报告条件：如果启用了指标收集，则报告。

当您将混合实例组配置为以不同单位衡量其所需容量时，例如根据每种实例类型的 vCPU 数量分配权重，那么以下指标会计算您的自动扩缩组使用的单位数。如果您未配置混合实例组以不同单位衡量其所需容量，则会填充以下指标，但等于上表中定义的指标。有关更多信息，请参阅 创建混合实例组的设置概述。

指标	描述
GroupInServiceCapacity	作为 Auto Scaling 组的一部分运行的容量单位数量。 报告条件：如果启用了指标收集，则报告。
GroupPendingCapacity	待处理的容量单位数。 报告条件：如果启用了指标收集，则报告。
GroupStandbyCapacity	处于 Standby 状态的容量单位数。 报告条件：如果启用了指标收集，则报告。
GroupTerminatingCapacity	正处于终止过程中的容量单位的数量。 报告条件：如果启用了指标收集，则报告。
GroupTotalCapacity	Auto Scaling 组中的容量单位总数。 报告条件：如果启用了指标收集，则报告。

Amazon A EC2 uto Scaling 还会报告具有温池的 Auto Scaling 群组的以下指标。有关更多信息，请参阅 使用暖池减少启动时间较长的应用程序的延迟。

指标	描述
`WarmPoolMinSize`	暖池的最小大小。 报告条件：如果启用了指标收集，则报告。
`WarmPoolDesiredCapacity`	Amazon A EC2 uto Scaling 尝试在温池中保持的容量。 这相当于 Auto Scaling 组的最大大小减去所需容量，或者等于 Auto Scaling 组的最大预热容量（如果设置）减去所需容量。 但是，如果暖池的最小大小等于或大于最大大小（或者是最大预热容量（如果设置））与 Auto Scaling 组的所需容量之间的差值，则暖池所需容量将等于 `WarmPoolMinSize` 。 报告条件：如果启用了指标收集，则报告。
`WarmPoolPendingCapacity`	暖池中待处理的容量数。这包括在 Auto Scaling 组缩容后返回到温池的实例。该指标不包括处于运行、挂起或终止状态的实例。 报告条件：如果启用了指标收集，则报告。
`WarmPoolTerminatingCapacity`	暖池中处于终止过程的容量数。该指标不包括处于运行、已停止或挂起状态的实例。 报告条件：如果启用了指标收集，则报告。
`WarmPoolWarmedCapacity`	横向扩展期间可进入 Auto Scaling 组的容量数。该指标不包括处于挂起或终止状态的实例。 报告条件：如果启用了指标收集，则报告。
`WarmPoolTotalCapacity`	暖池的总容量，包括处于运行、已停止、挂起或终止状态的实例。 报告条件：如果启用了指标收集，则报告。
`GroupAndWarmPoolDesiredCapacity`	Auto Scaling 组和暖池结合起来的所需容量。 报告条件：如果启用了指标收集，则报告。

指标	描述
GroupAndWarmPoolTotalCapacity	Auto Scaling 组和暖池结合起来的总容量。这包括处于运行、已停止、挂起、终止或服务中状态的实例。 报告条件：如果启用了指标收集，则报告。

Auto Scaling 组指标的维度

您可以使用以下维度来优化上表中列出的指标。

维度	描述
AutoScalingGroupName	按 Auto Scaling 组名称筛选。

预测性扩缩指标和维度

AWS/AutoScaling 命名空间包括以下预测性扩缩指标。

提供精度为一小时的指标。

您可以通过将预测值与实际值进行比较来评估预测准确性。有关使用这些指标评估预测准确性的更多信息，请参阅使用监控预测性扩展指标 CloudWatch。

指标	描述	Dimensions
PredictiveScalingLoadForecast	应用程序预计将生成的负载量。 Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	AutoScalingGroupName，PolicyName，PairIndex
PredictiveScalingCapacityForecast	满足应用程序需求所需的预期容量。这基于负载预测以及您希望 Auto Scaling 实例维持的目标利用率水平。	AutoScalingGroupName，PolicyName

指标	描述	Dimensions
	Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	
PredictiveScalingMetricPairCorrelation	扩展指标与负载指标的每个实例平均值之间的相关性。预测性扩展假设相关性很高。因此，如果您观察到该指标的值很低，最好不要使用指标对。 Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	AutoScalingGroupName，PolicyName，PairIndex

> **Note**
>
> 该PairIndex维度返回与 Amazon A EC2 uto Scaling 分配的负载扩展指标对的索引相关的信息。目前唯一有效的值是 0。

启用 Auto Scaling 组指标（控制台）

启用组指标

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在监控选项卡上，选择位于 Auto Scaling下页面顶部的 Auto Scaling 组指标集合、启用复选框。

禁用组指标

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。

2. 选择您的 Auto Scaling 组。
3. 在监控选项卡上，清除 Auto Scaling 组指标集合、启用复选框。

启用 Auto Scaling 组指标 (Amazon CLI)

如需启用自动扩缩组指标

使用enable-metrics-collection命令启用一个或多个群组指标。例如，以下命令可为指定的自动扩缩组启用单个指标。

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --metrics GroupDesiredCapacity --granularity "1Minute"
```

如果省略 `--metrics` 选项，则启用所有指标。

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --granularity "1Minute"
```

如需禁用自动扩缩组指标

使用disable-metrics-collection命令禁用所有群组指标。

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

配置 Auto Scaling 实例的监控

Amazon EC2 收集来自实例的原始数据并将其处理为可读的、近乎实时的指标，这些指标描述了您的 Auto Scaling 组的 CPU 和其他使用率数据。对于间隔时间的配置，您可以选择按一分钟或五分钟的精度来监控这些指标。

每次启动实例时都会启用实例监控，可选择基本监控（五分钟精度）或详细监控（一分钟精度）。对于详细监控，将收取额外的费用。有关更多信息，请参阅亚马逊 EC2 用户指南 CloudWatch中的亚马逊 CloudWatch 定价和监控您的实例。

您首先需要创建一个启动模板或启动配置，以允许适合您的应用程序的监控类型，然后再创建自动扩缩组。如果您向组添加扩展策略，我们强烈建议您使用详细监控来获取每分钟粒度的 EC2 实例指标数据，因为这样可以更快地响应负载变化。

内容

启用详细监控（控制台）

默认情况下，当您使用创建启动模板或启动配置时，基本监控处于启用状态。 Amazon Web Services Management Console

在启动模板中启用详细监控

使用创建启动模板时 Amazon Web Services Management Console，在高级详细信息部分中，对于详细 CloudWatch监控，选择启用。否则，将启用基本监控。有关更多信息，请参阅 使用高级设置创建启动模板。

在启动配置中启用详细监控

使用创建启动配置时 Amazon Web Services Management Console，在 “其他配置” 部分中，选择在其中启用 EC2 实例详细监控 CloudWatch。否则，将启用基本监控。有关更多信息，请参阅 创建启动配置。

启用详细监控 (Amazon CLI)

预设情况下，使用 Amazon CLI创建启动模板时，将启用基本监控。在使用 Amazon CLI创建启动配置时，将默认启用详细监控。

在启动模板中启用详细监控

对于启动模板，请使用 create-launch-template 命令，并传递一个包含用于创建启动模板的信息的 JSON 文件。将监控属性设置为 `"Monitoring":{"Enabled":true}` 以启用详细监控，或设置为 `"Monitoring":{"Enabled":false}` 以启用基本监控。

在启动配置中启用详细监控

对于启动配置，请将 create-launch-configuration 命令与 `--instance-monitoring` 选项一起使用。将此选项设置为 `true` 可启用详细监控，将此选项设置为 `false` 可启用基本监控。

```
--instance-monitoring Enabled=true
```

在基本监控和详细监控之间切换

要更改在新 EC2 实例上启用的监控类型，请更新启动模板或更新 Auto Scaling 组以使用新的启动模板或启动配置。现有实例将继续使用以前启用的监控类型。要更新所有实例，请终止这些实例，以便您的 Auto Scaling 组可以替换这些实例，或使用 monitor-instances 和 unmonitor-instances 逐一更新实例。

> **Note**
>
> 借助实例刷新和最长实例生命周期和功能，您还可以替换 Auto Scaling 组中的所有实例，以启动使用新设置的新实例。有关更多信息，请参阅 替换自动扩缩组中的实例。

在基本监控和详细监控之间切换时：

如果您有与 Auto Scaling 组的分步扩展策略或简单扩展策略关联的 CloudWatch 警报，请使用put-metric-alarm命令更新每个警报。使每个时段与监控类型匹配（基本监控为 300 秒，详细监控为 60 秒）。如果从详细监控更改为基本监控，但未更新警报以与 5 分钟时间段匹配，这些警报将继续每分钟检查一次统计数据。在每个 5 分钟时间段内，这些警报可能会在 4 分钟内找不到可用的数据。

使用 CloudWatch 代理收集其他指标

要收集操作系统级别的指标，例如可用内存和已用内存，必须安装 CloudWatch 代理。可能会产生额外的费用。您可以使用 CloudWatch 代理从 Amazon EC2 实例收集系统指标和日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的 CloudWatch 代理收集的指标。

使用记录 Amazon A EC2 uto Scaling API 调用 Amazon CloudTrail

Amazon A EC2 uto Scaling 与Amazon CloudTrail一项服务集成，该服务提供用户、角色或角色所执行操作的记录 Amazon Web Services 服务。 CloudTrail 将 Auto Scaling 的 API 调用捕获为事件。捕获的调用包含来自 Amazon Web Services Management Console 的调用和对 Auto Scaling API 操作的代码调用。使用收集的信息 CloudTrail，您可以确定向 Auto Scaling 发出的请求、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。

- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon Web Services 服务发出。

CloudTrail 在您创建账户 Amazon Web Services 账户 时在您的账户中处于活动状态，并且您自动可以访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 Amazon Web Services 区域有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 Amazon Web Services 账户 过去 90 天内的事件，请创建跟踪。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 Amazon Web Services Management Console 都是多区域的。您可以通过使用 Amazon CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可以捕获账户 Amazon Web Services 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 Amazon Web Services 区域中记录的事件。有关跟踪的更多信息，请参阅《Amazon CloudTrail 用户指南》中的为您的 Amazon Web Services 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅Amazon CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

中的 Auto Scaling 管理事件 CloudTrail

管理事件提供有关对中的资源执行的管理操作的信息 Amazon Web Services 账户。这些也称为控制面板操作。默认情况下， CloudTrail 记录管理事件。

Amazon A EC2 uto Scaling 将所有 Auto Scaling 控制平面操作记录为管理事件。有关 Auto Scaling 记录的 Amazon A EC2 uto Scaling 控制平面操作的列表 CloudTrail，请参阅 Amazon A EC2 uto Scaling API 参考。

Auto Scaling 事件示例

事件代表来自任何来源的单个请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序出现。

以下示例显示了一个演示该CreateLaunchConfiguration操作 CloudTrail 的事件。


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
    },
    "eventTime": "2018-08-21T17:07:49Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "CreateLaunchConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "ebsOptimized": false,
        "instanceMonitoring": {
            "enabled": false
        },
        "instanceType": "t2.micro",
        "keyName": "EC2-key-pair-oregon",
        "blockDeviceMappings": [
            {
                "deviceName": "/dev/xvda",
                "ebs": {
                    "deleteOnTermination": true,
                    "volumeSize": 8,
                    "snapshotId": "snap-01676e0a2c3c7de9e",
                    "volumeType": "gp2"
                }
            }
        ],
        "launchConfigurationName": "launch_configuration_1",
        "imageId": "ami-6cd6f714d79675a5",
        "securityGroups": [
```

```
            "sg-00c429965fd921483"
        ]
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

有关 CloudTrail 录音内容的信息，请参阅《Amazon CloudTrail 用户指南》中的CloudTrail录制内容。

Auto Scaling RemoveAction 正在调用 CloudWatch

您的 Amazon CloudTrail 日志可能会显示，当 Auto Scaling 指示从警报中删除自动扩展操作时 CloudWatch ，Auto Scaling 会调用 CloudWatch `RemoveAction` API。如果您取消注册可扩展目标、删除扩展策略或警报调用了不存在的扩展策略，则可能会发生这种情况。

亚马逊 Auto Scaling 的亚马 EC2 逊 SNS 通知选项

可以配置自动扩缩组以通知您影响应用程序的重要事件。如果使用通知，您还可以不再使用轮询，并且不会遇到有时由轮询引起的 `RequestLimitExceeded` 错误。

有两种方法可以接收有关 Amazon A EC2 uto Scaling 的通知：

- Amazon Simple Notification Service：Amazon SNS 可以在自动扩缩组启动或终止实例时通知您。您只能打开或关闭 Amazon SNS 通知。有关更多信息，请参阅 亚马逊 SNS 和亚马逊 Auto Scalin EC2 g。
- 亚马逊 EventBridge — EventBridge 提供更高级的事件驱动型通知，这些通知符合指定标准，并发送到各种目标，包括 Amazon SNS。 EventBridge 还可以监控更广泛的 Auto Scaling 事件，以实现更精确的监控。有关更多信息，请参阅 用于处理 EventBridge Auto Scaling 事件。

您可以选择使用带有生命周期挂钩的通知，以在启动或终止期间对实例执行自定义操作。有关如何配置通知以与生命周期挂钩配合使用的更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

亚马逊 SNS 和亚马逊 Auto Scalin EC2 g

本节介绍如何使用 Amazon SNS 监控自动扩缩组何时启动或终止实例。

例如，如果将 Auto Scaling 组配置为使用 `autoscaling: EC2_INSTANCE_TERMINATE` 通知类型，并且您的 Auto Scaling 组终止了某个实例，则它会发送电子邮件通知。该电子邮件包含已终止实例的详细信息，如实例 ID 以及终止该实例的原因。

请注意，当 Amazon A EC2 uto Scaling 在组中添加或移除实例时，系统会向您发送有关这些更改的通知，每个实例只发送一条通知。但是，这些通知的发送基于尽力原则，您的实例在初始通知后仍可能失败，例如，稍后的运行状况检查失败。有关运行状况检查流程的更多信息，请参阅自动扩缩组中实例的运行状况检查。

有关 Amazon SNS 一般情况的更多信息，请参阅《Amazon Simple Notification Service Developer Guide》https://docs.amazonaws.cn/sns/latest/dg/。

内容

SNS 通知

Amazon A EC2 uto Scaling 支持在发生以下事件时发送亚马逊 SNS 通知。

事件	描述
`autoscaling:EC2_INSTANCE_LAUNCH`	实例启动成功
`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`	实例启动失败
`autoscaling:EC2_INSTANCE_TERMINATE`	实例终止成功

事件	描述
autoscaling:EC2_INSTANCE_TERMINATE_ERROR	实例终止失败

消息包含以下信息：

- Event — 事件。
- AccountId — 亚马逊云科技账户 ID。
- AutoScalingGroupName — Auto Scaling 组的名称。
- AutoScalingGroupARN — Auto Scaling 组的 ARN。
- EC2InstanceId— EC2 实例的 ID。

例如：

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints
 to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
Origin: AutoScalingGroup
Destination: EC2
```

为亚马逊 Auto Scaling 配置亚马 EC2 逊 SNS 通知

要使用 Amazon SNS 发送电子邮件通知，必须先创建一个主题，然后用您的电子邮件地址订阅该主题。

创建 Amazon SNS 主题

SNS 主题是一个逻辑接入点，即 Auto Scaling 组用来发送通知的通信通道。您可通过为主题指定名称来创建主题。

您在创建主题名称时，该名称必须满足以下要求：

- 介于 1 到 256 个字符之间
- 包含大写和小写 ASCII 字母、数字、下划线或连字符

有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的创建 Amazon SNS 主题。

订阅 Amazon SNS 主题

要接收您的 Auto Scaling 组发送到该主题的通知，必须让一个终端节点订阅该主题。在此过程中，对于 Endpoint，指定您要从中接收来自 Amazon A EC2 uto Scaling 的通知的电子邮件地址。

有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的订阅 Amazon SNS 主题。

确认您的 Amazon SNS 订阅

Amazon SNS 向在上一步骤中指定的电子邮件地址发送确认电子邮件。

确保打开并选择链接以确认订阅，然后再继续执行下一步。

您将收到来自的确认消息。 Amazon Amazon SNS 现已配置为接收通知并以电子邮件形式将通知发送到指定的电子邮件地址。

配置 Auto Scaling 组以发送通知

您可以配置 Auto Scaling 组，以便在发生扩展事件（例如，启动实例或终止实例）时向 Amazon SNS 发送通知。Amazon SNS 向您指定的电子邮件地址发送通知，通知中包含有关实例的信息。

为 Auto Scaling 组配置 Amazon SNS 通知（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   将在页面底部打开一个拆分窗格，其中显示有关所选组的信息。
3. 在活动选项卡上，选择活动通知、创建通知。
4. 在 Create notifications 窗格上，执行以下操作：

   a. 对于 SNS 主题，选择您的 SNS 主题。
   b. 对于事件类型，选择要发送通知的事件。
   c. 选择创建。

为 Auto Scaling 组配置 Amazon SNS 通知 (Amazon CLI)

使用以下 put-notification-configuration 命令。

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-
asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH"
 "autoscaling:EC2_INSTANCE_TERMINATE"
```

测试通知

要为启动事件生成通知，请通过将 Auto Scaling 组的所需容量增加 1 来更新 Auto Scaling 组。您将在实例启动后的几分钟内收到通知。

更改所需容量（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的 Auto Scaling 组旁边的复选框。

   将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格，其中显示有关所选组的信息。
3. 在 Details（详细信息）选项卡上，选择 Group details（组详细信息）、Edit (编辑)。
4. 对于 Desired capacity (所需容量)，将当前值增加 1。如果此值超过 Maximum capacity (最大容量)，则还必须将 Maximum capacity (最大容量) 的值增加 1。

5. 选择更新。
6. 在数分钟后，您将收到事件的通知。如果您不需要您为此测试启动的其他实例，则可以将 Desired capacity (所需容量) 减少 1。在数分钟后，您将收到事件的通知。

删除通知配置

如果不再使用您的 Amazon A EC2 uto Scaling 通知配置，则可以将其删除。

删除 Amazon A EC2 uto Scaling 通知配置（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择您的 Auto Scaling 组。
3. 在活动选项卡上，选中您要删除的通知旁边的复选框，然后选择操作、删除。

删除 Amazon A EC2 uto Scaling 通知配置 (Amazon CLI)

使用以下 delete-notification-configuration 命令。

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --
topic-arn arn
```

有关删除 Amazon SNS 主题以及与您的 Auto Scaling 组关联的所有订阅的信息，请参阅 Amazon Simple Notification Service 开发人员指南中的删除 Amazon SNS 订阅和主题。

加密 Amazon SNS 主题的密钥策略

您指定的 Amazon SNS 主题可能会使用通过 Amazon Key Management Service创建的客户托管密钥进行加密。要授予 Amazon A EC2 uto Scaling 发布加密主题的权限，您必须先创建 KMS 密钥，然后在 KMS 密钥的策略中添加以下声明。将示例 ARN 替换为允许访问密钥的相应服务相关角色的 ARN。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的配置 Amazon KMS 权限。

在此示例中，策略声明向名为的服务相关角色AWSServiceRoleForAutoScaling授予使用客户托管密钥的权限。要了解有关 Amazon A EC2 uto Scaling 服务相关角色的更多信息，请参阅Amazon A EC2 uto Scaling 的服务相关角色。

```
{
```

```
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

允许 Amazon A EC2 uto Scaling 向加密主题发布内容的密钥策略不支持和aws:SourceAccount条件密钥。aws:SourceArn

Amazon 与 Amazon A EC2 uto Scaling 集成的服务

Amazon A EC2 uto Scaling 可以与其他 Amazon 服务集成。查看以下集成选项，详细了解每项服务如何与 Amazon A EC2 uto Scaling 配合使用。

主题

在 Auto Scaling 中重新平衡容量以取代存在风险的竞价型实例

Auto Scaling 中的容量再平衡通过主动替换存在中断风险的竞价型实例来帮助您保持工作负载的可用性。

当竞价型实例的中断风险较高时，亚马逊 EC2 竞价服务会向 Amazon A EC2 uto Scaling 发送 EC2 实例再平衡建议。如果您启用容量再平衡，Auto Scaling 会尝试主动替换您组中已收到 EC2 实例再平衡建议的竞价型实例。这让您能够将工作负载转移到不具有较高中断风险的新竞价型实例。

当您不使用容量再平衡时，Auto Scaling 不会取代竞价型实例，直到 Amazon EC2 Spot 服务中断实例且其运行状况检查失败之后。在中断实例之前，Amazon EC2 始终会同时提供 EC2 实例再平衡建议和 Spot 两分钟实例中断通知。

内容

概览

要在自动扩缩组中使用容量再平衡，基本步骤为：

1. 将您的自动扩缩组配置为使用多种实例类型和多个可用区。这样，Amazon A EC2 uto Scaling 就可以查看每个可用区域中竞价型实例的可用容量。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。
2. 根据需要添加生命周期挂钩，以便在收到再平衡通知的实例内正常关闭应用程序。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

   以下是可能需要使用生命周期挂钩的一些原因：

   - 用于正常关闭 Amazon SQS 工件
   - 完成从域名系统 (DNS) 取消注册
   - 提取系统或应用程序日志并将其上传到 Amazon Simple Storage Service (Amazon S3)
3. 为生命周期挂钩开发自定义操作。若要尽快调用您的自定义操作，您需要知道实例何时可以终止。可以通过检测实例的生命周期状态来找出答案。
   - 要在实例外部调用操作，请编写一条 EventBridge 规则，并在事件模式与规则匹配时自动执行什么操作。
   - 要在实例内部调用操作，请将实例配置为运行关闭脚本并通过实例元数据检索生命周期状态。

   将自定义操作设计成在不到两分钟的时间内完成至关重要。这样可以确保在实例终止之前有足够的时间完成任务。

完成这些步骤后，可以开始使用容量再平衡。

容量再平衡行为

通过容量再平衡，当实例收到再平衡建议时，Amazon A EC2 uto Scaling 的行为方式如下：

- 当新的竞价型实例启动时，Amazon A EC2 uto Scaling 会等到新实例通过运行状况检查后才会终止先前的实例。替换多个实例时，每个旧实例的终止将在新实例启动并通过其运行状况检查后开始。
- 由于 Amazon A EC2 uto Scaling 会在终止之前尝试启动新实例，因此达到或接近指定的最大容量可能会阻碍或完全停止再平衡活动。为避免此问题，Amazon A EC2 uto Scaling 可以暂时超出组的最大容量，最多可超过所需容量的 10%。
- 如果您没有向 Auto Scaling 组添加生命周期挂钩，则在新实例通过运行状况检查后，Amazon A EC2 uto Scaling 会立即终止之前的实例。

- 如果您添加了生命周期挂钩，这将延长我们开始按您为生命周期挂钩指定的超时值终止之前的实例所花费的时间。
- 如果使用扩缩策略或计划缩，那么扩缩活动将并行运行。如果扩展活动正在进行中，并且您的 Auto Scaling 组低于其新的所需容量，则 Amazon A EC2 uto Scaling 会先进行扩展，然后再终止以前的实例。

如果您的实例类型在一个可用区域中没有容量，Amazon A EC2 uto Scaling 会继续尝试在其他已启用的可用区域中启动竞价型实例，直到成功为止。

在最坏的情况下，如果新实例无法启动或运行状况检查失败，Amazon A EC2 uto Scaling 会继续尝试重新启动它们。当它尝试启动新实例时，您之前的实例最终会被中断并强制终止，并发出两分钟的中断通知。

注意事项

使用容量再平衡时，请考虑以下因素：

将您的应用程序设计为可以容忍 Spot 中断

您的应用程序应该能够处理实例数量的动态变化以及竞价型实例提前中断的可能性。例如，如果您的 Auto Scaling 组位于 Elastic Load Balancing 负载均衡器之后，Amazon A EC2 uto Scaling 会等待实例从负载均衡器注销，然后再调用您的生命周期挂钩。如果注销实例和完成生命周期操作的时间过长，则实例可能会中断，而 Amazon A EC2 uto Scaling 会等待您的生命周期操作完成后再终止实例。

在两分钟竞价型实例中断 EC2 通知之前，Amazon 并不总是可能发送再平衡建议信号。有时候，再平衡建议信号可能会在两分钟的中断通知到达的同时一起到达。发生这种情况时，Amazon A EC2 uto Scaling 会调用生命周期挂钩并尝试立即启动新的竞价型实例。

避免替换竞价型实例中断的风险升高

如果您使用 `lowest-price` 分配策略，替换竞价型实例可能会面临中断升高的风险。这是因为，即使替换竞价型实例可能在启动后不久中断，我们也会在当时具有可用容量的价格最低池中启动实例。为避免中断风险增加，强烈建议您不要使用 `lowest-price` 分配策略。相反，我们建议使用 `price-capacity-optimized` 分配策略。此策略在中断可能性最小、价格尽可能最低的 Spot 池中启动替换竞价型实例。因此，它们在不久的将来不太可能被中断。

只有在可用性相同或更好的情况下，Amazon A EC2 uto Scaling 才会启动新实例

容量再平衡的目标之一是提高竞价型实例的可用性。如果现有竞价型实例收到再平衡建议，则只有在新实例提供与现有实例相同或更好的可用性时，Amazon A EC2 uto Scaling 才会启动新实例。如

果新实例的中断风险比现有实例更严重，那么 Amazon A EC2 uto Scaling 将不会启动新实例。但是，Amazon A EC2 uto Scaling 将继续根据亚马逊竞价服务提供的信息评估 EC2 竞价容量池，并在可用性提高时启动新实例。

如果没有 Amazon A EC2 uto Scaling 主动启动新实例，您的现有实例可能会被中断。发生这种情况时，Amazon A EC2 uto Scaling 会在收到竞价型实例中断通知后立即尝试启动新实例。无论新实例中断的风险是否很高，都会发生这种情况。

容量再平衡不会提高您的竞价型实例中断率

当您启用容量再平衡时，它不会增加您的竞价型实例中断率（Amazon EC2 需要恢复容量时回收的竞价型实例数量）。但是，如果容量再平衡检测到实例存在中断风险，Amazon A EC2 uto Scaling 将立即尝试启动新实例。因此，与您在风险实例中断后等待 Amazon A EC2 uto Scaling 启动新实例相比，替换的实例可能更多。

虽然您可能会在启用容量重新平衡的情况下替换更多的实例，但您会因为处于主动而非被动地位而受益。这使您在实例中断之前有更多时间采取行动。使用 Spot Instance interruption notice（竞价型实例中断通知），您通常最多只有两分钟的时间来正常关闭您的实例。借助容量再平衡提前启动新实例，可以让现有流程更有可能在有风险的实例上完成。您还可以启动实例关闭过程，防止在有风险的实例上安排新作业，并使新启动的实例做好接管应用程序的准备。借助容量再平衡的主动替换，您可以因绝佳的连续性而受益。

以下理论示例演示了使用容量再平衡的风险和优势：

- 下午 2:00 — 收到了重新平衡建议，例如 A。Amazon A EC2 uto Scaling 会立即尝试启动替换实例 B，这样您就可以有时间开始关闭程序。
- 下午 2:30 - 收到针对实例 B 的再平衡建议，该实例将被替换为实例 C，让您有时间启动关闭程序。
- 下午 2:32 - 如果未启用容量再平衡，并且实例 A 在下午 2:32 收到竞价型实例中断通知，则您只有两分钟的时间采取行动。但是，实例 A 将一直运行到此时。

启用容量再平衡以主动替换存在风险的 Spot 实例

您可以使用 Amazon Web Services Management Console 或 Amazon CLI 为 Auto Scaling 组启用容量重新平衡。启用容量再平衡后，Amazon A EC2 uto Scaling 会尝试主动替换您组中已收到 EC2 实例再平衡建议的竞价型实例。

启用容量再平衡（控制台）

您可以在创建或更新 Auto Scaling 组时启用或禁用容量再平衡。

为新的 Auto Scaling 组启用容量再平衡

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择 Create Auto Scaling group（创建 Auto Scaling 组）。
3. 对于步骤 1：选择启动模板或配置，为自动扩缩组输入一个名称，选择一个启动模板，然后选择Next（下一步）以继续执行下一步骤。
4. 对于步骤 2：选择实例启动选项以及对于实例类型要求，选择设置以创建混合实例组。这包括该实例可以启动的实例类型、实例购买选项以及竞价型实例和按需型实例的分配策略。预设情况下，这些设置均未配置。要进行配置，必须选择 Override launch template（覆盖启动模板）。有关创建混合实例组的更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。
5. 在网络下，根据需要选择选项。验证要使用的子网是否位于不同的可用区中。
6. 在 分配策略部分下，选择一个竞价型实例分配策略。要启用或禁用容量再平衡，请选中或清除 容量再平衡下的复选框。仅当您在 实例购买选项部分请求了自动扩缩组要作为竞价型实例启动的百分比时，才会看到此选项。
7. 创建 Auto Scaling 组。
8. （可选）根据需要添加生命周期挂钩。有关更多信息，请参阅 向自动扩缩组添加生命周期挂钩。

为现有自动扩缩组启用或禁用容量再平衡

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。这时将在页面底部打开一个拆分窗格。
3. 在 Details（详细信息）选项卡上，依次选择 Allocation strategies（分配策略）、Edit（编辑）。
4. 在分配策略部分下，可以通过选择或清除容量再平衡下的复选框来启用或禁用容量再平衡。
5. 选择更新。

启用容量再平衡（Amazon CLI）

以下示例说明如何使用启用和禁用容量重新平衡。 Amazon CLI

使用带有以下参数的 update-auto-scaling-groupo create-auto-scaling-groupr 命令：

- `--capacity-rebalance` / `--no-capacity-rebalance`：指示是否启用容量再平衡的布尔值。

在调用该create-auto-scaling-group命令之前，您需要配置为用于 Auto Scaling 组的启动模板的名称。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

> **Note**
>
> 以下过程显示如何使用 JSON 或 YAML 格式的配置文件。如果使用 Amazon CLI 版本 1，则必须指定 JSON 格式的配置文件。如果您使用 Amazon CLI 版本 2，则可以指定格式为 YAML 或 JSON 的配置文件。

JSON

创建和配置新的 Auto Scaling 组

- 使用以下create-auto-scaling-group命令创建新的 Auto Scaling 组并启用容量重新平衡。该命令引用 JSON 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

如果您还没有指定混合实例策略的 CLI 配置文件，请创建一个。

将以下行添加到配置文件中的顶级 JSON 对象。

```
{
    "CapacityRebalance": true
}
```

下面是一个 config.json 示例文件。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacity": 12,
    "MinSize": 12,
    "MaxSize": 15,
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 0,
            "OnDemandPercentageAboveBaseCapacity": 25,
            "SpotAllocationStrategy": "price-capacity-optimized"
```

```
        },
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        }
    },
    "TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

创建和配置新的 Auto Scaling 组

- 使用以下create-auto-scaling-group命令创建新的 Auto Scaling 组并启用容量重新平衡。该命令引用 YAML 文件作为自动扩缩组的唯一参数。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

将以下行添加到 YAML 格式的配置文件。

```
CapacityRebalance: true
```

下面是一个 config.yaml 示例文件。

```
---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: price-capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
TargetGroupARNs:
- arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-
group/943f017f100becff
```

```
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

为现有 Auto Scaling 组启用容量再平衡

- 使用以下update-auto-scaling-group命令启用容量重新平衡。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

验证是否为 Auto Scaling 组启用容量再平衡

- 使用以下describe-auto-scaling-groups命令验证容量重新平衡是否已启用并查看详细信息。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下为响应示例。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "CapacityRebalance": true
        }
    ]
}
```

禁用容量再平衡

使用带有--no-capacity-rebalance选项的update-auto-scaling-group命令可禁用容量重新平衡。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --no-capacity-rebalance
```

相关资源

有关容量再平衡的更多信息，请参阅 C Amazon ompute Blo g 上的 “使用适用于 A EC2 uto Scaling 的全新容量再平衡功能主动管理竞价型实例生命周期”。

有关 EC2 实例再平衡建议的更多信息，请参阅 A mazon EC2 用户指南中的EC2 实例再平衡建议。

要了解有关生命周期挂钩的更多信息，请参阅以下资源。

- 教程：配置调用 Lambda 函数的生命周期钩子（使用 EventBridge）
- 教程：使用数据脚本和实例元数据检索生命周期状态

限制

- Amazon A EC2 uto Scaling 可以替换收到再平衡通知的实例，前提是该实例没有受到缩容保护。但是，横向缩减保护并不能阻止因 Spot 中断而终止。有关更多信息，请参阅 使用实例横向缩减保护以控制实例终止。
- 除中东（阿联酋）地区外，所有提供 Amazon A EC2 uto Scaling 的商业 Amazon Web Services 区域 版均提供容量再平衡支持。

使用容量预留在特定可用区中预留容量

Amazon EC2 按需容量预留可帮助您在特定可用区域预留计算容量。如需开始使用容量预留，您可以在所需的可用区中创建容量预留。然后，您可以在预留容量中启动实例，实时查看其容量使用情况，以及根据需要增加或减少其容量。

容量预留配置为 `open` 或者 `targeted`。如果容量预留处于open状态，具有匹配属性的所有实例和现有实例将会自动在容量预留的容量中运行。如果容量预留处于 `targeted` 状态，只有专门定位到其中的实例才能在预留容量中运行。

容量预留首选项

容量预留首选项通过在使用按需容量之前先确定容量预留中的预留容量的优先级，从而帮助您高效地使用容量预留。您可以从以下容量预留首选项中进行选择：

- 默认-Auto Scaling 使用启动模板中的容量预留首选项或打开的容量预留。
- 无 — Auto Scaling 不会将实例启动到容量预留中。实例将以按需容量运行。

- 仅限容量预留 — Auto Scaling 只会将实例启动到容量预留组或容量预留组中。如果容量不可用，实例将无法启动。
- 首先是容量预留 — Auto Scaling 会将实例启动到容量预留组或容量预留组中。如果容量不可用，实例将以按需容量运行。

如果选择 “仅限容量预留” 或 “先容量预留”，则可以指定容量预留目标。

> **Note**
>
> 您必须选择容量预留首选项。容量预留目标是可选的。

容量预留首选项和启动模板的注意事项

如果您先选择 “仅限容量预留” 或 “容量预留”，请考虑以下事项：

- 如果您选择仅限容量预留或先选择容量预留，Auto Scaling 将使用在 Auto Scaling 组中指定的容量预留目标，而不是启动模板中的容量预留目标。
- 如果您先选择 “仅限容量预留” 或 “容量预留”，但未指定容量预留目标，则 Auto Scaling 将使用启动模板容量预留目标或打开的容量预留。

容量预留目标规范

如果您选择 “仅限容量预留” 或 “先容量预留”，则可以使用以下容量预留目标选项：

- 打开 — Auto Scaling 将在任何已打开的容量预留中启动实例。如果您选择了 “仅限容量预留”，但容量不可用，则实例将无法启动。如果您先选择容量预留但容量不可用，则实例将以按需容量启动。
- 指定容量预留-Auto Scaling 将在指定的容量预留中启动实例。如果您选择了 “仅限容量预留”，但容量不可用，则实例将无法启动。如果您先选择容量预留但容量不可用，则实例将以按需容量启动。
- 指定容量预留资源组-Auto Scaling 会将实例启动到指定容量预留资源组中已打开的容量预留中。如果您选择了 “仅限容量预留”，但容量不可用，则实例将无法启动。如果您先选择容量预留但容量不可用，则实例将以按需容量启动。

在 Auto Scaling 组中使用容量预留

要在 Auto Scaling 组中使用容量预留，您可以向 Auto Scaling 组添加容量预留首选项，也可以在启动模板中指定容量预留目标。无论选择哪种方法，都必须首先创建容量预留或容量预留资源组。

要创建容量预留，请参阅亚马逊 EC2 用户指南中的创建容量预留要创建容量预留组，请参阅亚马逊 EC2 用户指南中的创建容量预留组。

要查看创建使用`targeted`容量预留的 Auto Scaling 组和容量预留组的所有步骤，请参阅在 Auto Scaling 组中使用容量预留以及使用`targeted`容量预留的启动模板。

创建或编辑 Auto Scaling 组并使用容量预留首选项

创建或编辑 Auto Scaling 组时，使用以下方法之一使用容量预留首选项。

Console

在新组上使用容量预留首选项（控制台）

1. 按照中的使用亚马逊 EC2 启动向导创建 Auto Scaling 群组说明完成过程中的每个步骤，直到步骤 3。
2. 在配置组大小和扩展页面上，在其他容量设置、容量预留首选项下，选择容量预留首选项。有关容量预留首选项的更多信息，请参阅容量预留首选项。
3. 继续完成使用亚马逊 EC2 启动向导创建 Auto Scaling 群组中的步骤。

Amazon CLI

要对新组使用容量预留首选项 (Amazon CLI)

向 create-auto-scaling-group 命令添加 `--capacity-reservation-specification` 参数。

1. 指定容量预留首选项。有关更多信息，请参阅 容量预留首选项。
2. 指定容量预留目标。如果您先选择 “仅限容量预留” 或 “容量预留”，但未指定容量预留目标，则 Auto Scaling 将使用启动模板容量预留目标或打开的容量预留。

Console

在现有组上使用容量预留首选项（控制台）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 Auto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了自动扩缩组的 Amazon Web Services 区域 。
3. 选中 Auto Scaling 组旁边的复选框。

这时将在页面底部打开一个拆分窗格。

4. 在 “详细信息” 选项卡上的 “容量预留” 首选项下，选择 “编辑”。
5. 在 “其他容量设置” 的 “容量预留” 首选项下，选择容量预留首选项。有关容量预留首选项的更多信息，请参阅容量预留首选项。
6. 选择更新。

Amazon CLI

要对现有组使用容量预留首选项 (Amazon CLI)

向 update-auto-scaling-group 命令添加 `--capacity-reservation-specification` 参数。

1. 指定容量预留首选项。有关更多信息，请参阅 容量预留首选项。
2. 指定容量预留目标。如果您先选择 “仅限容量预留” 或 “容量预留”，但未指定容量预留目标，则 Auto Scaling 将使用启动模板容量预留目标或打开的容量预留。

在 Auto Scaling 组中使用容量预留以及使用**`targeted`**容量预留的启动模板

本主题演示如何创建自动扩缩组，使其在 `targeted` 容量预留中启动按需型实例。这样可以更好地控制何时使用特定的容量预留。

基本步骤如下：

1. 在具有相同实例类型、平台和实例数量的多个可用区中创建容量预留。
2. 使用 Amazon 资源组对容量预留进行分组。
3. 使用与容量预留相同的可用区，通过定位到资源组的启动模板创建自动扩缩组。

内容

步骤 1：创建容量预留

此过程使用targeted容量预留

> **Note**
>
> 只有在首次创建容量预留时才能创建 targeted 预留。

Console

如需创建您的容量预留

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 选择 容量预留 (容量预留)，然后选择 Create 容量预留 (创建容量预留)。
3. 在创建一个容量预留页面上，注意遵循实例详情章节中的设置。您启动的实例的实例类型、平台和可用区必须与您在此处指定的实例类型、平台和可用区匹配，否则将不会应用容量预留。
   a. 对于实例类型，在预留容量中选择启动的实例类型。
   b. 对平台，选择您实例的操作系统。
   c. 对于可用区，请选择要在其中预留容量的第一个可用区。
   d. 对于总容量，请选择所需的实例数量。计算自动扩缩组所需的实例总数除以计划使用的可用区数量。
4. 在容量预留的详细信息下，对于容量预留的结束方式，请选择下列选项之一：
   - 在特定时间：在指定的日期和时间自动取消容量预留。
   - 手动：容量将预留，直至您明确取消。
5. 对于实例资格，请选择目标：仅限定位到容量预留的实例。
6. （可选）对于标签，请指定要与容量预留关联的任何标签。
7. 选择创建。
8. 记录新创建的容量预留的 ID。您需要用它来设置容量预留组。

对要为自动扩缩组启用的每个可用区重复此过程，仅更改可用区选项的值。

Amazon CLI

如需创建您的容量预留

使用以下create-capacity-reservation命令创建容量预留。替换 `--availability-zone`、`--instance-type`、`--instance-platform` 和 `--instance-count` 的示例值。

```
aws ec2 create-capacity-reservation \
    --availability-zone us-east-1a \
    --instance-type c5.xlarge \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --instance-match-criteria targeted
```

生成的容量预留 ID 示例

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234567890abcdef1",
        "OwnerId": "123456789012",
        "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789012:capacity-
reservation/cr-1234567890abcdef1",
        "InstanceType": "c5.xlarge",
        "InstancePlatform": "Linux/UNIX",
        "AvailabilityZone": "us-east-1a",
        "Tenancy": "default",
        "TotalInstanceCount": 3,
        "AvailableInstanceCount": 3,
        "EbsOptimized": false,
        "EphemeralStorage": false,
        "State": "active",
        "StartDate": "2023-07-26T21:36:14+00:00",
        "EndDateType": "unlimited",
        "InstanceMatchCriteria": "targeted",
        "CreateDate": "2023-07-26T21:36:14+00:00"
    }
}
```

记录新创建的容量预留的 ID。您需要用它来设置容量预留组。

对要为自动扩缩组启用的每个可用区重复此命令，仅更改 `--availability-zone` 选项的值。

步骤 2：创建容量预留组

创建完容量预留后，您可以使用 Res Amazon ource Groups 服务将它们分组在一起。 Amazon Resource Groups 支持几种不同类型的群组，用于不同的用途。Amazon EC2 使用特殊用途组（称为

服务相关资源组）来定位一组容量预留。要与该服务相关资源组进行交互，您可以使用 Amazon CLI 或 SDK，但不能使用控制台。有关服务相关资源组的更多信息，请参阅 Amazon 资源组用户指南中的资源组服务配置。

要使用创建容量预留组 Amazon CLI

使用 create-group 命令创建只能包含容量预留的资源组。在此示例中，资源组名为 *my-cr-group*。

```
aws resource-groups create-group \
    --name my-cr-group \
    --configuration '{"Type":"AWS::EC2::CapacityReservationPool"}'
 '{"Type":"AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-resource-
types", "Values": ["AWS::EC2::CapacityReservation"]}]}'
```

以下为响应示例。

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-east-1:123456789012:group/my-cr-group",
        "Name": "my-cr-group"
    },
    "GroupConfiguration": {
        "Configuration": [
            {
                "Type": "AWS::EC2::CapacityReservationPool"
            },
            {
                "Type": "AWS::ResourceGroups::Generic",
                "Parameters": [
                    {
                        "Name": "allowed-resource-types",
                        "Values": [
                            "AWS::EC2::CapacityReservation"
                        ]
                    }
                ]
            }
        ],
        "Status": "UPDATE_COMPLETE"
    }
}
```

记下新资源组的 ARN。您需要它来为自动扩缩组设置启动模板。

使用 Amazon CLI将您的容量预留与新创建的群组相关联

使用以下 group-resources 命令将容量预留与新创建的容量预留组相关联。对于该--resource-arns选项，请使用容量预留指定容量预留 ARNs。 ARNs 使用相关区域、您的账户 ID 和您之前记下的预订 IDs 来构建。在此示例中，带有 IDs *cr-1234567890abcdef1*和的预留*cr-54321abcdef567890*将分组到名为的组中*my-cr-group*。

```
aws resource-groups group-resources \
    --group my-cr-group \
    --resource-arns \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-1234567890abcdef1 \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-54321abcdef567890
```

以下为响应示例。

```
{
   "Succeeded": [
   "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-1234567890abcdef1",
   "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-54321abcdef567890"
   ],
   "Failed": [],
   "Pending": []
}
```

有关修改或删除资源组的信息，请参阅 Amazon 资源组 API 参考。

步骤 3：创建启动模板

要使用启动模板，请完成步骤 1：创建容量预留和步骤 2：创建容量预留组。然后，创建启动模板

Console

创建启动模板

1. 打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在导航窗格中的实例下，选择启动模板。
3. 选择 Create launch template (创建启动模板)。为启动模板的初始版本输入名称并提供描述。
4. 在 Auto Scaling guidance（Auto Scaling 指导）下，选中复选框。
5. 创建启动模板。选择与计划使用的容量预留相匹配的 AMI 和实例类型，或者选择密钥对、一个或多个安全组，以及用于实例的任何其他 EBS 卷或实例存储卷。

6. 展开 高级设置，并执行以下操作：

   a. 对于容量预留，请选择按组定位。

   b. 对于容量预留 — 按组定位，选择您在上一节中创建的容量预留组，然后选择保存。

7. 选择Create launch template（创建启动模板）。

8. 在确认页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

Amazon CLI

创建启动模板

使用以下create-launch-template命令创建启动模板，该模板指定容量预留以特定的资源组为目标。替换 `--launch-template-name` 的样本值。将 *c5.xlarge* 替换为您在容量预留中使用的实例类型，并将 *ami-0123456789EXAMPLE* 替换为您要使用的 AMI 的 ID。将 *arn:aws:resource-groups:region:account-id:group/my-cr-group* 替换为上一节开头创建的资源组的 ARN。

```
aws ec2 create-launch-template \
    --launch-template-name my-launch-template \
    --launch-template-data \
        '{"InstanceType": "c5.xlarge",
          "ImageId": "ami-0123456789EXAMPLE",
          "CapacityReservationSpecification":
            {"CapacityReservationTarget":
                { "CapacityReservationResourceGroupArn": "arn:aws:resource-
groups:region:account-id:group/my-cr-group" }
            }
        }'
```

以下为响应示例。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-0dd77bd41dEXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "CreateTime": "2023-07-26T21:42:48+00:00",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
```

```
}
```

步骤 4：创建自动扩缩组

Console

像往常一样创建自动扩缩组，但是在选择 VPC 子网时，请从每个可用区中选择一个与您创建的 `targeted` 容量预留相匹配的子网。然后，当您的自动扩缩组在其中一个可用区启动按需型实例时，该实例将在该可用区的预留容量下运行。如果资源组在达到您的所需容量之前用完了容量预留，我们会将超出预留容量的任何容量作为常规按需容量启动。

创建一个简单的自动扩缩组

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏上，选择与创建启动模板时相同的 Amazon Web Services 区域 模板。
3. 选择 Create an Auto Scaling group (创建 Auto Scaling 组)。
4. 在选择启动模板或配置页面上，对于 Auto Scaling 组名称，输入 Auto Scaling 组的名称。
5. 对于启动模板，请选择现有启动模板。
6. 对于 Launch template version（启动模板版本），选择 Auto Scaling 组在扩展时使用启动模板的默认版本、最新版本还是特定版本。
7. 在选择实例启动选项页面上，跳过实例类型要求部分，使用启动模板中指定的 EC2 实例类型。
8. 在 Network (网络) 下，对于 VPC，选择相应的 VPC。必须在启动模板中指定的安全组所在的 VPC 中创建 Auto Scaling 组。如果您没有在启动模板中指定安全组，您可以选择与容量预留相同的可用区中拥有子网的任何 VPC。
9. 对于可用区和子网，根据您的容量预留所在的可用区，从要包含的每个可用区中选择子网。
10. 选择 Next（下一步）两次。
11. 在配置组大小和扩缩策略页面上，对于所需容量，输入要启动的初始实例数。将此数字更改为超出最小容量或最大容量限制的值时，必须更新最小容量或最大容量的值。有关更多信息，请参阅 为自动扩缩组设置扩缩限制。
12. 选择 Skip to review (跳转以查看)。
13. 在 Review（查看）页面上，选择 Create Auto Scaling group（创建 Auto Scaling 组）。

Amazon CLI

创建一个简单的自动扩缩组

使用以下create-auto-scaling-group命令并指定启动模板的名称和版本作为该--launch-template选项的值。替换 --auto-scaling-group-name、--min-size、--max-size 和 --vpc-zone-identifier 的示例值。

对于 --availability-zones 选项，请指定您为其创建容量预留的可用区。例如，如果您的容量预留指定了 us-east-1a 和 us-east-1b 可用区，则必须在相同的区域中创建自动扩缩组。然后，当您的自动扩缩组在其中一个可用区启动按需型实例时，该实例将在该可用区的预留容量下运行。如果资源组在达到您的所需容量之前用完了容量预留，我们会将超出预留容量的任何容量作为常规按需容量启动。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-launch-template,Version='1' \
    --min-size 6 \
    --max-size 6 \
    --vpc-zone-identifier "subnet-5f46ec3b,subnet-0ecac448" \
    --availability-zones us-east-1a us-east-1b
```

相关资源

有关示例实现，请参阅以下 Amazon 示例 GitHub 存储库中的 Amazon CloudFormation 模板：https://github.com/aws-samples/aws-auto-scaling-backed-by-on-demand-capacity-reservations/。

在您了解容量预留时，以下相关主题可能对您有所帮助。

- 按需容量预留
  - 在 Amazon EC2 用户指南中@@ 创建容量预留
  - Amazon EC2 用户指南中的@@ 按需容量预留
  - 在 Amazon 云运营@@ 和迁移博客上定位一组 Amazon EC2 按需容量预留
- 容量块（具有定义的持续时间的容量预留）
  - Amazon EC2 用户指南中的 ML 容量块
  - 使用 Capacity Blocks 适用于机器学习工作负载

使用命令行创建 Auto Scaling 组 Amazon CloudShell

在 s u pported 中 Amazon Web Services 区域，您可以使用直接从启动 Amazon CloudShell 的基于浏览器、预先验证的 shell 运行 Amazon CLI 命令。 Amazon Web Services Management Console您可以使用首选外壳（Bash PowerShell、或 Z shell）对服务运行 Amazon CLI 命令。

您可以使用以下两种 Amazon Web Services Management Console 方法之一 Amazon CloudShell 从启动：

- 选择控制台导航栏上的 Amazon CloudShell 图标。它位于搜索框的右侧。
- 使用控制台导航栏上的搜索框进行搜索，CloudShell然后选择CloudShell选项。

首次在新的浏览器窗口中 Amazon CloudShell 启动时，将显示一个欢迎面板并列出主要功能。关闭此面板后，系统会在 shell 配置和转发控制台凭证的同时提供状态更新。当系统显示命令提示符时，表示 shell 已经准备就绪，可以进行交互。

有关此服务的更多信息，请参阅 Amazon CloudShell 用户指南。

使用 Amazon CloudFormation创建 Auto Scaling 组

Amazon A EC2 uto Scaling 与 Amazon CloudFormation一项服务集成，可帮助您对 Amazon 资源进行建模和设置，从而减少创建和管理资源和基础设施所花费的时间。您可以创建一个描述所需的所有 Amazon 资源（例如 Auto Scaling 组）的模板，然后 Amazon CloudFormation 为您预置和配置这些资源。

使用时 Amazon CloudFormation，您可以重复使用您的模板来一致且重复地设置 Amazon A EC2 uto Scaling 资源。只需描述一次您的资源，然后在多个 Amazon Web Services 账户 区域中一遍又一遍地配置相同的资源。

Amazon A EC2 uto Scaling 和 Amazon CloudFormation 模板

要为 Amazon A EC2 uto Scaling 和相关服务预配置和配置资源，您必须了解Amazon CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述了您要在 Amazon CloudFormation 堆栈中配置的资源。如果你不熟悉 JSON 或 YAML，可以使用 D Amazon CloudFormation esigner 来帮助你开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅什么是 Amazon CloudFormation 设计器？ 在《Amazon CloudFormation 用户指南》中。

要开始为 Amazon A EC2 uto Scaling 创建自己的堆栈模板，请完成以下任务：

- 使用创建启动模板。
- 使用群组创建 Auto Scaling AWS::AutoScaling::AutoScaling群。

有关演示如何在应用程序负载均衡器后部署自动扩缩组的演练，请参阅《Amazon CloudFormation 用户指南》中的演练：创建经扩展和负载均衡的应用程序。

您可以在《Amazon CloudFormation 用户指南》的以下章节中找到创建自动扩缩组和相关资源的模板代码片段的更多有用示例：

- Amazon A EC2 uto Scaling 资源类型参考
- 使用配置 Amazon A EC2 uto Scaling 资源 Amazon CloudFormation

了解更多关于 Amazon CloudFormation

要了解更多信息 Amazon CloudFormation，请参阅以下资源：

- Amazon CloudFormation
- Amazon CloudFormation 用户指南
- Amazon CloudFormation API 引用
- Amazon CloudFormation 命令行界面用户指南

通过获取实例类型建议 Amazon Compute Optimizer

Amazon 提供 Amazon EC2 实例类型建议，通过使用由支持的功能帮助您提高性能、节省资金或两者兼而有之 Amazon Compute Optimizer。您可以根据这些建议来决定是否移动到自动扩缩组中的新实例类型。

为了生成建议，Compute Optimizer 会分析现有实例规范和最近指标历史记录。然后，使用编译后的数据来推荐哪些 Amazon EC2 实例类型最适合处理现有性能工作负载。建议随每小时实例定价一起返回。

> **Note**
>
> 要从 Compute Optimizer 中获取建议，您必须首先选择加入 Compute Optimizer。有关更多信息，请参阅 Amazon Compute Optimizer 用户指南中的 Amazon Compute Optimizer入门。

内容

限制

Compute Optimizer 为 Auto Scaling 组中的实例生成建议，这些实例配置为启动和运行 M、C、R、T 和 X 实例类型。但是，它不会针对由 G Amazon raviton2 处理器提供支持的-g 实例类型（例如 c6g）和具有更高网络带宽性能的-n 实例类型（例如 m5n）生成建议。

Auto Scaling 组还必须配置为运行单个实例类型（即无混合实例类型），不得附加扩展策略，并且对于所需容量、最小容量和最大容量（即具有固定数量实例的 Auto Scaling 组）具有相同的值。Compute Optimizer 为 Auto Scaling 组中满足所有这些配置要求的实例生成建议。

调查发现

Compute Optimizer 对 Auto Scaling 组的结果进行分类，如下所示：

- 未优化 – 当 Compute Optimizer 确定可为您的工作负载提供更好性能的建议时，Auto Scaling 组被视为未优化。
- 优化 – 当 Compute Optimizer 确定已根据所选实例类型正确预配置 Auto Scaling 组以运行工作负载时，该组将被视为已优化。对于优化的资源，Compute Optimizer 有时可能会建议新一代实例类型。
- 无 – 没有关于该 Auto Scaling 组的建议。如果您选择使用 Computer Optimizer 的时间少于 12 小时、Auto Scaling 组的运行时间少于 30 小时，或者 Computer Optimizer 不支持 Auto Scaling 组或实例类型，则可能会发生这种情况。想要了解更多信息，请参阅 限制 部分。

查看建议

选择使用 Compute Optimizer 后，您可以查看它为 Auto Scaling 组生成的结果和建议。如果您是最近选择使用的，可能在长达 12 小时内不会提供建议。

查看为 Auto Scaling 组生成的建议

1. 打开 Compute Optimizer 控制台，网址为。https://console.aws.amazon.com/compute-optimizer/

此时将打开“控制面板”页面。

2. 选择 View recommendations for all Auto Scaling groups（查看所有 Auto Scaling 组的建议）。
3. 选择您的 Auto Scaling 组。
4. 选择 View detail（查看详细信息）。

   视图根据默认表设置在预配置视图中更改为最多显示三种不同的实例建议。它还提供了 Auto Scaling 组的最新 CloudWatch 指标数据（平均 CPU 利用率、平均网络输入和平均网络输出）。

确定是否要使用其中某个建议。决定是否要进行优化以便提高性能和/或节省资金。

要更改 Auto Scaling 组中的实例类型，请更新启动模板或更新 Auto Scaling 组以使用新的启动配置。现有实例将继续使用以前的配置。要更新现有实例，请终止这些实例，以便 Auto Scaling 组替换这些实例，或者根据您的终止策略启用自动扩展，以逐步使用较新实例替换较旧实例。

> **Note**
>
> 借助最长实例生命周期和实例刷新功能，您还可以替换 Auto Scaling 组中的现有实例，以启动使用新启动模板或启动配置的新实例。有关更多信息，请参阅 基于最大实例生命周期替换 Auto Scaling 实例 和 使用实例刷新更新自动扩缩组中的实例。

评估建议时的注意事项

移至新的实例类型时，请考虑以下事项：

- 这些建议不会预测您的使用情况。建议基于您在最近 14 天时间段内的历史使用情况。请务必选择一种预计能够满足您的未来使用需求的实例类型。
- 关注图表指标以确定实际使用量是否低于实例容量。您还可以在中查看指标数据（平均值、峰值、百分位数），CloudWatch 以进一步评估您的 EC2 实例建议。例如，观察当天 CPU 百分比指标如何变化，以及是否有需要满足的峰值。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的查看可用指标。
- Compute Optimizer 可能会为可突增性能实例（即 T3、T3a 和 T2 实例）提供建议。如果您定期突破基准，请确保您可以根据新实例类型的 v CPUs 继续进行突破。有关更多信息，请参阅 Amazon EC2 用户指南中的突发性能实例的 CPU 积分和基准性能。
- 如果您购买了 Reserved Instance，您的按需型实例可能会作为预留实例进行计费。在更改当前实例类型之前，请首先评估对Reserved Instance使用率和覆盖率的影响。

- 尽可能考虑转换为较新一代实例。
- 在迁移到其他实例系列时，请确保当前实例类型和新实例类型在虚拟化、架构或网络类型等方面兼容。有关更多信息，请参阅 Amazon EC2 用户指南中的调整实例大小的兼容性。
- 最后，请考虑为每个建议提供的性能风险评级。性能风险指示您为了验证建议的实例类型是否满足工作负载的性能要求而可能需要执行的工作量。我们还建议在进行任何更改前后进行严格的负载和性能测试。

其他资源

除了本页面上的主题以外，还可以参阅以下资源：

- Amazon EC2 实例类型
- Amazon Compute Optimizer 用户指南

使用 Elastic Load Balancing 在 Auto Scaling 组中分配传入的应用程序流量

Elastic Load Balancing 会自动将您的传入应用程序流量分配到您正在运行的所有 EC2实例。Elastic Load Balancing 通过最佳路由流量管理传入请求，以便所有实例都不会负载过多。要将 Elastic Load Balancing 与您的 Auto Scaling 组配合使用，请将负载均衡器附加到您的 Auto Scaling 组。这将使用负载均衡器注册该组，负载均衡器将作为到您的 Auto Scaling 组的所有传入 Web 流量的单一接触点。

当您将 Elastic Load Balancing 与 Auto Scaling 组一起使用时，无需向负载均衡器注册单个 EC2实例。您的 Auto Scaling 组所启动的实例会自动注册到负载均衡器。同样，您的 Auto Scaling 组所终止的实例会自动从负载均衡器取消注册。

将负载均衡器附加到您的 Auto Scaling 组后，可以将您的 Auto Scaling 组配置为使用 Elastic Load Balancing 指标（如每个目标的 Application Load Balancer 请求计数）来随着需求波动而扩展该组中的实例数量。

或者，您可以将 Elastic Load Balancing 运行状况检查添加到您的 Auto Scaling 组中，这样 Amazon A EC2 uto Scaling 就可以根据这些额外的运行状况检查识别和替换运行状况不佳的实例。否则，您可以创建一个 CloudWatch 警报，在目标组的健康主机数低于允许值时通知您。

内容

- Elastic Load Balancing 类型
- 准备附加 Elastic Load Balancing 负载均衡器

- 将 Elastic Load Balancing 负载均衡器附加到自动扩缩组
- 从控制台配置应用程序负载均衡器或网络负载均衡器
- 验证负载均衡器的附加状态
- 添加可用区
- 删除可用区
- 从自动扩缩组分离目标组或经典负载均衡器
- 使用 Elastic Load Balancing 的示例 Amazon CLI

Elastic Load Balancing 类型

Elastic Load Balancing 提供四种类型的负载均衡器，可与您的 Auto Scaling 组配合使用：Application Load Balancer、Network Load Balancer、网关负载均衡器和经典负载均衡器。

负载均衡器类型的配置方式具有一个关键区别。通过 Application Load Balancer、Network Load Balancer 和网关负载均衡器，可以使用目标组将实例注册为目标并将流量路由到目标组。通过经典负载均衡器，可以使用负载均衡器直接注册实例。

Application Load Balancer

路由和负载均衡在应用程序层 (HTTP/HTTPS) 进行，并支持基于路径的路由。Application Load Balancer 可以将请求路由到您的虚拟私有云 (VPC) 中一个或多个注册目标（例如 EC2 实例）上的端口。

网络负载均衡器

路由和负载均衡在传输层（TCP/UDP 第 4 层）进行，依据是从第 4 层中提取的地址信息。Network Load Balancer 可以处理突发流量，保留客户端的源 IP，并在负载均衡器的使用寿命内使用固定 IP。

网关负载均衡器

将流量分配到设备实例队列。为第三方虚拟设备（如防火墙、入侵检测和防御系统以及其他设备）提供可扩展性、可用性和简单性。网关负载均衡器与支持 GENEVE 协议的虚拟设备配合使用。需要额外的技术集成，因此请务必在选择网关负载均衡器之前参考用户指南。

Classic 负载均衡器

在传输层进行路由和负载平衡 (TCP/SSL), or at the application layer (HTTP/HTTPS)。

要更深入地了解可用的不同类型的负载均衡器，请参阅以下资源：

- 什么是 Elastic Load Balancing？
- 什么是 Application Load Balancer？
- 什么是 Network Load Balancer？
- 什么是网关负载均衡器？
- 什么是经典负载均衡器？

准备附加 Elastic Load Balancing 负载均衡器

在将 Elastic Load Balancing 负载均衡器附加到自动扩缩组之前，您必须完成以下先决条件：

- 您必须已创建用于将流量路由到自动扩缩组的负载均衡器和目标组。

  有两种方法来创建负载均衡器和目标组：

  - 使用 Elastic Load Balancing：按照 Elastic Load Balancing 文档中的程序，在创建自动扩缩组之前创建并配置负载均衡器和目标组。跳过注册您的 Amazon EC2 实例的步骤。当您将目标组附加到 A EC2 uto Scaling 组时，Amazon Auto Scaling 会自动负责注册（和取消注册）实例。有关更多信息，请参阅 Elastic Load Balancing 用户指南中的 Elastic Load Balancing 入门。
  - 使用 Amazon A EC2 uto Scaling — 使用 Amazon A EC2 uto Scaling 控制台中的基本配置创建、配置和连接负载均衡器和目标组。有关更多信息，请参阅 从控制台配置应用程序负载均衡器或网络负载均衡器。
- 在创建负载均衡器之前，请先了解所需的负载均衡器类型。有关更多信息，请参阅 Elastic Load Balancing 类型。
- 负载均衡器及其目标组必须与您的 Auto Scaling 组位于相同 Amazon Web Services 账户的 VPC 和区域中。
- 目标组必须指定的 `instance` 目标类型。使用 Auto Scaling 组时，无法指定 `ip` 的目标类型。
- 如果自动扩缩组的启动模板未包含允许来自负载均衡器的必要入站流量的正确安全组，则必须更新启动模板。推荐规则取决于负载均衡器的类型和负载均衡器使用的后端类型。例如，要将流量路由到 Web 服务器，请允许从负载均衡器在端口 80 上进行入站 HTTP 访问。修改启动模板时，现有实例不会使用新设置进行更新。要更新现有实例，您可以启动实例刷新以替换实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。
- 启动模板中的安全组还必须允许从 Elastic Load Balancing 负载均衡器通过正确的端口访问以执行运行状况检查。
- 在网关负载均衡器后部署虚拟设备时，启动模板中的亚马逊机器映像（AMI）必须指定某个支持 GENEVE 协议的 AMI 的 ID，以允许自动扩缩组与网关负载均衡器交换流量。此外，启动模板中的安全组必须允许 UDP 流量通过端口 6081。

> **Tip**
>
> 如果您拥有需要一段时间才能完成的引导启动脚本，则可以选择向您的 Auto Scaling 组添加启动生命周期钩子，以便在引导启动脚本成功完成并且实例上的应用程序准备好接受流量之前，推迟将实例注册到负载均衡器后。最初在 Amazon A EC2 uto Scaling 控制台中创建 Auto Scaling 组时，您无法添加生命周期挂钩。不过，您可以在创建组后再添加生命周期挂钩。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。

配置目标的运行状况检查

您可以为向 Elastic Load Balancing 负载均衡器注册的目标配置运行状况检查，以确保其能够正确处理流量。具体步骤因您使用的负载均衡器类型而有所不同。有关更多信息，请参阅以下资源：

- 应用程序负载均衡器：请参阅《User Guide for Application Load Balancers》中的 Health checks for your target groups。
- 网络负载均衡器：请参阅《User Guide for Network Load Balancers》中的 Health checks for your target groups。
- 网关负载均衡器：请参阅《User Guide for Gateway Load Balancers》中的 Health checks for your target groups。
- 经典负载均衡器：《User Guide for Classic Load Balancers》中的 Configure health checks for your Classic Load Balancer。

默认情况下，Amazon A EC2 uto Scaling 不会认为实例运行状况不佳，如果该实例未通过 Elastic Load Balancing 运行状况检查，则将其替换。Auto Scaling 组的默认运行 EC2状况检查仅是运行状况检查。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

要让 Amazon A EC2 uto Scaling 替换由 Elastic Load Balancing 报告运行状况不佳的实例，您可以将 Auto Scaling 组配置为使用 Elastic Load Balancing 运行状况检查。这样，如果实例未通过运行 EC2 状况检查或 Elastic Load Balancing 运行状况检查，Amazon A EC2 uto Scaling 就会认为该实例运行状况不佳。如果您将多个负载均衡器目标组或经典负载均衡器附加到该组，则只有在所有负载均衡器目标组或经典负载均衡器均报告某实例运行状况良好的情况下，它才会认为该实例运行状况良好。如果其中任何一个负载均衡器目标组或经典负载均衡器将实例报告为运行状况不佳，则 Auto Scaling 组将替换该实例，即使其他负载均衡器目标组或经典负载均衡器将实例报告为运行状况良好也是如此。

有关如何为自动扩缩组启用这些运行状况检查的信息，请参阅将 Elastic Load Balancing 负载均衡器附加到自动扩缩组。

> **Note**
>
> 为确保这些运行状况检查尽快开始，确保组的运行状况检查宽限期未设置得过高，但应设置得足够高，以便 Elastic Load Balancing 运行状况检查确定目标是否可用于处理请求。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。

将 Elastic Load Balancing 负载均衡器附加到自动扩缩组

本主题描述了如何将 Elastic Load Balancing 负载均衡器附加到自动扩缩组。它还描述了如何启用 Elastic Load Balancing 运行状况检查，让 Amazon A EC2 uto Scaling 替换 Elastic Load Balancing 报告为运行状况不佳的实例。

默认情况下，Amazon A EC2 uto Scaling 仅根据亚马逊运行状况检查替换运行 EC2 状况不佳或无法访问的实例。如果您启用 Elastic Load Balancing 运行状况检查，则如果您连接到 A EC2 uto Scaling 组的任何弹性负载平衡负载均衡器报告运行状况不佳，Amazon Auto Scaling 可以替换正在运行的实例。

有关将应用程序负载均衡器附加到自动扩缩组的教程，请参阅教程：设置具有扩展和负载均衡功能的应用程序。

> **Important**
>
> 在继续之前，请完成上一节中的所有先决条件。

内容

附加目标组或经典负载均衡器

创建或更新自动扩缩组时，可以附加一个或多个目标组或经典负载均衡器。当您附加应用程序负载均衡器、网络负载均衡器或网关负载均衡器时，您将附加目标组而不是负载均衡器本身。

请按照本部分中的步骤，使用控制台来执行以下操作：

- 将目标组或经典负载均衡器附加到自动扩缩组
- 开启 Elastic Load Balancing 的运行状况检查

在创建新的 Auto Scaling 组时附加现有负载均衡器

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了负载均衡器的 Amazon Web Services 区域 。
3. 选择 Create Auto Scaling group（创建 Auto Scaling 组）。
4. 在步骤 1 和 2中，选择所需的选项，然后继续执行步骤 3：配置高级选项。
5. 对于负载均衡，选择附上现有负载均衡器。
6. 在附加到现有负载均衡器，请执行以下操作之一：

   a. 对于 Application Load Balancer、Network Load Balancer 和网关负载均衡器：

      选择从负载均衡器目标组中选择，然后在现有负载均衡器目标组字段中选择目标组。

   b. 对于经典负载均衡器：

      选择从经典负载均衡器中选择，然后在 经典负载均衡器 字段中选择您的负载均衡器。
7. （可选）对于运行状况检查、其他运行状况检查类型，请选择启用 Elastic Load Balancing 运行状况检查。
8. （可选）对于运行状况检查宽限期，输入时间长短（以秒为单位）。这是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
9. 继续创建 Auto Scaling 组。创建 Auto Scaling 组后，您的实例将自动注册到负载均衡器。

将现有的负载均衡器附加到您的自动扩缩组（在其创建后）

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 “集成” 选项卡上，选择 “负载平衡”、“编辑”。
4. 在 Load balancing (负载均衡) 下，执行下列操作之一：

   a. 对于应用程序、网络或网关负载均衡器目标组，选中其复选框，然后选择一个目标组。

   b. 对于 经典负载均衡器，选中其复选框，然后选择您的负载均衡器。

5. 选择更新。

附加完负载均衡器后，您可以选择开启使用该负载均衡器的运行状况检查。

开启 Elastic Load Balancing 运行状况检查

1. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
2. 对于运行状况检查、其他运行状况检查类型，请选择启用 Elastic Load Balancing 运行状况检查。
3. 对于运行状况检查宽限期，输入时间长短（以秒为单位）。这是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
4. 选择更新。

> **Note**
>
> 在附加负载均衡器时，您可以使用 Amazon CLI来监控负载均衡器的状态。当 Amazon A EC2 uto Scaling 成功注册了实例并且至少有一个注册的实例通过了运行状况检查后，您将收到的状态为`InService`。有关更多信息，请参阅 验证负载均衡器的附加状态。

分离目标组或经典负载均衡器

如果不再需要负载均衡器，请使用以下步骤将其与 Auto Scaling 组分离。

将负载均衡器与组分离

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details (详细信息) 选项卡上，选择 Load balancing (负载均衡)、Edit (编辑)。
4. 在 Load balancing (负载均衡) 下，执行下列操作之一：

   a. 对于应用程序、网络或网关负载均衡器目标组，请选择目标组旁边的删除 (X) 图标。

   b. 对于 经典负载均衡器，请选择负载均衡器旁边的删除 (X) 图标。
5. 选择更新。

完成分离目标组后，您可以关闭 Elastic Load Balancing 运行状况检查。

关闭 Elastic Load Balancing 运行状况检查

1. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
2. 对于运行状况检查、其他运行状况检查类型，取消选择开启弹性负载均衡运行状况检查。
3. 选择更新。

从控制台配置应用程序负载均衡器或网络负载均衡器

在创建您的 Auto Scaling 组时，使用以下过程创建和附加 Application Load Balancer 或 Network Load Balancer。

在创建新的 Auto Scaling 组时创建和附加新的负载均衡器

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选择 Create Auto Scaling group（创建 Auto Scaling 组）。
3. 在步骤 1 和 2 中，选择所需的选项，然后继续执行步骤 3：配置高级选项。
4. 对于负载均衡，选择附加到新的负载均衡器。
   a. 在 Attach to a new load balancer（附加到新的负载均衡器）下，对于 Load balancer type（负载均衡器类型），选择是创建 Application Load Balancer 还是 Network Load Balancer。
   b. 对于负载均衡器名称，输入负载均衡器的名称，或者保留默认名称。
   c. 对于负载均衡器模式，选择是创建面向互联网的公共负载均衡器，还是保留内部负载均衡器的默认负载均衡器。
   d. 对于可用区和子网，请为您选择启动 EC2 实例的每个可用区选择公有子网。（这些都是从步骤 2 预填充的。）
   e. 对于侦听器和路由，请更新侦听器的端口号（如有必要），然后在默认路由中，选择创建目标组。或者，您可以从下拉列表中选择现有目标组。
   f. 如果您选择在最后一步创建目标组，对于新目标组名称，请输入目标组的名称，或者保留默认名称。
   g. 要为负载均衡器添加标签，请选择 Add tag（添加标签），然后提供每个标签的标签键和值。
5. （可选）对于运行状况检查、其他运行状况检查类型，请选择启用 Elastic Load Balancing 运行状况检查。

6. （可选）对于运行状况检查宽限期，输入时间长短（以秒为单位）。这是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
7. 继续创建 Auto Scaling 组。创建 Auto Scaling 组后，您的实例将自动注册到负载均衡器。

> **Note**
>
> 创建 Auto Scaling 组后，您可以使用 Elastic Load Balancing 控制台创建其他侦听器。如果您需要使用安全协议（如 HTTPS，或 UDP 侦听器）创建侦听器，这将非常有用。只要使用不同的端口，您就可以向现有负载均衡器添加更多侦听器。

验证负载均衡器的附加状态

当您附加了负载均衡器后，它进入 `Adding` 状态，同时注册组中的实例。注册了组中的所有实例后，它进入 `Added` 状态。在至少一个注册实例通过运行状况检查后，它进入 `InService` 状态。当负载均衡器处于`InService`状态时，Amazon A EC2 uto Scaling 可以终止并替换任何报告为运行状况不佳的实例。如果注册的实例均未通过运行状况检查（例如，由于未正确配置运行状况检查），负载均衡器不会进入 `InService` 状态。Amazon A EC2 uto Scaling 不会终止和替换实例。

当分离负载均衡器时，它进入 `Removing` 状态，同时取消注册组中的实例。实例在取消注册后仍保持运行。预设情况下，Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer 启用 Connection Draining（注销延迟）功能。如果启用了 Connection Draining，则 Elastic Load Balancing 将等待动态请求完成或最大超时到期（以先到者为准），然后再取消注册实例。

您可以使用 Amazon Command Line Interface (Amazon CLI) 或，验证附件状态 Amazon SDKs。您无法通过控制台来验证附加状态。

使用 Amazon CLI 来验证附件状态

以下describe-traffic-sources命令返回指定 Auto Scaling 组的所有流量源的连接状态。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

该示例会返回附加到自动扩缩组的 Elastic Load Balancing 目标组的 ARN，以及 `State` 元素中目标组的附加状态。

```
{
    "TrafficSources": [
```

```
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456",
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

添加可用区

为利用地理冗余的安全性和可靠性，请使自动扩缩组跨您工作所在区域中的多个可用区，然后附加负载均衡器以跨这些可用区分配传入流量。

当一个可用区运行不佳或不可用时，Amazon A EC2 uto Scaling 会在未受影响的可用区中启动新实例。当运行状况不佳的可用区恢复到正常状态时，Amazon A EC2 uto Scaling 会自动在您的 Auto Scaling 组的所有可用区域中均匀地重新分配应用程序实例。Amazon A EC2 uto Scaling 通过尝试在可用区中启动实例最少的新实例来实现这一点。但是，如果尝试失败，Amazon A EC2 uto Scaling 会尝试在其他可用区启动，直到成功为止。

Elastic Load Balancing 会为您为负载均衡器启用的每个可用区创建一个负载均衡器节点。如果您为负载均衡器启用了跨区域负载均衡，则每个负载均衡器节点会在所有启用的可用区中的已注册目标之间平均分配流量。如果禁用了跨区域负载均衡，则每个负载均衡器节点会仅在其可用区中的已注册实例之间平均分配请求。

创建 Auto Scaling 组时，必须指定至少一个可用区。之后，您可以通过将可用区添加到您的 Auto Scaling 组中，然后为您的负载均衡器启用该可用区（如果负载均衡器支持），来扩展应用程序的可用性。

限制

要更新为负载均衡器启用的可用区，您需要了解以下限制：

- 如果为您的负载均衡器启用可用区，请指定该可用区中的一个子网。请注意，您最多可为负载均衡器启用每个可用区最多一个子网。
- 对于面向互联网的负载均衡器，您为负载均衡器指定的子网必须至少具有八个可用 IP 地址。
- 对于 Application Load Balancer，您必须启用至少两个可用区。
- 对于 Network Load Balancer，您无法禁用已启用的可用区，但可以启用其他可用区。
- 对于网关负载均衡器，您无法禁用已启用的可用区，但可以启用其他可用区。

使用以下过程将 Auto Scaling 组和负载均衡器扩展到其他可用区中的子网。

添加可用区

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details (详细信息) 选项卡上，选择 Network (网络)、Edit (编辑)。
4. 在子网中，选择与要添加到 Auto Scaling 组的可用区相对应的子网。
5. 选择更新。
6. 要更新负载均衡器的可用区，使其与您的 Auto Scaling 组共享相同的可用区，请完成以下步骤：

   a. 在导航窗格上的负载均衡下，选择负载均衡器。

   b. 选择负载均衡器。

   c. 请执行以下操作之一：

      - 对于 Application Load Balancer 和 Network Load Balancer：

        1. 在描述选项卡上，为可用区选择编辑子网。
        2. 在编辑子网页面上，为可用区选中要添加的可用区的复选框。如果该区域只有一个子网，则将选择此子网。如果该区域有多个子网，请选择其中一个子网。

      - 对于 VPC 中的经典负载均衡器：

        1. 在 Instances 选项卡中，选择 Edit Availability Zones。
        2. 在添加和删除子网页面上，对于可用子网，使用其添加 (+) 图标选择该子网。该子网将移到 Selected subnets 下。

   d. 选择保存。

相关资源

当您更改可用区域时，Amazon A EC2 uto Scaling 会重新平衡您的群组。这意味着要替换和重新分配某些实例。有关更多信息，请参阅 示例：在可用区之间分配实例。

如果您在可用区中注册了未启用负载均衡器的目标，则负载均衡器不会将流量路由到这些目标。有关更多信息，请参阅 弹性负载均衡 用户指南中的 Elastic Load Balancing 工作原理

删除可用区

要从您的 Auto Scaling 组和负载均衡器中删除可用区，请使用以下步骤。

删除可用区

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details (详细信息) 选项卡上，选择 Network (网络)、Edit (编辑)。
4. 在子网中，选择与您要从 Auto Scaling 组删除的可用区相对应的子网的删除 (X) 图标。如果该区域有多个子网，请为每个子网选择删除 (X) 图标。
5. 选择更新。
6. 要更新负载均衡器的可用区，使其与您的 Auto Scaling 组共享相同的可用区，请完成以下步骤：

   a. 在导航窗格上的负载均衡下，选择负载均衡器。

   b. 选择负载均衡器。

   c. 请执行以下操作之一：

      - 对于应用程序负载均衡器：

        1. 在描述选项卡上，为可用区选择编辑子网。
        2. 在编辑子网页面上，为可用区清除要删除该可用区子网的复选框。

      - 对于 VPC 中的经典负载均衡器：

        1. 在 Instances 选项卡中，选择 Edit Availability Zones。
        2. 在添加和删除子网页面上，对于可用子网，使用其删除 (-) 图标删除该子网。子网将移至可用子网下。

   d. 选择保存。

从自动扩缩组分离目标组或经典负载均衡器

如果不再需要负载均衡器，请使用以下步骤将其与 Auto Scaling 组分离。

将负载均衡器与组分离

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在 Auto Scaling groups（Auto Scaling 组）页面底部打开一个拆分窗格。
3. 在 Details (详细信息) 选项卡上，选择 Load balancing (负载均衡)、Edit (编辑)。
4. 在 Load balancing (负载均衡) 下，执行下列操作之一：

   a. 对于应用程序、网络或网关负载均衡器目标组，请选择目标组旁边的删除 (X) 图标。

   b. 对于 经典负载均衡器，请选择负载均衡器旁边的删除 (X) 图标。
5. 选择更新。

完成分离目标组后，您可以关闭 Elastic Load Balancing 运行状况检查。

关闭 Elastic Load Balancing 运行状况检查

1. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
2. 对于运行状况检查、其他运行状况检查类型，取消选择开启弹性负载均衡运行状况检查。
3. 选择更新。

使用 Elastic Load Balancing 的示例 Amazon CLI

使用 Amazon Command Line Interface (Amazon CLI) 来连接、分离和描述负载均衡器和目标组，添加和移除 Elastic Load Balancing 运行状况检查，以及更改启用了哪些可用区。

本主题显示了执行 Amazon A EC2 uto Scaling 常见任务的 Amazon CLI 命令示例。

> **Important**
>
> 有关更多命令示例，请参阅Amazon CLI 命令参考中的`aws elbv2` 和 `aws elb`。

内容

- 附加目标组或经典负载均衡器
- 描述您的目标组或经典负载均衡器

- 添加 Elastic Load Balancing 运行状况检查
- 更改您的可用区
- 分离目标组或经典负载均衡器
- 移除 Elastic Load Balancing 运行状况检查
- 旧版命令

附加目标组或经典负载均衡器

使用以下create-auto-scaling-group命令创建 Auto Scaling 组，并通过指定目标组的亚马逊资源名称 (ARN) 来同时附加目标组。目标组可以关联至应用程序负载均衡器、网络负载均衡器或网关负载均衡器。

替换 `--auto-scaling-group-name`、`--vpc-zone-identifier`、`--min-size` 和 `--max-size` 的示例值。对于 `--launch-template` 选项，请将 *`my-launch-template`* 和 *`1`* 替换为您的自动扩缩组的启动模板的名称和版本。对于 `--traffic-sources` 选项，将示例 ARN 替换为应用程序负载均衡器、网络负载均衡器或网关负载均衡器的目标组的 ARN。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1"
```

在 Auto Scaling 组创建后，使用attach-traffic-sources命令将其附加到 Auto Scaling 组。

以下命令可将另一个目标组添加到同一个组。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2"
```

或者，要将经典负载均衡器附加到您的组，请在使用 create-auto-scaling-group 或 attach-traffic-sources 时指定 `--traffic-sources` 和 `--type` 选项，如以下示例所示。请将 *`my-classic-load-balancer`* 替换为经典负载均衡器的名称。对于 `--type` 选项，请指定 **`elb`** 的值。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

描述您的目标组或经典负载均衡器

要描述附加到 Auto Scaling 组的负载均衡器或目标组，请使用以下describe-traffic-sources命令。将 *my-asg* 替换为您的组名。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

该示例会返回附加到自动扩缩组的 Elastic Load Balancing 目标组的 ARN。

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1",
            "State": "InService",
            "Type": "elbv2"
        },
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2",
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

有关输出中的 `State` 字段说明，请参阅 验证负载均衡器的附加状态。

添加 Elastic Load Balancing 运行状况检查

要将 Elastic Load Balancing 运行状况检查添加到您的 Auto Scaling 组对实例执行的运行状况检查中，请使用以下update-auto-scaling-group命令并指定**ELB**为该`--health-check-type`选项的值。将 *my-asg* 替换为您的组名。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "ELB"
```

新实例通常需要时间进行短暂的预热，然后才能通过运行状况检查。如果宽限期没有提供足够的预热时间，则实例可能未准备好提供流量。Amazon A EC2 uto Scaling 可能会认为这些实例运行状况不佳并替换它们。

若要更新运行状况检查宽限期，请在使用 update-auto-scaling-group 时使用 `--health-check-grace-period` 选项，如以下示例所示。如果发现新实例运行状况不佳，则将其终止之前，将其替换*300*为保持运行状态的秒数。

```
--health-check-grace-period 300
```

有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

更改您的可用区

更改可用区时，应注意某些限制。有关更多信息，请参阅 添加可用区。

更改应用程序负载均衡器或网络负载均衡器的可用区

1. 在更改负载均衡器的可用区之前，最好先更新自动扩缩组的可用区，以验证您的实例类型是否可在指定区域中使用。

   要更新您的 Auto Scaling 组的可用区域，请使用以下update-auto-scaling-group命令。将示例子网 IDs 替换 IDs 为可用区中要启用的子网。用指定的子网替换先前启用的子网。将 *my-asg* 替换为您的组名。

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
     --vpc-zone-identifier "subnet-41767929,subnet-cb663da2,subnet-8360a9e7"
   ```

2. 使用以下describe-auto-scaling-groups命令验证新子网中的实例是否已启动。如果实例已启动，您将看到实例及其状态的列表。将 *my-asg* 替换为您的组名。

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

3. 使用以下 set-subnets 命令为您的负载均衡器指定子网。将示例子网 IDs 替换 IDs 为可用区中要启用的子网。每个可用区您只能指定一个子网。用指定的子网替换先前启用的子网。将 *my-lb-arn* 替换为负载均衡器的 ARN。

   ```
   aws elbv2 set-subnets --load-balancer-arn my-lb-arn \
     --subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
   ```

更改经典负载均衡器的可用区

1. 在更改负载均衡器的可用区之前，最好先更新自动扩缩组的可用区，以验证您的实例类型是否可在指定区域中使用。

   要更新您的 Auto Scaling 组的可用区域，请使用以下update-auto-scaling-group命令。将示例子网 IDs 替换 IDs 为可用区中要启用的子网。用指定的子网替换先前启用的子网。将 *my-asg* 替换为您的组名。

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
     --vpc-zone-identifier "subnet-41767929,subnet-cb663da2"
   ```

2. 使用以下describe-auto-scaling-groups命令验证新子网中的实例是否已启动。如果实例已启动，您将看到实例及其状态的列表。将 *my-asg* 替换为您的组名。

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

3. 使用以下 attach-load-balancer-to-subnet s 命令为您的 Classic Load Balancer 启用新的可用区。将示例子网 ID 替换为要启用的可用区的子网 ID。将 *my-lb* 替换为您负载均衡器的名称。

   ```
   aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \
     --subnets subnet-cb663da2
   ```

   要禁用可用区，请使用以下 detach-load-balancer-from-subnets 命令。将示例子网 ID 替换为要禁用的可用区的子网 ID。将 *my-lb* 替换为您负载均衡器的名称。

   ```
   aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \
     --subnets subnet-8360a9e7
   ```

分离目标组或经典负载均衡器

当您不再需要目标组时，以下detach-traffic-sources命令会将其从 Auto Scaling 组中分离出来。

对于 `--auto-scaling-group-name` 选项，请将 *my-asg* 替换为您的组名称。对于 `--traffic-sources` 选项，将示例 ARN 替换为应用程序负载均衡器、网络负载均衡器或网关负载均衡器的目标组的 ARN。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
```

```
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456"
```

要将经典负载均衡器与您的组分离，请指定 `--traffic-sources` 和 `--type` 选项，如以下示例所示。请将 *`my-classic-load-balancer`* 替换为经典负载均衡器的名称。对于 `--type` 选项，请指定 **`elb`** 的值。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

移除 Elastic Load Balancing 运行状况检查

要从 Auto Scaling 组中移除 Elastic Load Balancing 运行状况检查，请使用以下update-auto-scaling-group命令并指定**EC2**为该`--health-check-type`选项的值。将 *`my-asg`* 替换为您的组名。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

旧版命令

以下示例演示如何使用旧版 CLI 命令附加、分离和描述负载均衡器和目标组。本文档中将保留这些命令，以供希望使用它们的客户参考。我们继续支持旧版 CLI 命令，但我们建议您使用新版“流量来源”CLI 命令，该命令可以附加和分离多种流量源类型。您可以在同一自动扩缩组上同时使用旧版 CLI 命令和“流量来源”CLI 命令。

附加您的目标组或经典负载均衡器（遗留）

附加您的目标组

以下create-auto-scaling-group命令创建带有附加目标组的 Auto Scaling 组。为 Application Load Balancer、Network Load Balancer 或网关负载均衡器指定目标组的 Amazon Resource Name (ARN)。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-
targets/1234567890123456" \
```

```
  --min-size 1 --max-size 5
```

以下 attach-load-balancer-target-groups 命令将目标组附加到现有的 Auto Scaling 组。

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

附加您的经典负载均衡器

以下create-auto-scaling-group命令创建附带了 Classic Load Balancer 的 Auto Scaling 组。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --load-balancer-names "my-load-balancer" \
  --min-size 1 --max-size 5
```

以下attach-load-balancers命令将指定的 Classic Load Balancer 附加到现有的 Auto Scaling 组。

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

描述您的目标组或经典负载均衡器（旧版）

描述目标组

要描述与 Auto Scaling 群组关联的目标群组，请使用 describe-load-balancer-target-groups 命令。以下示例列出了的目标群体*my-asg*。

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

描述经典负载均衡器

要描述与 Auto Scaling 组关联的传统负载均衡器，请使用describe-load-balancers命令。以下示例列出了适用于的经典负载均衡器。*my-asg*

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

分离目标组或经典负载均衡器（旧版）

分离目标组

当你不再需要目标detach-load-balancer-target组时，以下- groups命令会将目标组从你的 Auto Scaling 组中分离出来。

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

分离经典负载均衡器

当您不再需要 Classic Load Balancer 时，以下detach-load-balancers命令会将其从您的 Auto Scaling 组中分离。

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

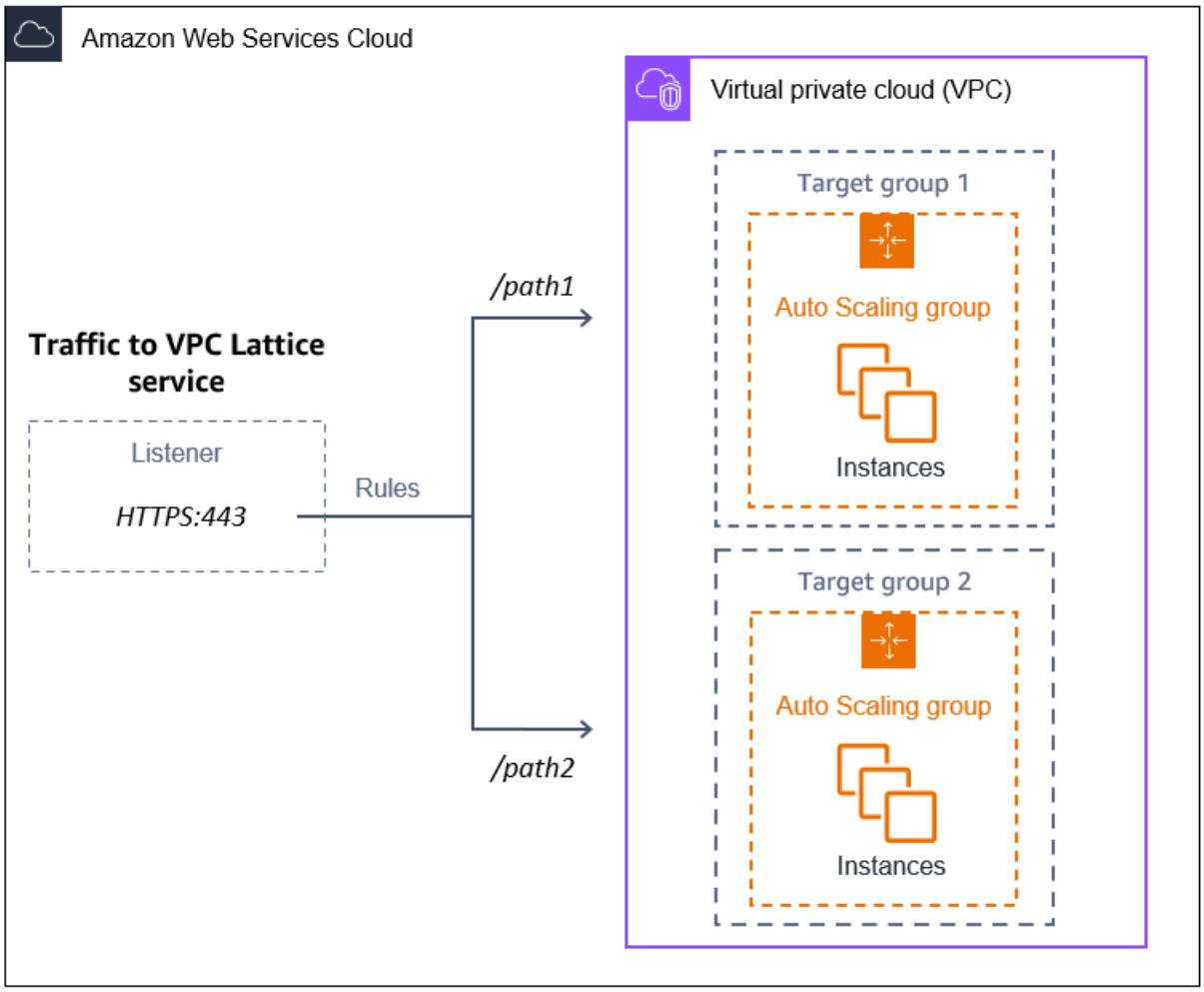
使用 VPC Lattice 目标组来管理流量

您可以使用 Amazon VPC Lattice 来管理应用程序和在不同资源（例如自动扩缩组或 Lambda 函数）上运行的服务之间的流量和 API 调用。VPC Lattice 是一项应用网络服务，可让您跨多个账户和虚拟私有云连接、保护和监控所有服务（VPCs）。要了解有关 VPC Lattice 的更多信息，请参阅什么是 VPC Lattice？

要开始使用 VPC Lattice，首先要创建必要的 VPC Lattice 资源，使与服务网络关联的 VPC 中的资源能够相互连接。这些资源包括服务、侦听器、侦听器规则和目标组。

要将自动扩缩组与 VPC Lattice 服务关联，请为将请求路由到按实例 ID 注册的实例的服务创建目标组、并将侦听器添加到向目标组发送请求的服务。然后将目标组附加到自动扩缩组。Amazon A EC2 uto Scaling 会自动将这些 EC2 实例注册为目标组的目标。之后，当 Amazon A EC2 uto Scaling 需要终止实例时，它会在终止之前自动从目标组注销该实例。

在您附加目标组之后，它将成为您的自动扩缩组的所有传入请求的入口点。如下图中的示例所示，然后可以使用为 VPC Lattice 服务指定的侦听器规则将传入请求路由到相应的目标组。



当流量通过 VPC Lattice 路由到您的自动扩缩组时，VPC Lattice 会使用循环负载均衡在组中的实例之间平衡请求。VPC Lattice 还会监控其已注册实例的运行状况，并且只将流量路由到运行状况良好的实例。

为了使您的实例可用于传入的请求，您可以选择将 VPC Lattice 运行状况检查添加到您的自动扩缩组中。这样，如果其中一个 EC2 实例出现故障，您的 Auto Scaling 组会自动启动一个新实例来替换它。VPC Lattice 运行状况检查的行为与 Elastic Load Balancing 运行状况检查的行为类似。Auto Scaling 组的默认运行 EC2 状况检查仅是运行状况检查。

要了解有关 VPC Lattice 的更多信息，请参阅博客上的 Amazon VPC Lattice 简化 Service-to-Service 连接、安全和监控——现已正式上 Amazon 线。

内容

- 验证您的 VPC Lattice 目标组的附件状态

做好准备将 VPC Lattice 目标组附加到您的自动扩缩组

将 VPC Lattice 目标组附加到您的自动扩缩组之前，您必须满足以下先决条件：

- 您必须已经创建了VPC Lattice服务网络、服务、侦听器和目标组。有关更多信息，请参阅 VPC Lattice 用户指南中的以下主题：
  - 服务网络
  - 服务
  - 侦听器
  - 目标组
- 目标组必须与您的 Auto Scaling 组位于相同 Amazon Web Services 账户的 VPC 和区域。
- 目标组必须指定的 `instance` 目标类型。使用 Auto Scaling 组时，无法指定 `ip` 的目标类型。
- 您必须拥有足够的 IAM 权限才能将目标组附加到自动扩缩组。以下示例策略显示了附加和分离目标组所需的最低权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:AttachTrafficSources",
                "autoscaling:DetachTrafficSources",
                "autoscaling:DescribeTrafficSources",
                "vpc-lattice:RegisterTargets",
                "vpc-lattice:DeregisterTargets"
            ],
            "Resource": "*"
        }
    ]
}
```

- 如果您的自动扩缩组的启动模板不包含 VPC Lattice 的正确设置，例如兼容的安全组，则必须更新启动模板。修改启动模板时，现有实例不会使用新设置进行更新。要更新现有实例，您可以启动实例刷新以替换实例。有关更多信息，请参阅 使用实例刷新更新自动扩缩组中的实例。

- 在您的自动扩缩组上启用 VPC Lattice 运行状况检查之前，您可以配置基于应用程序的运行状况检查，以验证您的应用程序是否按预期响应。有关更多信息，请参阅 VPC Lattice 用户指南中的目标群体的运行状况检查。

安全组：入站和出站规则

安全组充当关联 EC2 实例的防火墙，在实例级别控制入站和出站流量。

> **Note**
>
> 网络配置非常复杂，我们强烈建议您创建一个新的安全组以便与 VPC Lattice 结合使用。如果您需要与他们联系 Amazon Web Services 支持 ，它还可以更轻松地为您提供帮助。以下各节基于您遵循此建议的假设。
>
> 要详细了解如何为 VPC Lattice 创建可与自动扩缩组一起使用的安全组，请参阅 VPC Lattice 用户指南中的使用安全组控制流量。要解决流量问题，请查阅 VPC Lattice用户指南以获取更多信息。

有关如何创建安全组的信息，请参阅 Amazon EC2 用户指南中的创建安全组，并使用下表确定要选择的选项。

选项	值
名称	一个很容易记住的名字。
描述	有关描述可帮助您识别安全组。
VPC	与自动扩缩组相同的 VPC。

入站规则

当您创建一个安全组时，它没有入站规则。在您向安全组添加入站规则之前，不允许来自 VPC Lattice 服务网络内客户端的入站流量传输到您的实例。

要允许 VPC Lattice 服务网络中的客户端连接到您的自动扩缩组中的实例，必须正确设置您的自动扩缩组的安全组。在这种情况下，为其提供入站规则，允许来自 VPC Lattice Amazon 托管前缀列表名称的

流量，而不是特定 IP 地址的流量。VPC Lattice 前缀列表是 VPC Lattice 以 CIDR 表示法使用的一系列 IP 地址。有关更多信息，请参阅 Amazon VPC 用户指南中的使用Amazon托管前缀列表。

有关如何向安全组添加规则的信息，请参阅 Amazon VPC 用户指南中的配置安全组规则，并使用下表确定要选择的选项。

选项	值	
HTTP 规则	类型：HTTP 来源：com.amazonaws。*region*.vpc-lattice	
HTTPS 规则	类型：HTTPS 来源：com.amazonaws。*region*.vpc-lattice	

安全组是有状态的：它允许来自VPC Lattice服务网络中客户端的流量传输到您的自动扩缩组中的实例，然后将响应发回给之前离开的客户端。

出站规则

默认情况下，安全组包含允许所有出站流量的出站规则。您可以选择删除此默认规则并添加出站规则以满足特定的安全需求。

限制

- 不支持混合实例组。如果您尝试将 VPC Lattice 目标组附加到采用混合实例策略的自动扩缩组，则会收到错误消息目前，具有混合实例的自动扩缩组无法与 VPC 莱迪思服务集成。这是因为负载均衡算法会将负载均匀地分配到所有可用资源上，并假设实例足够相似，可以处理相等的负载。

将 VPC Lattice 目标组附加到您的自动扩缩组

本主题介绍如何将 VPC Lattice 目标组附加到自动扩缩组。它还描述了如何启用 VPC Lattice 运行状况检查，让 Amazon A EC2 uto Scaling 替换 VPC Lattice 报告为运行状况不佳的实例。

默认情况下，Amazon A EC2 uto Scaling 仅根据亚马逊运行状况检查替换运行 EC2 状况不佳或无法访问的实例。如果您启用 VPC Lattice 运行状况检查，则如果您附加到 A EC2 uto Scaling 组的任何 VPC

Lattice 目标组报告运行状况不佳，Amazon Auto Scaling 可以替换正在运行的实例。有关更多信息，请参阅 自动扩缩组中实例的运行状况检查。

> **Important**
>
> 在继续之前，请完成上一节中的所有先决条件。

附加 VPC Lattice 目标组

创建或更新组时，可以将一个或多个目标组附加到自动扩缩组。

Console

请按照本部分中的步骤，使用控制台来执行以下操作：

- 将 VPC Lattice 目标组附加到自动扩缩组
- 开启 VPC Lattice 的运行状况检查

要将 VPC Lattice 目标组附加到新的自动扩缩组

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 在屏幕顶部的导航栏中，选择您在其中创建了目标组的 Amazon Web Services 区域 。
3. 选择 Create Auto Scaling group（创建 Auto Scaling 组）。
4. 在步骤 1 和 2 中，选择您所需的选项，然后继续执行步骤 3：配置高级选项。
5. 要获取 VPC Lattice 集成选项，请选择连接到 VPC Lattice 服务。
6. 在选择 VPC Lattice 目标组下，选择您的目标组。
7. （可选）对于运行状况检查、其他运行状况检查类型，请选择启用 VPC Lattice 运行状况检查。
8. （可选）对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
9. 继续创建 Auto Scaling 组。创建自动扩缩组后，您的实例将自动注册到 VPC Lattice 目标组。

将 VPC Lattice 目标组附加到现有的自动扩缩组

使用以下过程将服务的目标组附加到现有的自动扩缩组。

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中您的自动扩缩组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在详细信息选项卡上，选择 VPC Lattice 集成选项，然后选择编辑。
4. 在 VPC Lattice 集成选项下，选择连接到 VPC Lattice 服务。
5. 在选择 VPC Lattice 目标组下，选择您的目标组。
6. 选择更新。

连接完目标群组后，您可以选择开启使用该群组的运行状况检查。

开启 VPC Lattice 运行状况检查

1. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
2. 对于运行状况检查、其他运行状况检查类型，请选择启用 VPC Lattice 运行状况检查。
3. 对于运行状况检查宽限期，输入时间长短（以秒为单位）。这段时间是 Amazon A EC2 uto Scaling 在实例进入状态后需要等待多长时间才能检查其运行`InService`状况。有关更多信息，请参阅 设置自动扩缩组的运行状况检查宽限期。
4. 选择更新。

Amazon CLI

按照本节中的步骤 Amazon CLI 使用：

- 将 VPC Lattice 目标组附加到自动扩缩组
- 开启 VPC Lattice 的运行状况检查

将 VPC Lattice 目标组附加到自动扩缩组

使用以下create-auto-scaling-group命令创建 Auto Scaling 组，并通过指定 VPC Lattice 目标组的亚马逊资源名称 (ARN) 来同时连接该组。

替换 `--auto-scaling-group-name`、`--vpc-zone-identifier`、`--min-size` 和 `--max-size` 的示例值。对于该 `--launch-template` 选项，将 *my-launch-template* 和 *1* 替换为您为注册到 VPC Lattice 目标组的实例创建的启动模板的名称和版本。对于 `--traffic-sources` 选项，请将 ARN 示例 VPC Lattice 目标组的 ARN。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

在VPC Lattice目标组创建完成后，使用以下attach-traffic-sources命令将其附加到Auto Scaling组。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

启用 VPC Lattice 的运行状况检查

如果您已为 VPC Lattice 目标组配置了基于应用程序的运行状况检查，则可以启用这些运行状况检查。使用带有`--health-check-type`选项且值为的create-auto-scaling-group或update-auto-scaling-group命令**`VPC_LATTICE`**。要为自动扩缩组执行的运行状况检查指定宽限期，请添加`--health-check-grace-period`选项并以秒为单位提供其值。

```
--health-check-type "VPC_LATTICE" --health-check-grace-period 60
```

分离 VPC Lattice 目标组

如果不再需要使用 VPC Lattice，请使用以下步骤将目标组与自动扩缩组分离。

Console

请按照本部分中的步骤，使用控制台来执行以下操作：

- 将 VPC Lattice 目标组与自动扩缩组分离
- 关闭 VPC Lattice 的运行状况检查

将 VPC Lattice 目标组与自动扩缩组分离

1. 在上打开亚马逊 EC2 控制台 https://console.aws.amazon.com/ec2/，然后从导航窗格中选择 A uto Scaling Gro ups。
2. 选中现有组旁边的复选框。

   这时将在页面底部打开一个拆分窗格。
3. 在详细信息选项卡上，选择 VPC Lattice 集成选项，然后选择编辑。
4. 在 VPC Lattice 集成选项下，选择目标组旁边的删除 (X) 图标。
5. 选择更新。

完成分离目标组后，您可以关闭 VPC Lattice 运行状况检查。

关闭 VPC Lattice 运行状况检查

1. 在 Details (详细信息) 选项卡上，选择 Health checks (运行状况检查)、Edit (编辑)。
2. 对于运行状况检查、其他运行状况检查类型，请取消选择启用 VPC Lattice 运行状况检查。
3. 选择更新。

Amazon CLI

按照本节中的步骤 Amazon CLI 使用：

- 将 VPC Lattice 目标组与自动扩缩组分离
- 关闭 VPC Lattice 的运行状况检查

当您不再需要目标组时，使用detach-traffic-sources命令将其与 Auto Scaling 组分离。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

要更新 Auto Scaling 组的运行状况检查，使其不再使用 VPC Lattice 运行状况检查，请使用update-auto-scaling-group命令。包括 `--health-check-type` 选项和 **`EC2`** 的值。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

验证您的 VPC Lattice 目标组的附件状态

将 VPC Lattice 目标组附加到自动扩缩组后，该目标组将在向该组注册实例时进入Adding状态。注册了组中的所有实例后，它进入 Added 状态。在至少一个注册实例通过运行状况检查后，它进入 InService 状态。当目标组处于InService状态时，Amazon A EC2 uto Scaling 可以终止并替换任何报告为运行状况不佳的实例。如果注册的实例均未通过运行状况检查（例如，由于未正确配置运行状况检查），目标组不会进入 InService 状态。Amazon A EC2 uto Scaling 不会终止和替换实例。

当分离服务的目标组时，它进入 Removing 状态，同时取消注册该组中的实例。实例在取消注册后仍保持运行。默认情况下，将启用连接耗尽（取消注册延迟）功能。如果启用了连接耗尽功能，则 VPC Lattice 将等待动态请求完成或最大超时到期（以先到者为准），然后再取消注册实例。

您可以使用 Amazon Command Line Interface (Amazon CLI) 或，验证附件状态 Amazon SDKs。您无法通过控制台来验证附加状态。

使用 Amazon CLI 来验证附件状态

以下describe-traffic-sources命令返回指定 Auto Scaling 组的所有流量源的连接状态。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

该示例返回附加到自动扩缩组的 VPC Lattice 目标组的 ARN 以及该元素中目标组的连接状态。State

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:vpc-lattice:region:account-
id:targetgroup/tg-0e2f2665eEXAMPLE",
            "State": "InService",
            "Type": "vpc-lattice"
        }
    ]
}
```

用于处理 EventBridge Auto Scaling 事件

Amazon EventBridge（前身为“ CloudWatch Events”）可帮助您设置事件驱动的规则，这些规则用于监控资源并启动使用其他 Amazon 服务的目标操作。

来自 Amazon A EC2 uto Scaling 的事件几乎是实时的。 EventBridge 您可以制定 EventBridge 规则，调用编程操作和通知以响应各种此类事件。例如，当实例处于启动或终止过程时，您可以调用 Amazon Lambda 函数来执行预配置的任务。

EventBridge 规则的目标可以包括 Amazon Lambda 函数、Amazon SNS 主题、API 目标 Amazon Web Services 账户、其他中的事件总线等等。有关支持的目标的信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge 目标。

首先，通过使用 Amazon SNS 主题和 EventBridge 规则的示例创建规则。 EventBridge 然后在用户启动实例刷新后，每当到达某个检查点时，Amazon SNS 都会通过电子邮件通知您。有关更多信息，请参阅 为实例刷新事件创建 EventBridge 规则。

内容

Amazon A EC2 uto Scaling 事件参考

使用 Amazon EventBridge，您可以创建匹配传入事件的规则，并将它们路由到目标进行处理。

内容

生命周期操作事件

当您向 Auto Scaling 组添加生命周期挂钩 EventBridge时，Amazon A EC2 uto Scaling 会在实例过渡到等待状态时向发送事件。事件会尽可能生成。

事件类型

横向扩展生命周期操作

以下示例事件显示，由于启动生命周期挂钩，Amazon A EC2 uto Scaling 将实例移至`Pending:Wait`状态。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

横向缩减生命周期操作

以下示例事件显示，由于终止生命周期挂钩，Amazon A EC2 uto Scaling 将实例移至`Terminating:Wait`状态。

> **Important**
>
> 当自动扩缩组在横向缩减时将实例退回暖池的情况下，将实例退回暖池也会生成 EC2 Instance-terminate Lifecycle Action 事件。当实例在横向缩减时移至等待状态的情况下，所传递的事件都将 WarmPool 作为 Destination 的值。有关更多信息，请参阅 Instance reuse policy。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-1234567890abcdef0",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata":"additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

扩缩成功的事件

以下示例演示扩缩成功的事件的事件类型。事件会尽可能生成。

事件类型

- 横向扩展成功的事件
- 横向缩减成功的事件

横向扩展成功的事件

以下示例事件显示 Amazon A EC2 uto Scaling 成功启动了一个实例。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
```

```
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Launching a new EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

横向缩减成功的事件

以下示例事件显示 Amazon A EC2 uto Scaling 成功终止了一个实例。

> **Important**
>
> 当自动扩缩组在横向缩减时将实例退回暖池的情况下，将实例退回暖池也会生成 EC2 Instance Terminate Successful 事件。实例成功退回暖池时传送的事件都将 WarmPool 作为 Destination 的值。有关更多信息，请参阅 Instance reuse policy。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
```

```
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Terminating EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

扩缩失败的事件

以下示例演示扩缩失败的事件的事件类型。事件会尽可能生成。

事件类型

- 横向扩展失败的事件
- 横向缩减失败的事件

横向扩展失败的事件

以下示例事件显示 Amazon A EC2 uto Scaling 未能启动实例。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Instance Launch Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

横向缩减失败的事件

以下示例事件显示 Amazon A EC2 uto Scaling 未能终止实例。

> **Important**
>
> 当自动扩缩组在横向缩减时将实例退回暖池的情况下，未能将实例退回暖池也会生成 EC2 Instance Terminate Unsuccessful 事件。实例未能成功退回暖池时传送的事件都将 WarmPool 作为 Destination 的值。有关更多信息，请参阅 Instance reuse policy。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Instance Terminate Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

实例刷新事件

以下示例演示实例刷新功能的事件。事件会尽可能生成。

事件类型

- 达到检查点
- 实例刷新已开始
- 实例刷新成功
- 实例刷新失败
- 实例刷新已取消
- 实例刷新回滚已开始
- 实例刷新回滚成功

- 实例刷新回滚失败

达到检查点

当已替换的实例数量达到为检查点定义的百分比阈值时，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
    "AutoScalingGroupName": "my-asg",
    "CheckpointPercentage": "50",
    "CheckpointDelay": "300"
  }
}
```

实例刷新已开始

当实例刷新状态更改为时`InProgress`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
```

```
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

实例刷新成功

当实例刷新状态更改为时`Successful`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

实例刷新失败

当实例刷新状态更改为时`Failed`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
```

```
    "AutoScalingGroupName": "my-asg"
  }
}
```

实例刷新已取消

当实例刷新状态更改为时`Cancelled`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

实例刷新回滚已开始

当实例刷新状态更改为时`RollbackInProgress`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
```

```
  }
}
```

实例刷新回滚成功

当实例刷新状态更改为时`RollbackSuccessful`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

实例刷新回滚失败

当实例刷新状态更改为时`Failed`，Amazon A EC2 uto Scaling 会发送以下事件。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
```

```
}
```

暖池示例事件类型和模式

Amazon A EC2 uto Scaling 支持亚马逊中的多种预定义模式 EventBridge。这简化了事件模式的创建方式。您可以在表单上选择字段值，然后为您 EventBridge 生成模式。目前，Amazon A EC2 uto Scaling 不支持使用温池的 Auto Scaling 组发出的任何事件的预定义模式。您必须以 JSON 对象的形式输入模式。这一部分和 为温水池活动创建 EventBridge 规则 主题演示例了如何使用事件模式来选择事件并将其发送到目标。

要创建 EventBridge 规则来筛选 Amazon A EC2 uto Scaling 发送到的与温池相关的事件 EventBridge，请在事件`detail`部分添加`Origin`和`Destination`字段。

它可以是以下 `Origin` 和 `Destination` 值之一：

`EC2` | `AutoScalingGroup` | `WarmPool`

内容

示例事件

当您向 Auto Scaling 组添加生命周期挂钩 EventBridge时，Amazon A EC2 uto Scaling 会在实例过渡到等待状态时向发送事件。有关更多信息，请参阅 在自动扩缩组中将生命周期挂钩与暖池一起使用。

本节包含自动扩缩组具有暖池时的这些事件的示例。尽最大努力发布事件。

> **Note**
>
> 有关 Amazon A EC2 uto Scaling 在成功扩展 EventBridge 时发送到的事件，请参阅扩缩成功的事件。有关扩缩失败时的事件，请参阅 扩缩失败的事件。

事件示例

横向扩展生命周期操作

当实例在横向扩展时转换为等待状态的情况下，所传递的事件都将 EC2 Instance-launch Lifecycle Action 作为 detail-type 的值。在 detail 对象中，Origin 和 Destination 属性的值显示实例的来源和去向。

在此横向扩展事件示例中，启动了一个新实例，由于该实例已添加到暖池中，因此将其状态更改为 Warmed:Pending:Wait。有关更多信息，请参阅 暖池中实例的生命周期状态转换。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

在此示例事件中，实例的状态在其被从暖池中添加到自动扩缩组时更改为 Pending:Wait。有关更多信息，请参阅 暖池中实例的生命周期状态转换。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
```

```
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "WarmPool",
    "Destination": "AutoScalingGroup"
  }
}
```

横向缩减生命周期操作

当实例在横向缩减时转换为等待状态的情况下，所传递的事件都将 EC2 Instance-terminate Lifecycle Action 作为 detail-type 的值。在 detail 对象中，Origin 和 Destination 属性的值显示实例的来源和去向。

在此示例事件中，实例的状态因其被退回到暖池而更改为 Warmed:Pending:Wait。有关更多信息，请参阅 暖池中实例的生命周期状态转换。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-03-28T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-termination-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
```

```
    "Origin": "AutoScalingGroup",
    "Destination": "WarmPool"
  }
}
```

示例事件模式

上一节提供了 Amazon A EC2 uto Scaling 发出的示例事件。

EventBridge 事件模式与它们匹配的事件具有相同的结构。模式引用了您要匹配的字段，并提供您所查找的值。

事件中的下列字段构成规则中定义的事件模式以调用操作：

`"source": "aws.autoscaling"`

标识该事件来自 Amazon A EC2 uto Scaling。

`"detail-type": "EC2 Instance-launch Lifecycle Action"`

识别事件类型。

`"Origin": "EC2"`

标识实例的来源。

`"Destination": "WarmPool"`

标识实例的目标位置。

使用以下示例事件模式捕获与进入暖池的实例相关联的所有EC2 Instance-launch Lifecycle Action事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

使用以下示例事件模式捕获与因横向扩展事件而离开暖池的实例有关的所有EC2 Instance-launch Lifecycle Action事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "WarmPool" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

使用以下示例事件模式捕获与直接启动到自动扩缩组的实例相关联的所有EC2 Instance-launch Lifecycle Action事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

使用以下示例事件模式捕获与在横向缩减时退回暖池的实例有关的所有EC2 Instance-terminate Lifecycle Action事件。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
      "Origin": [ "AutoScalingGroup" ],
      "Destination": [ "WarmPool" ]
   }
}
```

使用以下示例事件模式捕获与 EC2 Instance-launch Lifecycle Action 关联的所有相关事件，无论起点或目的地如何。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

使用 Amazon EventBridge 规则自动执行操作

当 Amazon A EC2 uto Scaling 发出事件时，将以 JSON 文件 EventBridge 形式向亚马逊发送事件通知。您可以编写一条 EventBridge 规则，自动执行当事件模式与规则匹配时要执行的操作。如果 EventBridge 检测到的事件模式与规则中定义的模式相匹配，则 EventBridge 调用规则中指定的一个或多个目标。

您可以使用这一部分的示例过程作为起点。

以下文档也可能会非常有用：

- 要在实例启动时或使用 Lambda 函数终止实例之前对实例执行自定义操作，请参阅教程：配置调用 Lambda 函数的生命周期钩子。
- 要对使用记录的 API 调用调用 Lambda 函数 CloudTrail，请参阅亚马逊 EventBridge 用户指南 EventBridge中的教程：使用记录 Amazon API 调用。
- 有关如何创建事件规则的更多信息，请参阅 Amazon EventBridge 用户指南中的创建对事件做出反应的 Amazon EventBridge 规则。

主题

为实例刷新事件创建 EventBridge 规则

以下示例创建了发送电子邮件通知的 EventBridge 规则。每次在实例刷新期间到达某个检查点，自动扩缩组发出事件时，系统都会执行此操作。包括使用 Amazon SNS 设置电子邮件通知的过程。要使用 Amazon SNS 发送电子邮件通知，必须先创建一个主题，然后用您的电子邮件地址订阅该主题。

有关实例刷新功能的更多信息，请参阅使用实例刷新更新自动扩缩组中的实例。

创建 Amazon SNS 主题

SNS 主题是一个逻辑接入点，即 Auto Scaling 组用来发送通知的通信通道。您可通过为主题指定名称来创建主题。

主题名称必须满足以下要求：

- 使用 1 到 256 个字符

- 包含大写和小写 ASCII 字母、数字、下划线或连字符

有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的创建 Amazon SNS 主题。

订阅 Amazon SNS 主题

要接收您的 Auto Scaling 组发送到该主题的通知，必须让一个终端节点订阅该主题。在此过程中，对于 Endpoint，指定您要从中接收来自 Amazon A EC2 uto Scaling 的通知的电子邮件地址。

有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的订阅 Amazon SNS 主题。

确认您的 Amazon SNS 订阅

Amazon SNS 向在上一步骤中指定的电子邮件地址发送确认电子邮件。

在继续下一步之前，请务必打开 Amazon 通知中的电子邮件并选择确认订阅的链接。

您将收到来自的确认消息。 Amazon Amazon SNS 现已配置为接收通知并以电子邮件的形式将通知发送到指定的电子邮件地址。

将事件路由到您的 Amazon SNS 主题

创建匹配选定事件的规则，并将它们路由到您的 Amazon SNS 主题，以通知订阅的电子邮件地址。

创建向您的 Amazon SNS 主题发送通知的规则

1. 打开亚马逊 EventBridge 控制台，网址为https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 对于定义规则详细信息，请执行以下操作：

   a. 输入规则的 Name (名称) 和“Description (描述)”（可选）。

      规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。

   b. 对于事件总线，选择默认。当你账户中的 Amazon 服务生成事件时，它总是会进入你账户的默认事件总线。

   c. 对于规则类型，选择具有事件模式的规则。

d. 选择下一步。

5. 对于 Build event pattern（构建事件模式），执行以下操作：

   a. 对于事件来源，选择Amazon 事件或 EventBridge 合作伙伴事件。

   b. 对于 Event pattern（事件模式），执行以下操作：

      i. 对于事件源，选择 Amazon Web Services 服务。

      ii. 对于 Amazon Web Services 服务，选择 Auto Scaling。

      iii. 对于事件类型，选择实例刷新。

      iv. 预设情况下，该规则与任何实例刷新事件匹配。要创建在实例刷新期间到达检查点时通知您的规则，请选择特定实例事件，然后选择已到达 EC2 Auto S caling 实例刷新检查点。

      v. 默认情况下，该规则与区域中任何 Auto Scaling 组匹配。若要使该规则与特定 Auto Scaling 组匹配，请选择 特定组名称并选择一个或多个 Auto Scaling 组。

      vi. 选择下一步。

6. 对于 Select target(s)（选择目标），请执行以下操作：

   a. 对于 Target types（目标类型），选择 Amazon Web Services 服务。

   b. 对于 Select a target（选择一个目标），选择 SNS topic（SNS 主题）。

   c. 对于 Topic（主题），选择您的 Amazon SNS 主题。

   d. （可选）在 Additional settings（其他设置）下，您可以选择配置其他设置。有关更多信息，请参阅 EventBridge 《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的亚马逊规则。

   e. 选择下一步。

7. （可选）对于 Tags（标签），您可以选择向规则分配一个或多个标签，然后选择 Next（下一步）。

8. 对于 Review and create（检查并创建），检查规则的详细信息并根据需要对其进行修改。然后选择 Create rule（创建规则）。

为温水池活动创建 EventBridge 规则

以下示例创建了调用编程操作的 EventBridge 规则。每当您的自动扩缩组在有新实例被添加到暖池时发出事件时，都会执行此操作。

在创建规则之前，请创建要将该规则用作目标的 Amazon Lambda 函数。您必须将此函数指定为该规则的目标。以下过程仅提供创建规则的步骤，该 EventBridge 规则在新实例进入温池时起作用。有关演

示如何创建可在传入的事件与规则匹配时进行调用的简单 Lambda 函数的入门教程，请参阅教程：配置调用 Lambda 函数的生命周期钩子。

有关创建和使用暖池的更多信息，请参阅 使用暖池减少启动时间较长的应用程序的延迟。

创建会调用 Lambda 函数的事件规则

1. 打开亚马逊 EventBridge 控制台，网址为https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 对于定义规则详细信息，请执行以下操作：

   a. 输入规则的 Name (名称) 和“Description (描述)”（可选）。

      规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。

   b. 对于事件总线，选择默认。当您的账户 Amazon Web Services 服务 中的某项生成事件时，它始终会转到您账户的默认事件总线。

   c. 对于规则类型，选择具有事件模式的规则。

   d. 选择下一步。

5. 对于 Build event pattern（构建事件模式），执行以下操作：

   a. 对于事件来源，选择Amazon 事件或 EventBridge 合作伙伴事件。

   b. 对于事件模式，选择自定义模式（JSON 编辑器），然后将以下模式粘贴到事件模式框中，将中的*italics*文本替换为您的 Auto Scaling 组的名称。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ],
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

      要创建与其他事件匹配的规则，请修改事件模式。有关更多信息，请参阅 示例事件模式。

   c. 选择下一步。

6. 对于 Select target(s)（选择目标），请执行以下操作：

a. 对于 Target types（目标类型），选择 Amazon Web Services 服务。

b. 对于 Select a target（选择目标），选择 Lambda function（Lambda 函数）。

c. 然后，对于 Function（函数），选择要将事件发送到的函数。

d. （可选）对于 Configure version/alias（配置版本/别名），输入目标 Lambda 函数的版本和别名设置。

e. （可选）对于 Additional settings（其他设置），视应用程序的情况输入任何其他设置。有关更多信息，请参阅 EventBridge 《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的亚马逊规则。

f. 选择下一步。

7. （可选）对于 Tags（标签），您可以选择向规则分配一个或多个标签，然后选择 Next（下一步）。

8. 对于 Review and create（检查并创建），检查规则的详细信息并根据需要对其进行修改。然后选择 Create rule（创建规则）。

使用 Amazon VPC 为 Auto Scaling 实例提供网络连接

Amazon Virtual Private Cloud（亚马逊 VPC）是一项服务，可让您在您定义的逻辑隔离的虚拟网络中启动 Auto Scaling 组等 Amazon 资源。

Amazon VPC 中的子网是可用区的一个部分，由 VPC 的 IP 地址范围段定义。您可以根据您的安全和运行需求，使用子网对您的实例进行分组。子网完全位于其最初创建时所在的可用区内。在子网中启动 Auto Scaling 实例。

要支持 Internet 与您的子网中的实例之间的通信，您必须创建一个 Internet 网关并将其附加到您的 VPC。互联网网关使您在子网中的资源能够通过 Amazon EC2 网络边缘连接到互联网。如果一个子网的流量被路由到 Internet 网关，这个子网便是公有子网。如果子网的流量不路由到 Internet 网关，则子网称为私有 子网。请将公有子网用于必须连接到 Internet 的资源，并将私有子网用于不需要连接到 Internet 的资源。有关向 VPC 中的实例授予互联网访问的更多信息，请参阅 Amazon VPC 用户指南中的访问互联网。

内容

- VPC 中的 IP 寻址
- VPC 中的网络接口
- 实例部署租期
- Amazon Outposts
- 更多可供学习的资源 VPCs

默认 VPC

如果您是在 2013 年 12 月 4 日 Amazon Web Services 账户 之后创建的，或者您要在新的中创建 Auto Scaling 组 Amazon Web Services 区域，我们会为您创建一个默认 VPC。默认 VPC 随每个可用区中的默认子网提供。如果您有默认 VPC，则在默认情况下，将在默认 VPC 中创建 Auto Scaling 组。

您可以 VPCs 在 Amazon VPC 控制台的我的 VPCs 页面上查看您的。

有关默认 VPC 的更多信息，请参阅 Amazon VPC 用户指南 VPCs中的默认。

非默认 VPC

您可以前往中的 VPC 控制面板页面 Amazon Web Services Management Console 并选择创建 VPC，选择创建更多 VPC。 VPCs

有关更多信息，请参阅《Amazon VPC 用户指南》。

> **Note**
>
> VPC 跨越其 Amazon Web Services 区域中的所有可用区。向 VPC 添加子网时，请选择多个可用区以确保在这些子网中托管的资源具有高可用性。可用区是 Amazon Web Services 区域中一个或多个具有冗余电源、网络和连接的离散数据中心。可用区可为生产级应用程序提供高可用性、容错能力和可扩展性。

选择 VPC 子网时的注意事项

在为自动扩缩组选择 VPC 子网时，请注意以下事项：

- 如果您要将 Elastic Load Balancing 负载均衡器附加到 Auto Scaling 组，则可以在公有或私有子网中启动实例。但是，只能在公有子网中创建负载均衡器。
- 如果您直接通过 SSH 访问 Auto Scaling 实例，则只能在公有子网中启动实例。

- 如果您使用 Amazon Systems Manager 会话管理器访问无入口 Auto Scaling 实例，则这些实例可以在公有或私有子网中启动。
- 如果您使用的是私有子网，则可以使用公有 NAT 网关允许 Auto Scaling 实例访问互联网。
- 预设情况下，默认 VPC 中的默认子网为公有子网。

VPC 中的 IP 寻址

在 VPC 中启动 Auto Scaling 实例时，您的实例将从分配启动实例的子网 CIDR 范围自动分配私有 IP 地址。这样，这些实例就能够与 VPC 中的其他实例通信。

您可以配置启动模板或启动配置，为您的实例分配公有 IPv4地址。为您的实例分配公有 IP 地址使它们能够与互联网或其他 Amazon 服务进行通信。

当您在配置为自动分配 IPv6 地址的子网中启动实例时，它们会同时接收 IPv4 和 IPv6 地址。否则，他们只会收到 IPv4 地址。有关更多信息，请参阅 Amazon EC2 用户指南中的IPv6地址。

有关为您的 VPC 或子网指定 CIDR 范围的更多信息，请参阅 Amazon VPC 用户指南。

当您使用指定其他网络接口的启动模板时，Amazon A EC2 uto Scaling 可以在实例启动时自动分配额外的私有 IP 地址。每个网络接口都将从启动实例的子网的 CIDR 范围中获得单个私有 IP 地址。在这种情况下，系统无法再为主网络接口自动分配公共 IPv4 地址。除非您将可用的弹性 IP IPv4 地址关联到 Auto Scaling 实例，否则您将无法通过公共地址连接到您的实例。

VPC 中的网络接口

VPC 中的每个实例具有一个默认网络接口（主网络接口）。您无法从实例断开主网络接口。您可以创建其他网络接口并将其挂载至您的 VPC 中的任何实例。您可以挂载的网络接口数因实例类型而有所差异。

使用启动模板启动实例时，您可以指定其他网络接口。但是，启动具有多个网络接口的 Auto Scaling 实例会自动在与实例相同的子网中创建每个接口。这是因为 Amazon A EC2 uto Scaling 会忽略启动模板中定义的子网，而改用 Auto Scaling 组中指定的子网。有关更多信息，请参阅为自动扩缩组创建启动模板。

如果创建来自同一子网的两个或多个网络接口或将其连接到一个实例，可能会遇到非对称路由等联网问题，尤其是使用非 Amazon Linux 变体的实例。如果需要此类配置，则必须在操作系统中配置辅助网络接口。有关示例，请参阅如何使我的辅助网络接口在我的 Ubuntu EC2 实例中运行？ 在 Amazon 知识中心中。

实例部署租期

预设情况下，VPC 中的所有实例将作为共享租期实例运行。Amazon A EC2 uto Scaling 还支持专用实例和专用主机。有关更多信息，请参阅 使用高级设置创建启动模板。

Amazon Outposts

Amazon Outposts 使用可在 Amazon 该区域访问的 VPC 组件（包括互联网网关、虚拟私有网关、Amazon VPC 传输网关和 VPC 终端节点）将 Amazon VPC 从一个区域扩展到前哨站。Outpost 位于该区域内的一个可用区中，是该可用区的延伸，让您可以用来实现弹性。

有关更多信息，请参阅 用户指南。Amazon Outposts

更多可供学习的资源 VPCs

使用以下主题了解有关 VPCs 和子网的更多信息。

- VPC 中的私有子网
  - 示例：在私有子网中部署服务器并且具有 NAT 中的 VPC
  - NAT 网关
- VPC 中的公有子网
  - 示例：用于测试环境的 VPC
  - 示例：用于 Web 和数据库服务器的 VPC
- Application Load Balancer 的子网
  - 您的负载均衡器的子网
- 一般 VPC 信息
  - Amazon VPC User Guide
  - VPCs 使用 VPC 对等互连进行连接
  - 弹性网络接口
  - 使用 VPC 终端节点建立私有连接

亚马逊 A EC2 uto Scaling 中的安全

云安全 Amazon 是重中之重。作为 Amazon 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 Amazon 的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础架构。 Amazon 还为您提供可以安全使用的服务。作为的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon A EC2 uto Scaling 的合规计划，请查看Amazon 范围内按合规计划Amazon 提供的服务。
- 云端安全-您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Amazon A EC2 uto Scaling 时如何应用分担责任模型。以下主题向您展示了如何配置 Amazon A EC2 uto Scaling 以实现您的安全和合规目标。您还将学习如何使用其他 Amazon 服务来帮助您监控和保护您的 Amazon A EC2 uto Scaling 资源。

主题

Amazon A EC2 uto Scaling 中的基础设施安全

作为一项托管服务，Amazon A EC2 uto Scaling 受到 Amazon 全球网络安全的保护。有关 Amazon 安全服务以及如何 Amazon 保护基础设施的信息，请参阅Amazon 云安全。要使用基础设施安全的最佳实践来设计您的 Amazon 环境，请参阅 S Amazon ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 Amazon 已发布的 API 调用通过网络访问 Amazon A EC2 uto Scaling。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service（Amazon STS）生成临时安全凭证来对请求进行签名。

您也可以将虚拟私有云 (VPC) 终端节点用于 Amazon A EC2 uto Scaling。接口 VPC 终端节点使您的 Amazon VPC 资源能够使用其私有 IP 地址访问 Amazon A EC2 uto Scaling，而无需接触公共互联网。有关更多信息，请参阅 Amazon A EC2 uto Scaling 和接口 VPC 终端节点

相关资源

有关亚马逊提供的隔离服务流量的功能的信息 EC2，请参阅《亚马逊 EC2 用户指南》中的 “亚马逊 EC2基础设施安全”。

Amazon A EC2 uto Scaling 中的弹性

Amazon 全球基础设施是围绕 Amazon Web Services 区域 可用区构建的。 Amazon Web Services 区域 提供多个物理分隔和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon Web Services 区域 和可用区的更多信息，请参阅Amazon 全球基础设施。

要从可用区设计的地理冗余中获益，请执行以下操作：

- 在多个可用区之间跨越您的自动扩缩组。
- 在每个可用区内至少维护一个实例。
- 附加负载均衡器以在同一可用区之间分配传入流量。如果您使用 Application Load Balancer，请启用跨区域负载均衡，确保每个 EC2 实例获得的流量相似。这有助于限制失效转移事件期间增加的负载对现有实例的影响，并比没有跨区域负载平衡时具有更高的弹性。
- 确保正确配置了 Elastic Load Balancing 运行状况检查，并在自动扩缩组上启用了这些检查。然后，如果实例未通过运行状况检查，Elastic Load Balancing 将停止向其发送流量并将流量重新路由到运行状况良好的实例，而 Amazon A EC2 uto Scaling 会替换运行状况不佳的实例。

Amazon A EC2 uto Scaling 通过以下方式帮助支持您的应用程序弹性需求：

- 检查实例是否存在运行状况和可访问性问题。如果某个实例运行状况不佳，则它将自动终止该实例，并启动新实例。
- 如果动态扩展策略生效，则根据传入流量自动扩展容量。
- 检测支持扩展策略的 Amazon CloudWatch 指标的可靠性问题，并在可靠指标不可用时（例如缺少数据点时）暂停缩容活动。
- 尝试在您的组扩展时在每个已启用的可用区中维持相同数量的实例。
- 使用可用区以保持高可用性。当可用区域变得不健康时，Amazon A EC2 uto Scaling 会执行以下操作：
  - 在为你的 Auto Scaling 组启用的不同可用区中启动新实例。
  - 当运行状况不佳的可用区恢复正常状态时，在所有已启用的可用区中重新分配实例。
- 如果某个实例无法在指定的可用区中启动，则继续尝试在其他已启用的可用区中启动实例。
- 自动向与自动扩缩组关联的负载均衡器注册和取消注册实例。这样，您无需单独注册和取消注册实例。
- Amazon A EC2 uto Scaling 服务的控制平面中断 APIs 不会影响现有 Auto Scaling 组的扩展。

相关资源

有关 Amazon EBS 提供的帮助支持数据弹性需求的功能的信息，请参阅《Amazon EBS User Guide》中的 Resilience in Amazon Elastic Block Store。

Amazon A EC2 uto Scaling 中的数据保护

Amazon 分适用于 Amazon A EC2 uto Scaling 中的数据保护。如本模型所述 Amazon ，负责保护运行所有内容的全球基础架构 Amazon Web Services 云。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 Amazon Web Services 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 Amazon 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 Amazon CloudTrail。有关使用 CloudTrail 跟踪捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的使用跟 CloudTrail 踪。

- 使用 Amazon 加密解决方案以及其中的所有默认安全控件 Amazon Web Services 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 Amazon 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您 Amazon Web Services 服务 使用控制台、API 或与 Amazon A EC2 uto Scaling 或其他 Amazon CLI机构合作时 Amazon SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

当您启动 Amazon EC2 实例时，您可以选择将用户数据传递给该实例，以便在实例启动时进行额外配置。我们还建议您不要在将传递到实例的用户数据中包含机密或敏感信息。

用于加密 Amazon KMS keys Amazon EBS 卷

您可以将 Auto Scaling 组配置为使用 Amazon KMS keys对存储在云中的 Amazon EBS 卷数据进行加密。Amazon A EC2 uto Scaling 支持 Amazon 托管密钥和客户托管密钥来加密您的数据。请注意，当您使用启动配置时，指定客户托管密钥的 `KmsKeyId` 选项不可用。要指定您的客户托管密钥，请改用启动模板。有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。有关如何创建、存储和管理 Amazon KMS 加密密钥的信息，请参阅Amazon Key Management Service 开发者指南。

在设置启动模板或启动配置之前，您还可以在 EBS 支持的 AMI 中配置客户托管密钥，或在预设情况下使用加密来强制对您创建的新 EBS 卷和快照副本进行加密。有关更多信息，请参阅《亚马逊用户指南》 AMIs中的 “在 EBS 支持下使用加密” 和 “亚马逊 EBS EC2 用户指南” 中的 “默认加密”。

> **Note**
>
> 有关如何在使用客户托管密钥进行加密时设置需要启动 Auto Scaling 实例所需的密钥策略的信息，请参阅 使用加密卷所需的 Amazon KMS 密钥策略。

相关资源

有关 Amazon EBS 提供的数据保护指南，请参阅《Amazon EBS User Guide》中的 Data protection in Amazon Elastic Block Store。

使用加密卷所需的 Amazon KMS 密钥策略

Amazon A EC2 uto Scaling 使用与服务相关的角色向其他 Amazon Web Services 服务角色委派权限。Amazon A EC2 uto Scaling 服务相关角色是预定义的，包括 Amazon A EC2 uto Scaling 代表您呼叫他人所需的权限。 Amazon Web Services 服务 预定义的权限还包括对您的访问权限 Amazon 托管式密钥。但是，它们不包括对客户管理密钥的访问权限，因此您可以保持对这些密钥的完全控制。

本主题介绍当您为 Amazon EBS 加密指定客户托管密钥时如何设置启动 Auto Scaling 实例所需的密钥策略。

> **Note**
>
> Amazon A EC2 uto Scaling 无需额外授权即可使用默认设置 Amazon 托管式密钥 来保护您账户中的加密卷。

内容

概览

当 Amazon A EC2 uto Scaling 启动实例时，以下内容 Amazon KMS keys 可用于亚马逊 EBS 加密：

- Amazon 托管式密钥：您的账户中 Amazon EBS 创建、拥有和管理的加密密钥。这是新账户的默认加密密钥。除非您指定客户托管密钥，否则将使用加密。 Amazon 托管式密钥
- 客户托管密钥：您创建、拥有和管理的自定义加密密钥。有关更多信息，请参阅 Amazon Key Management Service 开发人员指南中的创建密钥。

  注意：密钥必须是对称的。Amazon EBS 不支持非对称客户托管密钥。

在创建加密的快照或指定加密卷的启动模板，或者默认启用加密时，您可以配置客户托管密钥。

配置密钥策略

您的 KMS 密钥必须具有密钥策略，允许 Amazon A EC2 uto Scaling 启动使用客户托管密钥加密的 Amazon EBS 卷的实例。

使用本页上的示例配置密钥策略，让 Amazon A EC2 uto Scaling 能够访问您的客户托管密钥。您可以在创建密钥时或以后的某个时间修改客户托管密钥的密钥策略。

要使其与 Amazon A EC2 uto Scaling 配合使用，您必须至少在密钥策略中添加两份政策声明。

- 第一条语句允许在 `Principal` 元素中指定的 IAM 身份直接使用客户托管密钥。它包括对密钥执行 Amazon KMS `Encrypt`、`Decrypt`、`ReEncrypt*`、`GenerateDataKey*`、和`DescribeKey`操作的权限。
- 第二条语句允许`Principal`元素中指定的 IAM 身份使用该`CreateGrant`操作生成授权，将其自己的权限子集委托给与 Amazon KMS 或其他委托人集成的权限。 Amazon Web Services 服务 这样，他们可以使用密钥代表您创建加密的资源。

当您向密钥策略添加新的策略语句时，不要更改策略中任何已存在的语句。

对于以下每个示例，必须替换的参数（例如密钥 ID 或服务相关角色的名称）显示为*`user placeholder text`*。在大多数情况下，您可以将服务相关角色的名称替换为 Amazon A EC2 uto Scaling 服务相关角色的名称。

有关更多信息，请参阅以下资源：

- 要使用创建密钥 Amazon CLI，请参阅创建密钥。
- 要使用更新密钥策略 Amazon CLI，请参阅put-key-policy。
- 要查找密钥 ID 和 Amazon Resource Name (ARN)，请参阅 Amazon Key Management Service 开发人员指南中的查找密钥 ID 和 ARN。
- 有关 Amazon A EC2 uto Scaling 服务相关角色的信息，请参阅Amazon A EC2 uto Scaling 的服务相关角色。
- 有关 Amazon EBS 加密和 KMS 的一般信息，请参阅《Amazon EBS User Guide》中的 Amazon EBS encryption 和《Amazon Key Management Service Developer Guide》https://docs.amazonaws.cn/kms/latest/developerguide/。

示例 1：允许访问客户托管密钥的关键策略部分

将以下两个策略语句添加到客户托管密钥的密钥策略中，同时将示例 ARN 替换为允许访问密钥的相应服务相关角色的 ARN。在本例中，策略部分为名为 AWSServiceRoleForAutoScaling 的服务相关角色提供使用客户托管密钥的权限。

```
{
   "Sid": "Allow service-linked role use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
       ]
   },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
   ],
   "Resource": "*"
}
```

```
{
   "Sid": "Allow attachment of persistent resources",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
       ]
   },
   "Action": [
       "kms:CreateGrant"
   ],
   "Resource": "*",
   "Condition": {
       "Bool": {
           "kms:GrantIsForAWSResource": true
       }
    }
```

```
}
```

示例 2：允许跨账户访问客户托管密钥的关键策略部分

如果您在与自动扩缩组不同的账户中创建了客户管理型密钥，则必须组合使用授权和密钥策略以允许对该密钥的跨账户存取。

必须按以下顺序完成两个步骤：

1. 首先，将以下两个策略语句添加到客户管理型密钥的密钥策略中。将示例 ARN 替换为另一个账户的 ARN，并确保将其*111122223333*替换为要在中创建 Auto Scaling 组的实际账户 ID。 Amazon Web Services 账户 这将允许您向指定账户中的 IAM 用户或角色授予使用下面的 CLI 命令为密钥创建授权的权限。但这本身并不会向任何用户授予对密钥的访问权限。

```
{
   "Sid": "Allow external account 111122223333 use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
   ],
   "Resource": "*"
}
```

```
{
   "Sid": "Allow attachment of persistent resources in external
 account 111122223333",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
```

```
        "kms:CreateGrant"
    ],
    "Resource": "*"
}
```

2. 然后从您要在其中创建自动扩缩组的账户中，创建一个授权以将相关权限委托给相应的服务相关角色。该授权的 `Grantee Principal` 元素是相应的服务相关角色的 ARN。`key-id` 是密钥的 ARN。

   以下是 create-grant CL I 命令示例，该命令向账户AWSServiceRoleForAutoScaling中指定的服务相关角色授予在账户中使用客户托管密钥的*111122223333*权限。*444455556666*

   ```
   aws kms create-grant \
     --region us-west-2 \
     --key-id arn:aws:kms:us-
   west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
     --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/
   autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
     --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
    "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
   ```

   要使此命令成功，发出请求的用户必须具有执行 `CreateGrant` 操作的权限。

   以下示例 IAM 策略允许账户中的 IAM 身份（用户或角色）*111122223333*为账户中的客户托管密钥创建授权*444455556666*。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
         "Effect": "Allow",
         "Action": "kms:CreateGrant",
         "Resource": "arn:aws:kms:us-
   west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
       }
     ]
   }
   ```

   有关为不同 Amazon Web Services 账户中的 KMS 密钥创建授权的更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的 Amazon KMS中的授权。

> **Important**
>
> 指定为被授权者委托人的服务相关角色名称必须是现有角色的名称。创建授权后，为确保该授权允许 Amazon A EC2 uto Scaling 使用指定的 KMS 密钥，请勿删除和重新创建服务相关角色。

在 Amazon KMS 控制台中编辑密钥策略

之前部分中的示例仅显示如何向密钥策略添加语句，这只是更改密钥策略的一种方法。更改密钥策略的最简单方法是使用 Amazon KMS 控制台的默认密钥策略视图，并将 IAM 身份（用户或角色）设为相应密钥策略的关键用户之一。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的使用 Amazon Web Services Management Console 默认视图。

> **Important**
>
> 请务必谨慎。控制台的默认视图策略声明包括对客户托管密钥执行 Amazon KMS Revoke操作的权限。如果您授予对账户中客户托管密钥的 Amazon Web Services 账户 访问权限，但又不小心撤销了授予他们此权限的授权，则外部用户将无法再访问他们的加密数据或用于加密其数据的密钥。

适用于 Amazon A EC2 uto Scaling 的身份和访问管理

Amazon Identity and Access Management (IAM) Amazon Web Services 服务 可帮助管理员安全地控制对 Amazon 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 Amazon A EC2 uto Scaling 资源。您可以使用 IAM Amazon Web Services 服务 ，无需支付额外费用。

要使用 Amazon A EC2 uto Scaling，你需要一个 Amazon Web Services 账户 和用于登录账户的安全证书。有关更多信息，请参阅《IAM 用户指南》中的Amazon 安全凭证。

有关完整的 IAM 文档，请参阅 IAM 用户指南。

访问控制

您可以拥有有效的凭证来验证您的请求，但是除非您拥有权限，否则您无法创建或访问 Amazon A EC2 uto Scaling 资源。例如，您必须有权创建自动扩缩组、使用启动模板启动实例等。

以下各节详细介绍 IAM 管理员如何使用 IAM 通过控制谁可以执行 Amazon A EC2 uto Scaling 操作来帮助保护您的 Amazon A EC2 uto Scaling 资源。

我们建议您先阅读 Amazon EC2 主题。请参阅《亚马逊 EC2 用户指南》 EC2中的亚马逊身份和访问管理。阅读本节的主题后，您应该可以很好地了解亚马逊 EC2提供哪些访问控制权限，以及这些权限如何与您的 Amazon A EC2 uto Scaling 资源权限相适应。

主题

- Amazon A EC2 uto Scaling 如何与 IAM 配合使用
- Amazon A EC2 uto Scaling API 权限
- Amazon Amazon A EC2 uto Scaling 的托管策略
- Amazon A EC2 uto Scaling 的服务相关角色
- Amazon A EC2 uto Scaling 基于身份的策略示例
- 防止跨服务混淆代理
- 在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况
- 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色

Amazon A EC2 uto Scaling 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon A EC2 uto Scaling 的访问权限之前，请先了解哪些可用于 Amazon A EC2 uto Scaling 的 IAM 功能。

你可以在 Amazon A EC2 uto Scaling 中使用的 IAM 功能

IAM 特征	Amazon A EC2 uto Scaling 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是

IAM 特征	Amazon A EC2 uto Scaling 支持
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	部分
临时凭证	是
服务角色	是
服务相关角色	是

要全面了解 Amazon A EC2 uto Scaling 和其他功能如何 Amazon Web Services 服务 与大多数 IAM 功能配合使用 Amazon Web Services 服务 ，请在 IAM 用户指南中查看如何与 IAM 配合使用。

Ama EC2 zon Auto Scaling 的基于身份的政策

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Amazon A EC2 uto Scaling 中基于资源的政策

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时

Amazon Web Services 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amazon A EC2 uto Scaling 的政策行动

支持策略操作：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 Amazon A EC2 uto Scaling 操作列表，请参阅《服务授权参考》中的 Amazon A EC2 uto Scaling 定义的操作。

Amazon A EC2 uto Scaling 中的策略操作在操作前使用以下前缀：

```
autoscaling
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
  "autoscaling:action1",
  "autoscaling:action2"
]
```

您还可以使用通配符（*）指定多项操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "autoscaling:Describe*"
```

Amazon A EC2 uto Scaling 的策略资源

支持策略资源：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

您可以使用 ARNs 来识别 Auto Scaling 组和适用 IAM 策略的启动配置。

Auto Scaling 组具有以下 ARN。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

具有以下 ARN 的启动配置。

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

要使用 CreateAutoScalingGroup 操作指定自动扩缩组，您必须将 UUID 替换为如下例中所示的通配符（*）。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

要使用 CreateLaunchConfiguration 操作指定启动配置，您必须将 UUID 替换为如下例中所示的通配符（*）。

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:*:launchConfigurationName/lc-name"
```

有关 Amazon A EC2 uto Scaling 资源类型及其类型的更多信息 ARNs，请参阅《服务授权参考》中的 Amazon A EC2 uto Scaling 定义的资源。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 A mazon A EC2 uto Scaling 定义的操作。

> **Note**
>
> 有关用于控制 Auto Sc ARNs aling 群组访问权限的 IAM 策略的示例，请参阅控制可以删除哪些自动扩缩组。

并非所有 Amazon A EC2 uto Scaling 操作都支持资源级权限。对于不支持资源级别权限的操作，您必须将通配符（*）作为资源。

以下 Amazon A EC2 uto Scaling 操作不支持资源级权限。

- `DescribeAccountLimits`
- `DescribeAdjustmentTypes`
- `DescribeAutoScalingGroups`
- `DescribeAutoScalingInstances`
- `DescribeAutoScalingNotificationTypes`
- `DescribeInstanceRefreshes`
- `DescribeLaunchConfigurations`
- `DescribeLifecycleHooks`
- `DescribeLifecycleHookTypes`
- `DescribeLoadBalancers`
- `DescribeLoadBalancerTargetGroups`
- `DescribeMetricCollectionTypes`
- `DescribeNotificationConfigurations`
- `DescribePolicies`
- `DescribeScalingActivities`
- `DescribeScalingProcessTypes`
- `DescribeScheduledActions`
- `DescribeTags`
- `DescribeTerminationPolicyTypes`
- `DescribeWarmPool`

Amazon A EC2 uto Scaling 的策略条件密钥

支持特定于服务的策略条件键：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 Amazon 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 Amazon 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

Amazon 支持全局条件密钥和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

Amazon A EC2 uto Scaling 支持以下条件键，这些条件键可用于控制对支持操作的访问权限和强制配置 Auto Scaling 群组：

- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/key-name: tag-value`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`

以下条件键特定于创建启动配置请求：

- `autoscaling:ImageId`
- `autoscaling:InstanceType`

- autoscaling:MetadataHttpEndpoint
- autoscaling:MetadataHttpPutResponseHopLimit
- autoscaling:MetadataHttpTokens
- autoscaling:SpotPrice

Amazon A EC2 uto Scaling 还支持以下全局条件密钥，您可以使用这些条件密钥根据请求中的标签或 Auto Scaling 组中存在的标签来定义权限。有关更多信息，请参阅 为 Auto Scaling 组和实例添加标签。

- aws:RequestTag/*key-name*: *tag-value*
- aws:ResourceTag/*key-name*: *tag-value*
- aws:TagKeys: *[tag-key, ...]*

要了解您可以将条件密钥与哪 EC2 些 Amazon Auto Scaling API 操作一起使用，请参阅《服务授权参考》中的 Amazon A EC2 uto Scaling 定义的操作。有关 Amazon A EC2 uto Scaling 条件键的更多信息，请参阅 Amazon A EC2 uto Scaling 的条件密钥。

> **Note**
>
> 有关使用条件键控制对受支持操作的访问和强制执行自动扩缩组配置的 IAM policy 示例，请参阅以下资源：
>
> - 需要启动模板和版本号：此示例强制要求在创建或更新自动扩缩组时必须指定启动模板和该启动模板的版本号。
> - 控制可以创建的自动扩缩组的大小：此示例用于在使用特定标签创建或更新自动扩缩组时，对 MinSize 和 MaxSize 属性的可能值进行限制。
> - 控制可以删除哪些扩展策略：此示例强制要求只允许删除没有特定标签的自动扩缩组的扩缩策略。

ACLs 在 Amazon A EC2 uto Scaling 中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

ABAC 搭载 Amazon A EC2 uto Scaling

支持 ABAC（策略中的标签）：部分支持

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 Amazon，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 Amazon 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

ABAC 可用于支持标签的资源，但并非所有资源都支持标签。启动配置和扩缩策略不支持标签，但自动扩缩组支持标签。

有关更多信息，请参阅 为 Auto Scaling 组和实例添加标签。

在 Amazon A EC2 uto Scaling 中使用临时证书

支持临时凭证：是

当你使用临时证书登录时，有些 Amazon Web Services 服务 不起作用。有关更多信息，包括哪些 Amazon Web Services 服务 适用于临时证书，请参阅 IAM 用户指南中的Amazon Web Services 服务与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 Amazon Web Services Management Console 使用的是临时证书。例如，当您 Amazon 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的从用户切换到 IAM 角色（控制台）。

您可以使用 Amazon CLI 或 Amazon API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 Amazon。 Amazon 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon A EC2 uto Scaling 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。

在创建通知亚马逊 SNS 主题或 Amazon SQS 队列的生命周期挂钩时，必须指定一个角色以允许 Amazon A EC2 uto Scaling 代表您访问亚马逊 SNS 或亚马逊 SQS。使用 IAM 控制台为您的生命周期挂钩设置服务角色。控制台可帮助您使用托管式策略创建一个具有足够权限集的角色。有关更多信息，请参阅使用 Amazon SNS 接收通知 和使用 Amazon SQS 接收通知。

创建 Auto Scaling 组时，您可以选择传入服务角色，以允许 Amazon EC2 实例代表您访问其他 Amazon Web Services 服务 实例。Amazon EC2 实例的服务角色（也称为启动模板或启动配置的 Amazon EC2 实例配置文件）是一种特殊类型的服务角色，在 EC2 实例启动时分配给 Auto Scaling 组中的每个实例。您可以使用 IAM 控制台和 Amazon CLI 来创建或编辑此服务角色。有关更多信息，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 Amazon A EC2 uto Scaling 的功能。只有在 Amazon A EC2 uto Scaling 提供相关指导时才编辑服务角色。

Amazon A EC2 uto Scaling 的服务相关角色

支持服务相关角色：是

服务相关角色是一种链接到的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理 Amazon A EC2 uto Scaling 服务相关角色的详细信息，请参阅Amazon A EC2 uto Scaling 的服务相关角色。

Amazon A EC2 uto Scaling API 权限

您必须向用户授予调用他们所需的 Amazon A EC2 uto Scaling API 操作的权限，如中所述Amazon A EC2 uto Scaling 的政策行动。此外，对于某些 Amazon A EC2 uto Scaling 操作，您必须向用户授予从其他人调用特定操作的权限 Amazon APIs。

其他人所需的权限 Amazon APIs

除了 Amazon A EC2 uto Scaling API 权限外，用户还必须拥有其他 Amazon APIs 人的以下权限才能成功执行相关操作。

创建自动扩缩组 (autoscaling:CreateAutoScalingGroup)

- iam:CreateServiceLinkedRole：创建默认的服务相关角色（如果该角色尚不存在）。
- iam:PassRole— 在启动时将 IAM 角色传递给服务或 EC2 实例。在提供非默认服务相关角色、生命周期挂钩的 IAM 角色或指定实例配置文件（IAM 角色的容器）的启动模板时需要。
- ec2:RunInstances：在提供启动模板时启动实例。
- ec2:CreateTags：在提供带有标签规格的启动模板时，在启动时标记实例和卷。

创建生命周期挂钩 (autoscaling:PutLifecycleHook)

- iam:PassRole：将 IAM 角色传递给相应服务。提供 IAM 角色时需要。

附加 VPC Lattice 目标组 (autoscaling:AttachTrafficSources)

- vpc-lattice:RegisterTargets：自动向目标组注册实例。

分离 VPC Lattice 目标组 (autoscaling:DetachTrafficSources)

- vpc-lattice:DeregisterTargets：自动取消注册目标组中的实例。

创建启动配置（autoscaling:CreateLaunchConfiguration）

- ec2:DescribeImages
- ec2:DescribeInstances
- ec2:DescribeInstanceAttribute
- ec2:DescribeKeyPairs
- ec2:DescribeSecurityGroups
- ec2:DescribeSpotInstanceRequests
- ec2:DescribeVpcClassicLink
- iam:PassRole— 在启动时将 IAM 角色传递给 EC2实例。启动配置文件指定实例配置文件（IAM 角色的容器）时需要。

Amazon Amazon A EC2 uto Scaling 的托管策略

Amazon 托管策略是由创建和管理的独立策略 Amazon。 Amazon 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， Amazon 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 Amazon 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 Amazon 托管策略中定义的权限。如果 Amazon 更新 Amazon 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 Amazon 最有可能在启动新的 API 或现有服务可以使用新 Amazon Web Services 服务 的 API 操作时更新 Amazon 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略。

Amazon A EC2 uto Scaling 托管策略

您可以将以下托管策略附加到您的 Amazon Identity and Access Management (IAM) 身份（用户或角色）。每项策略都提供对 Amazon A EC2 uto Scaling 的全部或部分 API 操作的访问权限。

- AutoScalingConsoleFullAccess— 使用授予对 Amazon A EC2 uto Scaling 的完全访问权限 Amazon Web Services Management Console。此策略在您使用启动配置时有效，但在使用启动模板时不适用。
- AutoScalingConsoleReadOnlyAccess— 使用授予对 Amazon A EC2 uto Scaling 的只读访问权限 Amazon Web Services Management Console。此策略在您使用启动配置时有效，但在使用启动模板时不适用。
- AutoScalingFullAccess— 为需要从 Amazon CLI 或 SDKs获得完全的 Amazon A EC2 uto Scaling 访问权限但不能访问的 IAM 身份授予对 Amazon A EC2 uto Scaling 的完全 Amazon Web Services Management Console 访问权限。
- AutoScalingReadOnlyAccess— 为仅调用 Amazon CLI 或的 IAM 身份授予对 Amazon A EC2 uto Scaling 的只读访问权限 SDKs。

当您使用控制台中的启动模板时，您需要授予特定于启动模板的其他权限，在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况 中对此进行了讨论。Amazon A EC2 uto Scaling 控制台需要ec2操作权限，这样它才能显示有关启动模板和使用启动模板启动实例的信息。

AutoScalingServiceRolePolicy Amazon 托管策略

此策略附加到一个服务相关角色，该角色允许 Amazon A EC2 uto Scaling 代表您执行操作。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。

要查看此策略的权限，请参阅《Amazon 托管式策略参考》中的 AutoScalingServiceRolePolicy。

Amazon A EC2 uto Scaling 更新 Amazon 了托管策略

查看自该服务开始跟踪这些更改以来对 Amazon A EC2 uto Scaling Amazon 托管策略的更新的详细信息。要获取有关此页面变更的自动提醒，请在 Amazon A EC2 uto Scaling 文档历史记录页面上订阅 RSS 提要。

更改	描述	日期
Amazon A EC2 uto Scaling 为其服务相关角色添加了权限	该`AutoScalingServiceRolePolicy` 策略现在包括调用 Amazon Resource Groups ListGroupResources API 操作以获取属于指定资源组的资源的所有资源名称 (ARNs) 的权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。	2024 年 11 月 20 日
Amazon A EC2 uto Scaling 为其服务相关角色添加了权限	现在，该`AutoScalingServiceRolePolicy` 策略授予了调用 Amazon EC2 GetSecurityGroupsForVpcAPI 操作以获取 VPC 的所有安全组以改善验证的权限，并授予了 Amazon EC2 GetInstanceTypesFromInstanceRequirementsAPI 操作以获取有关哪些实例类型满足特定实例要求的信息的权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。	2024 年 2 月 29 日
Amazon A EC2 uto Scaling 为其服务相关角色添加了权限	`AutoScalingServiceRolePolicy` 策略现在授予服务访问与 VPC Lattice 集成所需的 API 操作的权限。	2022 年 12 月 6 日

更改	描述	日期
	• `GetTargetGroup` 和 `ListTargetGroup` 操作。检索 VPC Lattice 目标组相关信息所需。 • `RegisterTargets` 和 `DeregisterTargets` 操作。在 VPC Lattice 目标组中注册和取消注册实例所需。 • `ListTargets` 。 允许 Amazon A EC2 uto Scaling 检索注册到 VPC 莱迪思目标组的实例的运行状况信息。 有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。	
Amazon A EC2 uto Scaling 为其服务相关角色添加了权限	为了支持在创建启动模板时使用 Amazon Systems Manager 参数作为 AMI ID 的别名，该`AutoScalingServiceRolePolicy` 策略现在授予了调用 Amazon Systems Manager GetParametersAPI 操作的权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。	2022 年 3 月 28 日

更改	描述	日期
Amazon A EC2 uto Scaling 为其服务相关角色添加了权限	为支持预测式扩展，`AutoScalingServiceRolePolicy` 策略现在包含调用 CloudWatch GetMetricData API 操作的权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。	2021 年 5 月 19 日
Amazon A EC2 uto Scaling 开始跟踪变更	Amazon A EC2 uto Scaling 开始跟踪其 Amazon 托管策略的更改。	2021 年 5 月 19 日

Amazon A EC2 uto Scaling 的服务相关角色

Amazon A EC2 uto Scaling 使用服务相关角色来获得 Amazon Web Services 服务 代表您呼叫他人所需的权限。服务相关角色是一种独特的 IAM 角色，直接链接到。 Amazon Web Services 服务

服务相关角色提供了一种将权限委托给其他 Amazon Web Services 服务 的安全方式，因为只有相关服务才能担任服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。服务相关角色还允许通过 Amazon CloudTrail查看所有 API 调用。这有助于满足监控和审计要求，因为您可以跟踪 Amazon A EC2 uto Scaling 代表您执行的所有操作。有关更多信息，请参阅 使用记录 Amazon A EC2 uto Scaling API 调用 Amazon CloudTrail。

以下各节介绍如何创建和管理 Amazon A EC2 uto Scaling 服务相关角色。首先配置权限以允许 IAM 身份（如用户或角色）创建、编辑或删除服务相关角色。

内容

- 编辑服务相关角色
- 删除服务相关角色

概览

Amazon A EC2 uto Scaling 服务相关角色有两种类型：

- 您账户的默认服务相关角色，名为 AWSServiceRoleForAutoScaling。 除非您指定其他服务相关角色，否则此角色会自动分配给您的 Auto Scaling 组。
- 具有自定义后缀的服务相关角色，该后缀由您在创建角色时指定，例如，AWSServiceRoleForAutoScaling_*mysuffix*.

自定义后缀服务相关角色的权限与默认的服务相关角色的权限完全相同。在这两种情况下，如果某个 Auto Scaling 组仍在使用这些角色，您不能编辑也不能删除它们。唯一的区别是角色名称后缀。

在编辑 Amazon Key Management Service 密钥策略时，您可以指定任一角色，以允许使用您的客户托管密钥对由 Amazon A EC2 uto Scaling 启动的实例进行加密。但是，如果您计划对特定的客户托管密钥提供精细访问，应使用自定义后缀服务相关角色。使用自定义后缀服务相关角色可为您提供：

- 对客户托管密钥的更多控制
- 能够在您的 CloudTrail日志中跟踪哪个 Auto Scaling 群组进行了 API 调用

如果您创建并非所有用户都有权访问的客户托管密钥，请按照以下步骤操作，以允许使用自定义后缀服务相关角色：

1. 创建具有自定义后缀的服务相关角色。有关更多信息，请参阅 创建服务相关角色（手动）。
2. 向服务相关角色授予对客户托管密钥的访问权限。有关允许服务相关角色使用密钥的密钥策略的更多信息，请参阅 使用加密卷所需的 Amazon KMS 密钥策略。
3. 授予用户对您创建的服务相关角色的访问权限。有关创建 IAM policy 的更多信息，请参阅 控制可以传递哪个服务相关角色（使用 PassRole）。如果用户尝试指定服务相关角色而无权将该角色传递给服务，则会收到错误。

服务相关角色授予的权限

Amazon A EC2 uto Scaling 使用名为的服务相关角色 AWSServiceRoleForAutoScaling 或者您的自定义后缀服务相关角色。

服务相关角色仅信任以下服务来担任该角色：

- autoscaling.amazonaws.com

角色权限策略，AutoScalingServiceRolePolicy，允许 Amazon A EC2 uto Scaling 完成以下操作：

- ec2— 创建、描述、修改、启动/停止和终止 EC2 实例。
- iam— 将 IAM 角色传递给 EC2 实例，以便在实例上运行的应用程序可以访问该角色的临时证书。
- iam— 创建AWSServiceRoleForEC2竞价服务相关角色以允许 Amazon A EC2 uto Scaling 代表您启动竞价型实例。
- elasticloadbalancing：向 Elastic Load Balancing 注册和取消注册实例，并检查注册目标的运行状况。
- cloudwatch— 创建、描述、修改和删除扩展策略的 CloudWatch警报，并检索用于预测性扩展的指标。
- sns：在实例启动或终止时向 Amazon SNS 发布通知。
- events— 代表您创建、描述、更新和删除 EventBridge规则。
- ssm：在启动模板中使用 Systems Manager 参数作为 AMI ID 的别名时，从 Parameter Store 中读取参数。
- vpc-lattice：向 VPC Lattice 注册和取消注册实例，并检查注册目标的运行状况。
- resource-groups— 获取属于指定资源组的资源的所有资源名称 (ARNs)。

Amazon A EC2 uto Scaling 服务相关角色支持的区域

Amazon A EC2 uto Scaling 支持在所有提供服务 Amazon Web Services 区域 的地方使用服务相关角色。

创建、编辑和删除服务相关角色

创建服务相关角色（自动）

Amazon A EC2 uto Scaling 创建了 AWSServiceRoleForAutoScaling 首次创建 Auto Scaling 组时将为您提供服务关联角色，除非您手动创建自定义后缀服务相关角色并在创建组时指定该角色。

您必须具备创建服务相关角色的 IAM 权限。否则，自动创建操作将失败。有关更多信息，请参阅 IAM 用户指南和本指南中的 创建服务相关角色 的服务相关角色权限。

创建服务相关角色（手动）

创建服务相关角色（控制台）

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Roles（角色）和 Create role（创建角色）。
3. 对于选择可信实体，选择 Amazon 服务。
4. 在 “选择将使用此角色的服务” 中，选择 EC2 Auto Scaling 和 A EC2 uto Scaling 用例。
5. 依次选择 Next: Permissions (下一步: 权限)、Next: Tags (下一步: 标签) 和 Next: Review (下一步: 审核)。注意：您无法在创建过程中将标签附加到服务相关角色。
6. 在 “审阅” 页面上，将角色名称留空以创建名称为的服务相关角色 AWSServiceRoleForAutoScaling，或者输入后缀以创建名为的服务相关角色 AWSServiceRoleForAutoScaling_***suffix***.
7. （可选）对于角色描述，编辑服务相关角色的描述。
8. 选择 Create role（创建角色）。

创建服务相关角色 (Amazon CLI)

使用以下 create-service-linked-roleCLI 命令为 Amazon A EC2 uto Scaling 创建名为的服务相关角色 AWSServiceRoleForAutoScaling_***suffix***.

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --
custom-suffix suffix
```

此命令的输出包含服务相关角色的 ARN，您可以用来向服务相关角色提供对客户托管密钥的访问权限。

```
{
    "Role": {
        "RoleId": "ABCDEF0123456789ABCDEF",
        "CreateDate": "2018-08-30T21:59:18Z",
        "RoleName": "AWSServiceRoleForAutoScaling_suffix",
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",
        "Path": "/aws-service-role/autoscaling.amazonaws.com/",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
```

```
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Principal": {
                        "Service": [
                            "autoscaling.amazonaws.com"
                        ]
                    },
                    "Effect": "Allow"
                }
            ]
        }
    }
}
```

有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。

编辑服务相关角色

您无法编辑为 Amazon A EC2 uto Scaling 创建的服务相关角色。创建服务相关角色后，您将无法更改角色的名称或其权限。但是，您可以编辑角色的说明。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色描述。

删除服务相关角色

如果您不使用某个 Auto Scaling 组，我们建议您删除其服务相关角色。删除此角色会防止您拥有不使用或不主动监视和维护的实体。

只有在先删除相关资源后，才能删除服务相关角色。这样可以防止您无意中撤销 Amazon A EC2 uto Scaling 对您的资源的权限。如果某个服务相关角色与多个 Auto Scaling 组结合使用，则必须删除使用该服务相关角色的所有 Auto Scaling 组，然后才能删除该服务相关角色。有关更多信息，请参阅 删除 Auto Scaling 基础设施。

您可以使用 IAM 删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

如果你删除 AWSServiceRoleForAutoScaling 服务相关角色，Amazon A EC2 uto Scaling 会在您创建 Auto Scaling 组时再次创建该角色，并且不指定其他服务相关角色。

Amazon A EC2 uto Scaling 基于身份的策略示例

默认情况下，您中的全新用户 Amazon Web Services 账户 无权执行任何操作。IAM 管理员必须创建和分配 IAM 策略，以授予 IAM 身份（例如用户或角色）执行 Amazon A EC2 uto Scaling API 操作的权限。

要了解如何使用这些示例 JSON 策略文档创建 IAM policy，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

下面介绍权限策略示例。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup",
          "autoscaling:DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
      }
   },
   {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
   }]
}
```

此示例策略授予创建、更新和删除自动扩缩组的权限，但仅限于组使用标签 **purpose=testing** 时。由于 Describe 操作不支持资源级权限，因此，您必须在不带条件的单独语句中必须指定它们。要使用启动模板启动实例，用户还必须拥有 ec2:RunInstances 权限。有关更多信息，请参阅 在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况。

> **Note**
>
> 您可以创建自己的自定义 IAM 策略，以允许或拒绝 IAM 身份（用户或角色）执行 Amazon A EC2 uto Scaling 操作的权限。您可以将这些自定义策略附加到需要指定权限的 IAM 身份。以下示例显示了一些常见使用情形的权限。
>
> 某些 Amazon A EC2 uto Scaling API 操作允许您在策略中包含特定的 Auto Scaling 组，这些组可以通过该操作创建或修改。您可以通过指定单个 Auto Scaling 组来限制这些操作的目标资源 ARNs。但是，作为最佳做法，我们建议您使用基于标签的策略，以允许（或拒绝）对具有特定标签的 Auto Scaling 组执行操作。

示例

- 控制可以创建的自动扩缩组的大小
- 控制可以使用哪些标签键和标签值
- 控制可以删除哪些自动扩缩组
- 控制可以删除哪些扩展策略
- 控制对实例刷新操作的访问权限
- 创建服务相关角色
- 控制可以传递哪个服务相关角色（使用 PassRole）

控制可以创建的自动扩缩组的大小

以下策略授予创建和更新具有标签 **environment=development** 的所有自动扩缩组的权限，只要请求者指定的最小大小不小于 **1** 或最大大小不大于 **10**。尽可能使用标签来帮助您控制对账户中自动扩缩组的访问权限。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
```

```
            "StringEquals": { "autoscaling:ResourceTag/environment": "development" },
            "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
            "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
        }
    }]
}
```

或者，如果您不使用标签来控制对 Auto Scaling 群组的访问权限，则可以使用 ARNs 来标识 IAM 策略所适用的 Auto Scaling 群组。

Auto Scaling 组具有以下 ARN。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/my-asg"
```

您也可以 ARNs 通过将它们包含在列表中来指定多个。有关在Resource元素中指定 Amazon A EC2 uto Scaling 资源的更多信息，请参阅Amazon A EC2 uto Scaling 的策略资源。 ARNs

控制可以使用哪些标签键和标签值

您还可以在 IAM 策略中使用条件来控制可应用到自动扩缩组的标签键和标签值。要授予创建自动扩缩组或为该组添加标签（仅当请求者指定特定标签时）的权限，请使用 aws:RequestTag 条件键。要仅允许特定的标签键，请使用带 aws:TagKeys 修饰符的 ForAllValues 条件键。

以下策略需要请求者在请求中指定包含键 **environment** 的标签。**"?*"** 值强制对于标签键有某个值。要使用通配符，您必须使用 StringLike 条件运算符。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateOrUpdateTags"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": { "aws:RequestTag/environment": "?*" }
        }
    }]
}
```

以下策略指定请求者只能使用标签 **purpose=webserver** 和 **cost-center=cc123** 标记自动扩缩组，并且只允许 **purpose** 和 **cost-center** 标签（不能指定其他标签）。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "aws:RequestTag/purpose": "webserver",
              "aws:RequestTag/cost-center": "cc123"
          },
          "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
      }
   }]
}
```

以下策略需要请求者在请求中指定至少一个标签，并且仅允许 **cost-center** 和 **owner** 键。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
      }
   }]
}
```

> **Note**
>
> 对于条件，条件键不区分大小写，条件值区分大小写。因此，要强制标签键区分大小写，请使用 `aws:TagKeys` 条件键，其中标签键指定为条件中的值。

控制可以删除哪些自动扩缩组

以下策略仅允许在自动扩缩组具有标签时删除该组**environment=development**。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "aws:ResourceTag/environment": "development" }
      }
   }]
}
```

或者，如果您不使用条件键来控制对 Auto Scaling 组 ARNs 的访问，则可以改为指定`Resource`元素中的资源来控制访问权限。

以下策略向用户授予使用 `DeleteAutoScalingGroup` API 操作的权限，但仅适用于名称以 **devteam-** 开头的自动扩缩组。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*"
   }]
}
```

您也可以 ARNs 通过将它们包含在列表中来指定多个。包括 UUID 可确保将访问权授予特定的 Auto Scaling 组。新组的 UUID 与删除的同名组的 UUID 不同。

```
"Resource": [
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

控制可以删除哪些扩展策略

以下策略授权使用 DeletePolicy 操作删除扩展策略。但是，如果对其执行操作的 Auto Scaling 组具有 **environment=production** 标签，此策略也会拒绝操作。尽可能使用标签来帮助您控制对账户中自动扩缩组的访问权限。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*"
   },
   {
      "Effect": "Deny",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
      }
   }]
}
```

控制对实例刷新操作的访问权限

仅当对其执行操作的自动扩缩组具有标签 **environment=testing** 时，以下策略授予启动、回滚和取消实例刷新的权限。由于 Describe 操作不支持资源级权限，因此，您必须在不带条件的单独语句中必须指定它们。

```
{
   "Version": "2012-10-17",
   "Statement": [{
```

```
        "Effect": "Allow",
        "Action": [
            "autoscaling:StartInstanceRefresh",
            "autoscaling:CancelInstanceRefresh",
            "autoscaling:RollbackInstanceRefresh"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/environment": "testing" }
        }
    },
    {
        "Effect": "Allow",
        "Action": "autoscaling:DescribeInstanceRefreshes",
        "Resource": "*"
    }]
}
```

要在 StartInstanceRefresh 调用中指定所需的配置，用户可能需要一些相关权限，例如：

- ec2: RunInstances — 要使用启动模板启动 EC2实例，用户必须拥有 IAM 策略中的ec2:RunInstances权限。有关更多信息，请参阅 在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况。
- ec2: CreateTags — 要使用在创建时向 EC2实例和卷添加标签的启动模板启动实例，用户必须拥有 IAM 策略中的ec2:CreateTags权限。有关更多信息，请参阅 标记实例和卷所需的权限。
- ia PassRole m: — 要从包含 EC2实例配置文件（IAM 角色的容器）的启动模板启动实例，用户还必须拥有 IAM 策略中的iam:PassRole权限。有关更多信息和示例 IAM policy，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。
- ssm: GetParameters — 要从使用 Amazon Systems Manager 参数的启动模板启动 EC2实例，用户还必须拥有 IAM 策略中的ssm:GetParameters权限。有关更多信息，请参阅 IDs 在启动模板中使用 Amazon Systems Manager 参数而不是 AMI。

创建服务相关角色

当你中的任何用户首次 Amazon Web Services 账户 调用 Amazon A EC2 uto Scaling API 操作时，Amazon A EC2 uto Scaling 需要权限才能创建服务相关角色。如果服务相关角色尚不存在，Amazon A EC2 uto Scaling 会在您的账户中创建该角色。服务相关角色向 Amazon A EC2 uto Scaling 授予权限，以便它可以 Amazon Web Services 服务 代表您呼叫其他人。

为使自动角色创建操作成功，用户必须具有 iam:CreateServiceLinkedRole 操作的权限。

```
"Action": "iam:CreateServiceLinkedRole"
```

以下显示了一个权限策略示例，该策略允许用户为 Amazon Auto Scaling 创建 Amazon A EC2 uto Scaling 服务相关角色。 EC2

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
      "Condition": {
          "StringLike": { "iam:AWSServiceName":"autoscaling.amazonaws.com" }
      }
   }]
}
```

控制可以传递哪个服务相关角色（使用 PassRole）

创建或更新自动扩缩组并在请求中指定自定义后缀服务相关角色的用户需要 iam:PassRole 权限。

如果您授予不同的服务相关角色访问不同密钥的iam:PassRole权限，则可以使用该权限来保护 Amazon KMS 客户托管密钥的安全。根据您组织的需求，您可能有三个密钥分别供开发团队、QA 团队和财务团队使用。首先，创建对所需密钥具有访问权限的服务相关角色，例如，名为 AWSServiceRoleForAutoScaling_devteamkeyaccess 的服务相关角色。然后，将策略附加到 IAM 身份，例如用户或角色。

以下策略授予权限，将 **AWSServiceRoleForAutoScaling_devteamkeyaccess** 角色传递给名称以 **devteam-** 开头的任意自动扩缩组。如果创建自动扩缩组的 IAM 身份尝试指定另一个与服务相关的角色，则会收到错误。如果用户选择不指定服务相关角色，则改为使用默认 AWSServiceRoleForAutoScaling 角色。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:PassRole",
```

```
      "Resource": "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_devteamkeyaccess",
      "Condition": {
          "StringEquals": { "iam:PassedToService": [ "autoscaling.amazonaws.com" ] },
          "StringLike": { "iam:AssociatedResourceARN":
 [ "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*" ] }
      }
   }]
}
```

有关自定义后缀服务相关角色的更多信息，请参阅 Amazon A EC2 uto Scaling 的服务相关角色。

防止跨服务混淆代理

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。

在中 Amazon，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。

为了防止这种情况，我们 Amazon 提供了一些工具，帮助您保护所有服务的数据，这些服务委托人已被授予对您账户中资源的访问权限。我们建议在 Amazon A EC2 uto Scaling 服务角色的信任策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文密钥。这些密钥限制了 Amazon A EC2 uto Scaling 向该资源提供的其他服务的权限。

SourceArn和SourceAccount字段的值是在 Amazon A EC2 uto Scaling 使用 Amazon Security Token Service (Amazon STS) 代表您担任角色时设置的。

要使用aws:SourceArn或aws:SourceAccount全局条件键，请将该值设置为 Amazon A EC2 uto Scaling 存储的亚马逊资源名称 (ARN) 或资源账户。请尽可能使用更具体的 aws:SourceArn。将值设置为 ARN 或带通配符 (*) 的 ARN 模式，用于 ARN 的未知部分。如果您不知道资源的 ARN，请改用 aws:SourceAccount。

以下示例显示了如何在 Amazon A EC2 uto Scaling 中使用aws:SourceArn和aws:SourceAccount全局条件上下文键来防止出现混淆的副手问题。

示例：使用 aws:SourceArn 和 aws:SourceAccount 条件键

由一项服务担任、代表您执行操作的角色称为服务角色。如果您想创建生命周期挂钩以向亚马逊以外的任何地方发送通知 EventBridge，则必须创建一个服务角色以允许 Amazon A EC2 uto Scaling 代表您

向亚马逊 SNS 主题或 Amazon SQS 队列发送通知。如果您只希望将一个自动扩缩组与跨服务访问相关联，则可以指定服务角色的信任策略，如下所示。

此示例信任策略使用条件语句，将服务角色的 AssumeRole 功能限制为只能执行影响指定账户中指定自动扩缩组的操作。aws:SourceArn 和 aws:SourceAccount 条件会得到独立评估。使用服务角色的任何请求都必须满足这两个条件。

在使用此策略之前，请将区域、账户 ID、UUID 和组名称替换为您账户中的有效值。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "autoscaling.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn":
 "arn:aws:autoscaling:region:account_id:autoScalingGroup:uuid:autoScalingGroupName/my-
asg"
      },
      "StringEquals": {
        "aws:SourceAccount": "account_id"
      }
    }
  }
}
```

在上述示例中：

- Principal 元素指定服务的服务主体 (autoscaling.amazonaws.com)。
- Action 元素指定 sts:AssumeRole 操作。
- Condition 元素指定 aws:SourceArn 和 aws:SourceAccount 全局条件键。源的 ARN 包含账户 ID，因此不必将 aws:SourceAccount 与 aws:SourceArn 结合使用。

其他信息

有关更多信息，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥、混乱的代理问题和更新角色信任策略。

在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况

Amazon A EC2 uto Scaling 支持在您的 Auto Scaling 群组中使用亚马逊 EC2 启动模板。我们建议您允许用户通过启动模板创建 Auto Scaling 群组，因为这样做可以让他们使用 Amazon A EC2 uto Scaling 和亚马逊的最新功能 EC2。例如，用户必须指定启动模板才能使用混合实例策略。

您可以使用该AmazonEC2FullAccess政策向用户授予使用其账户中的 Amazon A EC2 uto Scaling 资源、启动模板和其他 EC2 资源的完全访问权限。或者，您可以创建自己的自定义 IAM policy，为用户授予使用启动模板的精细访问权限，如本主题所述。

您可以为自己使用量身定制的示例策略

下面显示您可以为自身使用量身定制的基础权限策略示例。此策略授予创建、更新和删除所有自动扩缩组的权限，但仅限于组使用标签 **purpose=testing** 时。然后，它授予所有 Describe 操作的权限。由于 Describe 操作不支持资源级权限，因此，您必须在不带条件的单独语句中必须指定它们。

具有此策略的 IAM 身份（用户或角色）有权使用启动模板创建或更新自动扩缩组，因为他们还有权使用 ec2:RunInstances 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "autoscaling:Describe*",
                "ec2:RunInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

创建或更新自动扩缩组的用户可能需要一些相关权限，例如：

- ec2: CreateTags — 要在创建时向实例和卷添加标签，用户必须拥有 IAM 策略中的`ec2:CreateTags`权限。有关更多信息，请参阅 标记实例和卷所需的权限。
- ia PassRole m: — 要从包含 EC2实例配置文件（IAM 角色的容器）的启动模板启动实例，用户还必须拥有 IAM 策略中的`iam:PassRole`权限。有关更多信息和示例 IAM policy，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。
- ssm: GetParameters — 要从使用 Amazon Systems Manager 参数的启动模板启动 EC2实例，用户还必须拥有 IAM 策略中的`ssm:GetParameters`权限。有关更多信息，请参阅 IDs 在启动模板中使用 Amazon Systems Manager 参数而不是 AMI。

当用户与自动扩缩组交互时，会检查启动实例时要完成的操作的这些权限。有关更多信息，请参阅 `ec2:RunInstances 和 iam:PassRole 的权限验证`。

以下示例显示了您可用于控制 IAM 用户使用启动模板时具有的权限的策略语句。

主题

需要具有特定标签的启动模板

在授予ec2:RunInstances权限时，您可以指定用户在启动带有启动模板的实例时只能使用带有特定标签的启动模板或特定于 IDs 限制权限的启动模板。您还可以通过指定 RunInstances 调用的其他资源级权限，控制 AMI 和使用启动模板的任何人都可以在启动实例时引用和使用的其他资源。

以下示例限制了针对 ec2:RunInstances 操作的权限，该操作可用于启动位于指定区域中且具有标签 **purpose=testing** 的模板。它还允许用户访问启动模板中指定的资源：实例类型 AMIs、卷、密钥对、网络接口和安全组。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:launch-template/*",
            "Condition": {
                "StringEquals": { "aws:ResourceTag/purpose": "testing" }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region::image/ami-*",
                "arn:aws:ec2:region:account-id:instance/*",
                "arn:aws:ec2:region:account-id:subnet/*",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/*"
            ]
        }
    ]
}
```

有关在启动模板中使用基于标签的策略的更多信息，请参阅 A mazon EC2 用户指南中的使用 IAM 权限控制启动模板的访问权限。

需要启动模板和版本号

您还可以使用 IAM 权限强制要求在创建或更新自动扩缩组时必须指定启动模板和启动模板的版本号。

以下示例仅在指定启动模板和启动模板版本号时才允许用户创建和更新自动扩缩组。如果具有此策略的用户忽略版本号以指定 $Latest 或 $Default 启动模板版本，或者改为尝试使用启动配置，操作将失败。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": { "autoscaling:LaunchConfigurationName": "false" }
            }
        }
    ]
}
```

需要使用实例元数据服务版本 2 (IMDSv2)

为了提高安全性，您可以将用户的权限设置为要求使用所需的启动模板 IMDSv2。有关更多信息，请参阅 Amazon EC2 用户指南中的配置实例元数据服务。

以下示例指定，除非该实例也被选中要求使用 IMDSv2（由"ec2:MetadataHttpTokens":"required"），否则用户无法调用ec2:RunInstances操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
            }
        }
    ]
}
```

> **Tip**
>
> 要强制替换 Auto Scaling 使用新启动模板或配置了实例元数据选项的启动模板的新版本启动模板启动，您可以启动实例刷新。有关更多信息，请参阅 更新自动扩缩实例。

限制对 Amazon EC2 资源的访问

以下示例说明如何使用资源级权限和带标签的资源来限制对 Ama EC2 zon 资源的访问。

示例 1：将 Amazon EC2 实例的启动限制为特定资源和实例类型

以下示例通过限制对 Amazon EC2 资源的访问来控制用户可以启动的实例的配置。要为启动模板中指定的资源指定资源级权限，必须在 RunInstances 操作语句中包含这些资源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region:account-id:launch-template/*",
```

```
                "arn:aws:ec2:region::image/ami-04d5cc9b88example",
                "arn:aws:ec2:region:account-id:subnet/subnet-1a2b3c4d",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/sg-903004f88example"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": { "ec2:InstanceType": ["t2.micro", "t2.small"] }
            }
        }
    ]
}
```

在此示例中，有两个语句：

- 第一条语句要求用户使用特定安全组 (**subnet-1a2b3c4d**) 并使用特定 AMI (**sg-903004f88example**) 将实例启动到特定子网 (**ami-04d5cc9b88example**) 中。它还允许用户访问启动模板中指定的资源：网络接口、密钥对和卷。
- 第二个语句仅允许用户使用 **t2.micro** 和 **t2.small** 实例类型启动实例，您可以通过此操作控制成本。

  但请注意，目前尚无有效的方法可以完全阻止有权使用启动模板启动实例的用户启动其他实例类型。这是因为可以覆盖启动模板中指定的实例类型，以使用通过基于属性的实例类型选择进行定义的实例类型。

有关可用于控制用户可以启动的实例配置的资源级权限的完整列表，请参阅《服务授权参考》 EC2中的 Amazon 操作、资源和条件密钥。

示例 2：在 Amazon EC2 实例启动时需要标签并限制资源访问权限

以下示例策略展示了如何使用基于身份的策略中的条件根据标签控制对 Amazon EC2 资源的访问权限。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
         {
           "Sid": "InstanceTags",
           "Effect": "Allow",
           "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:555555555555:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/owner": "dev"
                }
            }
        },
        {
            "Sid": "InstanceBoundaries"
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1::image/*",
                "arn:aws:ec2:us-east-1:555555555555:subnet/*",
                "arn:aws:ec2:us-east-1:555555555555:network-interface/*"
            ],
        },
        {
            "Sid": "InstanceResourceTags",
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:555555555555:key-pair/*",
                "arn:aws:ec2:us-east-1:555555555555:security-group/*",
                "arn:aws:ec2:us-east-1:555555555555:launch-template/*"
                "arn:aws:ec2:us-east-1:555555555555:volume/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/purpose": "testing"
                }
```

```
            }
        }
    ]
}
```

在此示例中，有三个语句：

- 只有在请求中使用带owner=dev的标签时，第一条语句才允许用户在指定区域启动实例。
- 第二条语句允许用户访问指定区域中的 AMI、子网和网络接口。
- 第三条语句允许用户使用带有标签的现有密钥对、安全组、启动模板和卷在指定区域启动实例purpose=testing。

有关更多信息，请参阅 IAM 用户指南中的使用标签控制对 Amazon 资源的访问权限。

标记实例和卷所需的权限

以下示例允许用户在创建时标记实例和卷。如果在启动模板中指定了标签，则需要此策略。有关更多信息，请参阅 Amazon EC2 用户指南中的授予在创建期间为资源添加标签的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:region:account-id:*/*",
            "Condition": {
                "StringEquals": { "ec2:CreateAction": "RunInstances" }
            }
        }
    ]
}
```

其他启动模板权限

您必须向控制台用户授予对 ec2:DescribeLaunchTemplates 和 ec2:DescribeLaunchTemplateVersions 操作的权限。如果没有这些权限，则无法在 Auto Scaling 组向导中加载启动模板数据，并且用户无法逐步通过向导使用启动模板启动实例。您可以在 IAM policy 语句的 Action 元素中指定这些其他操作。

ec2:RunInstances 和 iam:PassRole 的权限验证

用户可以指定其自动扩缩组使用的启动模板版本。根据其权限，这可以是特定的编号版本，也可以是启动模板的 $Latest 或 $Default 版本。如果是后者，请特别小心。这可能会覆盖您打算限制的 ec2:RunInstances 和 iam:PassRole 的权限。

本节介绍在自动扩缩组中使用最新或默认版本的启动模板的场景。

当用户调用CreateAutoScalingGroupUpdateAutoScalingGroupStartInstanceRefresh APIs、或时，Amazon A EC2 uto Scaling 会根据当时最新或默认版本的启动模板版本检查其权限，然后再继续处理请求。这将验证启动实例时要完成的操作的权限，例如 ec2:RunInstances 和 iam:PassRole 操作。为此，我们发出 Amaz EC2 RunInstanceson 试运行调用，以验证用户是否具有执行该操作所需的权限，而无需实际提出请求。当返回响应时，Amazon A EC2 uto Scaling 会读取该响应。如果用户的权限不允许执行给定操作，Amazon A EC2 uto Scaling 会将请求失败，并向用户返回一个包含有关缺失权限信息的错误。

初始验证和请求完成后，每当实例启动时，Amazon A EC2 uto Scaling 都会使用其服务相关角色的权限启动最新或默认版本，即使实例已更改。这意味着即使启动模板的使用用户没有 iam:PassRole 权限，也可以更新启动模板以将 IAM 角色传递给实例。

如果您想限制谁有权访问配置群组以使用 $Latest 或 $Default 版本，请使用 autoscaling:LaunchTemplateVersionSpecified 条件键。这样可以确保 Auto Scaling 组仅在用户调用CreateAutoScalingGroup和时接受特定的编号版本UpdateAutoScalingGroup APIs。有关展示如何将此条件密钥添加到 IAM policy 的示例，请参阅 需要启动模板和版本号。

对于配置为使用$Latest或$Default启动模板版本的自动扩缩组，请考虑限制谁可以创建和管理启动模板的版本，包括允许用户指定默认启动模板版本的ec2:ModifyLaunchTemplate操作。有关更多信息，请参阅 Amazon EC2 用户指南中的控制版本控制权限。

相关资源

要详细了解查看、创建和删除启动模板以及启动模板版本的权限，请参阅 A mazon EC2 用户指南中的使用 IAM 权限控制启动模板的访问权限。

有关可用于控制RunInstances呼叫访问权限的资源级权限的更多信息，请参阅《服务授权参考》EC2中的 Amazon 操作、资源和条件密钥。

适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色

在 Amazon EC2 实例上运行的应用程序需要凭证才能访问其他 Amazon Web Services 服务。要以安全的方式提供这些凭证，请使用 IAM 角色。角色可提供临时权限，以供应用程序在访问其他 Amazon 资源时使用。角色的权限将确定允许访问资源的应用程序。

对于 Auto Scaling 组中的实例，您必须创建启动模板或启动配置，并选择要与实例关联的实例配置文件。实例配置文件是 IAM 角色的容器，它允许 Amazon EC2 在实例启动时将 IAM 角色传递给该实例。首先，创建具有访问 Amazon 资源所需的所有权限的 IAM 角色。然后，创建实例配置文件并将该角色分配给它。

> **Note**
>
> 作为最佳实践，我们强烈建议您创建该角色，使其拥有应用程序所需的最低其他 Amazon Web Services 服务 权限。

内容

前提条件

创建您在 Amazon 上运行的应用程序 EC2 可以代入的 IAM 角色。选择适当的权限，以便随后向应用程序提供可以进行所需 API 调用的角色。

如果您使用 IAM 控制台而不是其中一个 Amazon SDKs， Amazon CLI 则控制台会自动创建实例配置文件，并为其指定与其对应的角色相同的名称。

创建 IAM 角色（控制台）

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧的导航窗格中，选择角色。
3. 选择 Create role（创建角色）。
4. 对于选择可信实体，选择 Amazon 服务。

5. 对于您的用例，请选择，EC2然后选择 “下一步”。
6. 如果可能，选择要用作权限策略的策略，或选择 Create policy（创建策略）以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅《IAM 用户指南》中的创建 IAM 策略。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望服务具有的权限策略旁边的复选框。
7. （可选）设置权限边界。这是一项可用于服务角色的高级功能。有关更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
8. 选择下一步。
9. 在 Name, review, and create（命名、检查并创建）页面上，对于 Role name（角色名称），请输入一个角色名称以帮助标识此角色的作用。此名称在您的 Amazon Web Services 账户中必须唯一。由于其他 Amazon 资源可能会引用该角色，因此您无法在角色创建后对其名称进行编辑。
10. 检查该角色，然后选择创建角色。

IAM 权限

使用基于 IAM 身份的策略来控制对新 IAM 角色的访问。使用指定实例配置文件的启动模板创建或更新自动扩缩组的 IAM 身份（用户或角色）将需要 `iam:PassRole` 权限。

下面的策略示例授予仅传递名称以 **`qateam-`** 开头的 IAM 角色的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/qateam-*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        }
    ]
}
```

> **Important**
>
> 有关 Amazon A EC2 uto Scaling 如何验证使用启动模板的 Auto Scaling 组的`iam:PassRole`操作权限的信息，请参阅ec2:RunInstances 和 iam:PassRole 的权限验证。

创建启动模板

使用创建启动模板时 Amazon Web Services Management Console，在高级详细信息部分，从 IAM 实例配置文件中选择角色。有关更多信息，请参阅 使用高级设置创建启动模板。

使用中的create-launch-template命令创建启动模板时 Amazon CLI，请指定您的 IAM 角色的实例配置文件名称，如以下示例所示。

```
aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --
version-description version1 \
--launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","IamInstanceProfile":
{"Name":"my-instance-profile"}}'
```

另请参阅

有关帮助您开始学习和使用 Amazon 的 IAM 角色的更多信息 EC2，请参阅：

- 亚马逊 EC2 用户指南 EC2中的亚马逊 IA M 角色
- 在 IAM 用户指南中@@ 使用实例配置文件和使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限

Amazon A EC2 uto Scaling 的合规性验证

要了解是否属于特定合规计划的范围，请参阅Amazon Web Services 服务 “” Amazon Web Services 服务 中的 “按合规计划划分的范围”，然后选择您感兴趣的合规计划。 Amazon Web Services 服务 有关一般信息，请参阅合规计划。

您可以使用下载第三方审计报告 Amazon Artifact。有关更多信息，请参阅中的 “下载报告” Amazon Artifact。

您在使用 Amazon Web Services 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 Amazon 提供了以下资源来帮助实现合规性：

- Security & Compliance：这些解决方案实施指南讨论了架构考虑因素，并提供了部署安全性和合规性功能的步骤。
- 合规资源 — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用Amazon Config 开发人员指南中的规则评估资源 — 该 Amazon Config 服务评估您的资源配置在多大程度上符合内部实践、行业指导方针和法规。
- Amazon Security Hub— 这 Amazon Web Services 服务 提供了您内部安全状态的全面视图 Amazon。Security Hub 通过安全控制措施评估您的 Amazon 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控制措施的列表，请参阅 Security Hub 控制措施参考。
- Amazon GuardDuty — 它通过监控您的 Amazon Web Services 账户环境中是否存在可疑和恶意活动，来 Amazon Web Services 服务 检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。

PCI DSS 合规性

Amazon A EC2 uto Scaling 支持商家或服务提供商处理、存储和传输信用卡数据，并且已通过验证符合支付卡行业 (PCI) 数据安全标准 (DSS)。有关 PCI DSS 的更多信息，包括如何申请 PCI Compliance Package 的副本，请参阅 Amazon PCI DSS 第 1 级。

Amazon A EC2 uto Scaling 和接口 VPC 终端节点

您可以通过将 Amazon A EC2 uto Scaling 配置为使用接口 VPC 终端节点来改善 VPC 的安全状况。接口终端节点由一项技术提供支持 Amazon PrivateLink，通过限制您的 VPC 和 Amazon A EC2 uto Scaling APIs 之间的所有网络流量进入网络，使您能够私下访问 Amazon A EC2 uto Scaling。Amazon 借助接口终端节点，您也不需要 Internet 网关、NAT 设备或虚拟专用网关。

您无需进行配置 Amazon PrivateLink，但建议您这样做。有关 Amazon PrivateLink 和 VPC 终端节点的更多信息，请参阅什么是 Amazon PrivateLink？ 在Amazon PrivateLink 指南中。

主题

- 创建接口 VPC 终端节点
- 创建 VPC 端点策略

创建接口 VPC 终端节点

使用以下服务名称为 Amazon A EC2 uto Scaling 创建终端节点：

```
com.amazonaws.region.autoscaling
```

有关更多信息，请参阅Amazon PrivateLink 指南中的使用接口 VPC 终端节点访问 Amazon 服务。

您无需更改任何 Amazon A EC2 uto Scaling 设置。Amazon A EC2 uto Scaling 使用 Amazon 服务终端节点或私有接口 VPC 终端节点调用其他服务，以使用哪个为准。

创建 VPC 端点策略

您可以将策略附加到您的 VPC 终端节点，以控制对 Amazon A EC2 uto Scaling API 的访问。该策略指定：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

以下示例显示了一个 VPC 终端节点策略，该策略拒绝所有人通过终端节点删除扩展策略的权限。示例策略还授予所有人执行所有其他操作的权限。

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "autoscaling:DeleteScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

有关更多信息，请参阅 Amazon PrivateLink 指南中的使用端点策略控制对 VPC 端点的访问权限。

将此服务与 Amazon SDK 配合使用

Amazon 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档
Amazon CLI
适用于 Java 的 Amazon SDK
适用于 JavaScript 的 Amazon SDK
适用于 .NET 的 Amazon SDK
适用于 PHP 的 Amazon SDK
Amazon Tools for PowerShell
适用于 Python (Boto3) 的 Amazon SDK
适用于 Ruby 的 Amazon SDK
适用于 SAP ABAP 的 Amazon SDK

有关特定于此服务的示例，请参阅使用 Auto Scaling 的代码示例 Amazon SDKs。

使用 Auto Scaling 的代码示例 Amazon SDKs

以下代码示例展示了如何将 Auto Scaling 与 Amazon 软件开发套件 (SDK) 一起使用。

基础知识是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务服务结合来完成特定任务的代码示例。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

Hello Auto Scaling

以下代码示例显示如何开始使用自动扩缩。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
```

```
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
```

```
            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考[DescribeAutoScalingGroups](#)中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
```

```
        });
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeAutoScalingGroups中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
```

```
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
```

```
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Rust 的Amazon SDK API 参考。

代码示例

- 使用 Auto Scaling 的基本示例 Amazon SDKs
  - Hello Auto Scaling
  - 使用 Amazon SDK 学习 Auto Scaling 的基础知识
  - 使用 Auto Scaling 的操作 Amazon SDKs
    - 将 AttachInstances 与 CLI 配合使用
    - AttachLoadBalancerTargetGroups与 Amazon SDK 或 CLI 配合使用
    - 将 AttachLoadBalancers 与 CLI 配合使用
    - 将 CompleteLifecycleAction 与 CLI 配合使用
    - CreateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
    - 将 CreateLaunchConfiguration 与 CLI 配合使用
    - 将 CreateOrUpdateTags 与 CLI 配合使用
    - DeleteAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
    - 将 DeleteLaunchConfiguration 与 CLI 配合使用
    - 将 DeleteLifecycleHook 与 CLI 配合使用
    - 将 DeleteNotificationConfiguration 与 CLI 配合使用
    - 将 DeletePolicy 与 CLI 配合使用
    - 将 DeleteScheduledAction 与 CLI 配合使用

- 将 DeleteTags 与 CLI 配合使用
- 将 DescribeAccountLimits 与 CLI 配合使用
- 将 DescribeAdjustmentTypes 与 CLI 配合使用
- DescribeAutoScalingGroups与 Amazon SDK 或 CLI 配合使用
- DescribeAutoScalingInstances与 Amazon SDK 或 CLI 配合使用
- 将 DescribeAutoScalingNotificationTypes 与 CLI 配合使用
- 将 DescribeLaunchConfigurations 与 CLI 配合使用
- 将 DescribeLifecycleHookTypes 与 CLI 配合使用
- 将 DescribeLifecycleHooks 与 CLI 配合使用
- 将 DescribeLoadBalancers 与 CLI 配合使用
- 将 DescribeMetricCollectionTypes 与 CLI 配合使用
- 将 DescribeNotificationConfigurations 与 CLI 配合使用
- 将 DescribePolicies 与 CLI 配合使用
- DescribeScalingActivities与 Amazon SDK 或 CLI 配合使用
- 将 DescribeScalingProcessTypes 与 CLI 配合使用
- 将 DescribeScheduledActions 与 CLI 配合使用
- 将 DescribeTags 与 CLI 配合使用
- 将 DescribeTerminationPolicyTypes 与 CLI 配合使用
- 将 DetachInstances 与 CLI 配合使用
- 将 DetachLoadBalancers 与 CLI 配合使用
- DisableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- EnableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- 将 EnterStandby 与 CLI 配合使用
- 将 ExecutePolicy 与 CLI 配合使用
- 将 ExitStandby 与 CLI 配合使用
- 将 PutLifecycleHook 与 CLI 配合使用
- 将 PutNotificationConfiguration 与 CLI 配合使用
- 将 PutScalingPolicy 与 CLI 配合使用
- 将 PutScheduledUpdateGroupAction 与 CLI 配合使用
- 将 RecordLifecycleActionHeartbeat 与 CLI 配合使用

    - 将 ResumeProcesses 与 CLI 配合使用
    - SetDesiredCapacity与 Amazon SDK 或 CLI 配合使用
    - 将 SetInstanceHealth 与 CLI 配合使用
    - 将 SetInstanceProtection 与 CLI 配合使用
    - 将 SuspendProcesses 与 CLI 配合使用
    - TerminateInstanceInAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
    - UpdateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
- 使用 Auto Scaling 的场景 Amazon SDKs
    - 使用 Amazon SDK 构建和管理弹性服务

使用 Auto Scaling 的基本示例 Amazon SDKs

以下代码示例展示了如何使用 Amazon A EC2 uto Scaling 的基础知识 Amazon SDKs。

示例

- Hello Auto Scaling
- 使用 Amazon SDK 学习 Auto Scaling 的基础知识
- 使用 Auto Scaling 的操作 Amazon SDKs
    - 将 AttachInstances 与 CLI 配合使用
    - AttachLoadBalancerTargetGroups与 Amazon SDK 或 CLI 配合使用
    - 将 AttachLoadBalancers 与 CLI 配合使用
    - 将 CompleteLifecycleAction 与 CLI 配合使用
    - CreateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
    - 将 CreateLaunchConfiguration 与 CLI 配合使用
    - 将 CreateOrUpdateTags 与 CLI 配合使用
    - DeleteAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
    - 将 DeleteLaunchConfiguration 与 CLI 配合使用
    - 将 DeleteLifecycleHook 与 CLI 配合使用
    - 将 DeleteNotificationConfiguration 与 CLI 配合使用
    - 将 DeletePolicy 与 CLI 配合使用
    - 将 DeleteScheduledAction 与 CLI 配合使用

- 将 DeleteTags 与 CLI 配合使用
- 将 DescribeAccountLimits 与 CLI 配合使用
- 将 DescribeAdjustmentTypes 与 CLI 配合使用
- DescribeAutoScalingGroups与 Amazon SDK 或 CLI 配合使用
- DescribeAutoScalingInstances与 Amazon SDK 或 CLI 配合使用
- 将 DescribeAutoScalingNotificationTypes 与 CLI 配合使用
- 将 DescribeLaunchConfigurations 与 CLI 配合使用
- 将 DescribeLifecycleHookTypes 与 CLI 配合使用
- 将 DescribeLifecycleHooks 与 CLI 配合使用
- 将 DescribeLoadBalancers 与 CLI 配合使用
- 将 DescribeMetricCollectionTypes 与 CLI 配合使用
- 将 DescribeNotificationConfigurations 与 CLI 配合使用
- 将 DescribePolicies 与 CLI 配合使用
- DescribeScalingActivities与 Amazon SDK 或 CLI 配合使用
- 将 DescribeScalingProcessTypes 与 CLI 配合使用
- 将 DescribeScheduledActions 与 CLI 配合使用
- 将 DescribeTags 与 CLI 配合使用
- 将 DescribeTerminationPolicyTypes 与 CLI 配合使用
- 将 DetachInstances 与 CLI 配合使用
- 将 DetachLoadBalancers 与 CLI 配合使用
- DisableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- EnableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- 将 EnterStandby 与 CLI 配合使用
- 将 ExecutePolicy 与 CLI 配合使用
- 将 ExitStandby 与 CLI 配合使用
- 将 PutLifecycleHook 与 CLI 配合使用
- 将 PutNotificationConfiguration 与 CLI 配合使用
- 将 PutScalingPolicy 与 CLI 配合使用
- 将 PutScheduledUpdateGroupAction 与 CLI 配合使用
- 将 RecordLifecycleActionHeartbeat 与 CLI 配合使用

- 将 ResumeProcesses 与 CLI 配合使用
- SetDesiredCapacity与 Amazon SDK 或 CLI 配合使用
- 将 SetInstanceHealth 与 CLI 配合使用
- 将 SetInstanceProtection 与 CLI 配合使用
- 将 SuspendProcesses 与 CLI 配合使用
- TerminateInstanceInAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
- UpdateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用

Hello Auto Scaling

以下代码示例显示如何开始使用自动扩缩。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
```

```
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)
```

```
# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                 # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
```

```
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
```

```
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考[DescribeAutoScalingGroups](#)中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 [Amazon 代码示例存储库](#)中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeAutoScalingGroups中的。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import boto3


def hello_autoscaling(autoscaling_client):
```

```
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
{group['AavailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SDK 学习 Auto Scaling 的基础知识

以下代码示例演示了如何：

- 使用启动模板和可用区创建一个 Amazon A EC2 uto Scaling 群组，并获取有关正在运行的实例的信息。
- 启用 Amazon CloudWatch 指标收集。
- 更新组的所需容量，并等待实例启动。
- 终止组中的实例。
- 列出为响应用户请求和容量变化而发生的扩缩活动。
- 获取 CloudWatch 指标的统计数据，然后清理资源。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
```

```
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
 host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
 host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();
```

```
        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when
deleting the
        // launch template at the end of the application.
        var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);

        // Confirm that the template was created by asking for a description of
it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones.First().ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
```

```
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);

        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
```

```
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups!.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups is not null)
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities is not null)
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
```

```
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        Console.WriteLine($"Metrics collected for {groupName}:");
        metrics.ForEach(metric =>
        {
            Console.Write($"Metric name: {metric.MetricName}\t");
            Console.WriteLine($"Namespace: {metric.Namespace}");
        });

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        Console.WriteLine("Details for the metrics collected:");
        dataPoints.ForEach(detail =>
        {
            Console.WriteLine(detail);
        });

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
```

```
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
 autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
```

```
    {
        Console.WriteLine("This code example performs the following
operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the
group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
```

```
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
```

```
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

定义场景调用以管理启动模板和指标的函数。这些函数包含 Auto Scaling EC2、Amazon 和 CloudWatch 操作。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }
```

```
    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };

        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
```

```
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
 response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
 response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
```

```
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>?> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groupList = new List<string>
            {
```

```
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        var groups = response.AutoScalingGroups;

        return groups;
    }


    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
 _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
 {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };
```

```
        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
```

```
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
 launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
```

```
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
```

```
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
    public async Task<List<Amazon.CloudWatch.Model.Metric>>
 GetCloudWatchMetricsAsync(string groupName)
    {
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);

        return response.Metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
```

```
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = ${groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);

        return response.Datapoints;
    }

}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection

- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {
```

```
        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." <<
std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
"
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
```

```
                "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following
Availability Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }
```

```
    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
```

```
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
              "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
              << "to replace the terminated EC2 instance."
              << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                            1,

static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
```

```
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
```

```
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
```

```
    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const
 Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
```

```
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." <<
std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
```

```
            const
 Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
 &instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
 &details: instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
```

```
                                            const
Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                            const Aws::EC2::EC2Client &ec2Client)
{
   bool result = true;

   // 14. Delete the Auto Scaling group.
   if (!groupName.empty() &&
       (askYesNoQuestion(
               Aws::String("Delete the Auto Scaling group '") + groupName +
               "'  (y/n)?"))) {
       {
           Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
           request.SetAutoScalingGroupName(groupName);
           request.SetMinSize(0);
           request.SetDesiredCapacity(0);

           Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                   autoScalingClient.UpdateAutoScalingGroup(request);

           if (outcome.IsSuccess()) {
               std::cout
                       << "The minimum size and desired capacity of the Auto
Scaling group "
                       << "was set to zero before terminating the instances."
                       << std::endl;
           }
           else {
               std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                         << outcome.GetError().GetMessage() << std::endl;
               result = false;
           }
       }

       Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
       if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                              autoScalingClient)) {
           if (!autoScalingGroups.empty()) {
               Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                       autoScalingGroups[0].GetInstances());
               for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                   request.SetInstanceId(instanceID);
```

```
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =

autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                    request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before
deleting the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
```

```
                        << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was
 deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
```

```
                                                const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) throws InterruptedException {
       final String usage = """

               Usage:
                   <groupName> <launchTemplateName> <vpcZoneId>

               Where:
                   groupName - The name of the Auto Scaling group.
                   launchTemplateName - The name of the launch template.\s
                   vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String groupName = args[0];
       String launchTemplateName = args[1];
       String vpcZoneId = args[2];
       AutoScalingClient autoScalingClient = AutoScalingClient.builder()
               .region(Region.US_EAST_1)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Create an Auto Scaling group named " + groupName);
       createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
       System.out.println(
               "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
       Thread.sleep(60000);
       System.out.println(DASHES);

       System.out.println(DASHES);
```

```
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
```

```
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
```

```
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
```

```
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient
```

```
.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
           List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
           for (AutoScalingInstanceDetails instance : instances) {
               System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .maxRecords(10)
                   .build();

           DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
           List<AutoScalingGroup> groups = response.autoScalingGroups();
           for (AutoScalingGroup group : groups) {
               System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           String instanceId = "";
           DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
        try {
            DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
```

```
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
 in the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
 vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)
```

```
    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of
the groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
```

```
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
```

```
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
```

```
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
```

```
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

```
suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
```

```
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
```

```
         * Step 0: Create an EC2 launch template that you'll use to create an
 Auto Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you
 created in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
```

```
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the
 group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups:
 {$limits['NumberOfAutoScalingGroups']}\n";
```

```
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
```

```
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that
 have occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
```

```
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']}
 recorded:\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
```

```
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- 有关 API 详细信息，请参阅《适用于 PHP 的 Amazon SDK API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances

- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
 None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
 instances."
```

```
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
```

```
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
```

```
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities),
group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)

    if use_metrics:
```

```
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
    print("Set minimum size to 0.")
```

```
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
 demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

定义场景调用以管理启动模板和指标的函数。这些函数封装了 Amazon EC2 和 CloudWatch 操作。

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
```

```
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
```

```
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID
%s.",
                template_name,
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:
```

```
self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
```

```
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
```

```
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)
```

```
def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
 list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
 else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
 instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
            f"Here are the details of the instance{'s' if len(instances) > 1 else
 ''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection

- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};
```

```
use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();
```

```
    // 1. Create an EC2 launch template that you'll use to create an auto scaling
group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
```

```
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
the group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
```

```
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
```

```
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
 seconds.
```

```
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
 a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
```

```
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
```

```
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
```

```
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);
```

```
        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
        //   * Recommended: InstanceType='t1.micro',
ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
```

```
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
```

```
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };

        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
```

```
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
```

```
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting:
{}",
                                    group.status().unwrap_or("unknown error")
                                )));
                                return Err(errors);
                            }
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing
after three rounds",
            )])
        }
    }
```

```
   pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
       let group = self
           .autoscaling
           .describe_auto_scaling_groups()
           .auto_scaling_group_names(self.auto_scaling_group_name.clone())
           .send()
           .await
           .map(|s| {
               s.auto_scaling_groups()
                   .iter()
                   .map(|s| {
                       format!(
                           "{}: {}",
                           s.auto_scaling_group_name().unwrap_or("Unknown"),
                           s.status().unwrap_or("Unknown")
                       )
                   })
                   .collect::<Vec<String>>()
           })
           .map_err(|e| {
               ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
           });

       let instances = self
           .list_instances()
           .await
           .map_err(|e| anyhow!("There was an error listing instances: {e}",));

       // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
       //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
       //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
       //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
       let activities = self
           .autoscaling
           .describe_scaling_activities()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .into_paginator()
           .items()
```

```
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
```

```
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
{
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
```

```
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
 instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
```

```
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
```

```
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(0)
            .desired_capacity(0)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
```

```
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup

- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Auto Scaling 的操作 Amazon SDKs

以下代码示例演示了如何使用执行单个 Auto Scaling 操作 Amazon SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Auto Scaling API，是必须在上下文中运行的大型程序的代码节选。您可以在使用 Auto Scaling 的场景 Amazon SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。如需完整列表，请参阅 Amazon A EC2 uto Scaling API 参考。

示例

- 将 AttachInstances 与 CLI 配合使用
- AttachLoadBalancerTargetGroups与 Amazon SDK 或 CLI 配合使用
- 将 AttachLoadBalancers 与 CLI 配合使用
- 将 CompleteLifecycleAction 与 CLI 配合使用
- CreateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
- 将 CreateLaunchConfiguration 与 CLI 配合使用
- 将 CreateOrUpdateTags 与 CLI 配合使用
- DeleteAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
- 将 DeleteLaunchConfiguration 与 CLI 配合使用
- 将 DeleteLifecycleHook 与 CLI 配合使用
- 将 DeleteNotificationConfiguration 与 CLI 配合使用

- 将 DeletePolicy 与 CLI 配合使用
- 将 DeleteScheduledAction 与 CLI 配合使用
- 将 DeleteTags 与 CLI 配合使用
- 将 DescribeAccountLimits 与 CLI 配合使用
- 将 DescribeAdjustmentTypes 与 CLI 配合使用
- DescribeAutoScalingGroups与 Amazon SDK 或 CLI 配合使用
- DescribeAutoScalingInstances与 Amazon SDK 或 CLI 配合使用
- 将 DescribeAutoScalingNotificationTypes 与 CLI 配合使用
- 将 DescribeLaunchConfigurations 与 CLI 配合使用
- 将 DescribeLifecycleHookTypes 与 CLI 配合使用
- 将 DescribeLifecycleHooks 与 CLI 配合使用
- 将 DescribeLoadBalancers 与 CLI 配合使用
- 将 DescribeMetricCollectionTypes 与 CLI 配合使用
- 将 DescribeNotificationConfigurations 与 CLI 配合使用
- 将 DescribePolicies 与 CLI 配合使用
- DescribeScalingActivities与 Amazon SDK 或 CLI 配合使用
- 将 DescribeScalingProcessTypes 与 CLI 配合使用
- 将 DescribeScheduledActions 与 CLI 配合使用
- 将 DescribeTags 与 CLI 配合使用
- 将 DescribeTerminationPolicyTypes 与 CLI 配合使用
- 将 DetachInstances 与 CLI 配合使用
- 将 DetachLoadBalancers 与 CLI 配合使用
- DisableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- EnableMetricsCollection与 Amazon SDK 或 CLI 配合使用
- 将 EnterStandby 与 CLI 配合使用
- 将 ExecutePolicy 与 CLI 配合使用
- 将 ExitStandby 与 CLI 配合使用
- 将 PutLifecycleHook 与 CLI 配合使用
- 将 PutNotificationConfiguration 与 CLI 配合使用
- 将 PutScalingPolicy 与 CLI 配合使用

- 将 PutScheduledUpdateGroupAction 与 CLI 配合使用
- 将 RecordLifecycleActionHeartbeat 与 CLI 配合使用
- 将 ResumeProcesses 与 CLI 配合使用
- SetDesiredCapacity与 Amazon SDK 或 CLI 配合使用
- 将 SetInstanceHealth 与 CLI 配合使用
- 将 SetInstanceProtection 与 CLI 配合使用
- 将 SuspendProcesses 与 CLI 配合使用
- TerminateInstanceInAutoScalingGroup与 Amazon SDK 或 CLI 配合使用
- UpdateAutoScalingGroup与 Amazon SDK 或 CLI 配合使用

将 **AttachInstances** 与 CLI 配合使用

以下代码示例演示如何使用 `AttachInstances`。

CLI

Amazon CLI

将实例附加到自动扩缩组

此示例将指定的实例附加到指定的自动扩缩组。

```
aws autoscaling attach-instances \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AttachInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的实例附加到指定的自动扩缩组。Auto Scaling 会自动增加自动扩缩组的所需容量。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考AttachInstances中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AttachLoadBalancerTargetGroups`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachLoadBalancerTargetGroups`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
```

```
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考AttachLoadBalancerTargetGroups中的。

CLI

Amazon CLI

将目标组附加到自动扩缩组

此示例将指定目标组附加到指定的自动扩缩组。

```
aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Elastic Load Balancing 和 Amazon A EC2 uto Scaling。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AttachLoadBalancerTargetGroups中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    const client = new AutoScalingClient({});
```

```
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考AttachLoadBalancerTargetGroups中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
```

```
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
```

```
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- 有关 API 的详细信息，请参阅适用AttachLoadBalancerTargetGroups于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AttachLoadBalancers** 与 CLI 配合使用

以下代码示例演示如何使用 AttachLoadBalancers。

CLI

Amazon CLI

将经典负载均衡器附加到自动扩缩组

此示例将指定的经典负载均衡器附加到指定的自动扩缩组。

```
aws autoscaling attach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Elastic Load Balancing 和 Amazon A EC2 uto Scaling。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AttachLoadBalancers中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的负载均衡器附加到指定的自动扩缩组。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考AttachLoadBalancers中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CompleteLifecycleAction** 与 CLI 配合使用

以下代码示例演示如何使用 CompleteLifecycleAction。

CLI

Amazon CLI

完成生命周期操作

此示例通知 Amazon A EC2 uto Scaling 指定的生命周期操作已完成，因此它可以完成实例的启动或终止。

```
aws autoscaling complete-lifecycle-action \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-result CONTINUE \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CompleteLifecycleAction中的。

PowerShell

用于 PowerShell

示例 1：此示例完成了指定的生命周期操作。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考CompleteLifecycleAction中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateAutoScalingGroup`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateAutoScalingGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
        {
            availabilityZone,
        };

    var request = new CreateAutoScalingGroupRequest
```

```
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };

        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考CreateAutoScalingGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
```

```
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考CreateAutoScalingGroup中的。

CLI

Amazon CLI

示例 1：创建自动扩缩组

以下 `create-auto-scaling-group` 示例在区域内多个可用区中的子网中创建自动扩缩组。实例以指定启动模板的默认版本启动。请注意，大多数其他设置都使用默认值，例如，终止策略和运行状况检查配置。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 A EC2 uto Scaling 群组。

示例 2：附加应用程序负载均衡器、网络负载均衡器或网关负载均衡器

此示例为支持预期流量的负载均衡器指定目标组的 ARN。运行状况检查类型指定 ELB，以便在 Elastic Load Balancing 报告实例运行状况不佳时，自动扩缩组将取代它。该命令还定义了以 600 秒为单位的运行状况检查宽限期。宽限期有助于防止新启动的实例过早终止。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/943f017f100becff \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Elastic Load Balancing 和 Amazon A EC2 uto Scaling。

示例 3：指定置放群组，并使用最新版本的启动模板

此示例将实例启动到单个可用区中的置放群组。这对于具有 HPC 工作负载的低延迟群组很有用。此示例还将指定群组的最小大小、最大大小和所需容量。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \
    --min-size 1 \
```

```
    --max-size 5 \
    --desired-capacity 3 \
    --placement-group my-placement-group \
    --vpc-zone-identifier "subnet-6194ea3b"
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Linux 实例 EC2 用户指南》中的置放群组。

示例 4：指定单个实例自动扩缩组，并使用特定版本的启动模板

此示例创建一个自动扩缩组，并将其最小和最大容量均设置为 1 以强制运行一个实例。该命令还指定了启动模板的 v1，其中指定了现有 ENI 的 ID。当您使用为 eth0 指定现有 ENI 的启动模板时，必须为自动扩缩组指定与网络接口匹配的可用区，而无需在请求中同时指定子网 ID。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
 \
    --min-size 1 \
    --max-size 1 \
    --availability-zones us-west-2a
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 A EC2 uto Scaling 群组。

示例 5：指定不同的终止策略

此示例使用启动配置创建自动扩缩组，并将终止策略设置为首先终止最旧的实例。该命令还将标签应用于该组及其实例，其密钥为 Role，值为 WebServer。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-configuration-name my-lc \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "OldestInstance" \
    --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用 Amazon A EC2 uto Scaling 终止政策。

示例 6：指定启动生命周期挂钩

此示例将使用生命周期挂钩创建一个自动扩缩组，该挂钩支持在实例启动时的自定义操作。

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

config.json 文件的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-launch-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-
queue",
        "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
        "NotificationMetadata": "SQS message metadata",
        "HeartbeatTimeout": 4800,
        "DefaultResult": "ABANDON"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": [{
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "test",
        "Key": "environment"
    }]
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

示例 7：指定终止生命周期挂钩

此示例将使用生命周期挂钩创建一个自动扩缩组，该挂钩支持在实例终止时的自定义操作。

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-termination-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "HeartbeatTimeout": 120,
        "DefaultResult": "CONTINUE"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067"
    ],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

示例 8：指定自定义终止策略

此示例创建了一个 Auto Scaling 组，该组指定了自定义 Lambda 函数终止策略，该策略告诉 Amazon A EC2 uto Scaling 哪些实例可以安全地在扩展时终止。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling \
    --min-size 1 \
```

```
    --max-size 5 \
    --termination-policies "arn:aws:lambda:us-
west-2:123456789012:function:HelloFunction:prod" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅 A mazon A EC2 uto Scaling 用户指南中的使用 Lambda 创建自定义终止策略。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateAutoScalingGroup中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
 where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
 autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
```

```
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
 CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateAutoScalingGroup中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
```

```
    }
}
```

- 有关 API 的详细信息，请参阅适用CreateAutoScalingGroup于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考CreateAutoScalingGroup中的。

PowerShell

用于 PowerShell

示例 1：此示例使用指定的名称和属性创建自动扩缩组。默认所需容量为最小大小。因此，此自动扩缩组启动两个实例，指定的两个可用区中各一个。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考CreateAutoScalingGroup中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
```

```
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the
configuration of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
exists. "
                    "Please use a different name or update the existing group.",
```

```
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch
 configurations. "
                    "Consider deleting unused resources or request a limit
 increase. "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- 有关 API 的详细信息，请参阅适用CreateAutoScalingGroup于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
 {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;
```

```
    println!("Created AutoScaling group");

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用CreateAutoScalingGroup于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `CreateLaunchConfiguration` 与 CLI 配合使用

以下代码示例演示如何使用 `CreateLaunchConfiguration`。

CLI

Amazon CLI

示例 1：创建启动配置

此示例创建一个简单的启动配置。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的创建启动配置。

示例 2：使用安全组、密钥对和引导脚本创建启动配置

此示例使用用户数据中包含的安全组、密钥对和引导脚本创建启动配置。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
```

```
    --security-groups sg-eb2af88example \
    --key-name my-key-pair \
    --user-data file://myuserdata.txt
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的创建启动配置。

示例 3：使用 IAM 角色创建启动配置

此示例使用 IAM 角色的实例配置文件名称创建启动配置。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --iam-instance-profile my-autoscaling-role
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的在亚马逊 EC2 实例上运行的应用程序的 IA M 角色。

示例 4：创建启用详细监控的启动配置

此示例创建了启用 EC2 详细监控的启动配置，该配置将 CloudWatch 在 1 分钟内向发送 EC2 指标。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --instance-monitoring Enabled=true
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的配置 EC2 自动伸缩实例的监控。

示例 5：创建启动竞价型实例的启动配置

此示例创建使用竞价型实例作为唯一购买选项的启动配置。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --spot-price "0.50"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的请求竞价型实例。

示例 6：使用 EC2 实例创建启动配置

此示例基于现有实例的属性创建启动配置。它通过包含 `--placement-tenancy` 和 `--no-associate-public-ip-address` 选项来覆盖置放租赁以及是否设置公有 IP 地址。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc-from-instance \
    --instance-id i-0123a456700123456 \
    --instance-type m5.large \
    --no-associate-public-ip-address \
    --placement-tenancy dedicated
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用 EC2 实例创建启动配置。

示例 7：为 Amazon EBS 卷创建具有块设备映射的启动配置

此示例为设备名称为 `/dev/sdh` 和卷大小为 20 的 Amazon EBS gp3 卷创建具有块设备映射的启动配置。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":
{"VolumeSize":20,"VolumeType":"gp3"}}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling API 参考中的 EBS。

有关引用 JSON 格式参数值的语法的信息，请参阅《Amazon 命令行界面用户指南》中的 CL Amazon I 中对字符串使用引号。

示例 8：为实例存储卷创建具有块设备映射的启动配置

此示例创建一个启动配置，将 ephemeral1 作为实例存储卷，设备名称为 /dev/sdc。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/
sdc","VirtualName":"ephemeral1"}]'
```

此命令不生成任何输出。

有关更多信息，请参阅BlockDeviceMapping《Amazon A EC2 uto Scaling API 参考》。

有关引用 JSON 格式参数值的语法的信息，请参阅《Amazon 命令行界面用户指南》中的 CL Amazon I 中对字符串使用引号。

示例 9：创建启动配置并禁止在启动时附加块设备

此示例创建一个启动配置，用于禁止通过 AMI 的块设备映射指定的块设备（例如，/dev/sdf）。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

此命令不生成任何输出。

有关更多信息，请参阅BlockDeviceMapping《Amazon A EC2 uto Scaling API 参考》。

有关引用 JSON 格式参数值的语法的信息，请参阅《Amazon 命令行界面用户指南》中的 CL Amazon I 中对字符串使用引号。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateLaunchConfiguration中的。

PowerShell

用于 PowerShell

示例 1：此示例创建一个名为“my-lc”的启动配置。使用此启动配置的 Auto Scaling 组启动的 EC2 实例使用指定的实例类型、AMI、安全组和 IAM 角色。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考CreateLaunchConfiguration中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateOrUpdateTags** 与 CLI 配合使用

以下代码示例演示如何使用 `CreateOrUpdateTags`。

CLI

Amazon CLI

创建或更新自动扩缩组的标签

此示例向指定的自动扩缩组添加两个标签。

```
aws autoscaling create-or-update-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-
asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 A EC2 uto Scaling 组和实例添加标签。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateOrUpdateTags中的。

PowerShell

用于 PowerShell

示例 1：此示例向指定的自动扩缩组添加单个标签。标签键为 “myTag”，标签值为 “myTagValue”。Auto Scaling 会将此标签传播到 Auto Scaling 组启动的后续 EC2 实例。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 为标签参数创建标签。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考CreateOrUpdateTags中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteAutoScalingGroup与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteAutoScalingGroup。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

将自动扩缩组的最小大小更新为零，终止该组中的所有实例，然后删除该组。

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
```

```
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };
```

```
        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DeleteAutoScalingGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Auto Scaling group '" << groupName << "' was
 deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DeleteAutoScalingGroup中的。

CLI

Amazon CLI

示例 1：删除指定的自动扩缩组

此示例将删除指定的自动扩缩组。

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的删除 EC2 自动扩展基础设施。

示例 2：强制删除指定的自动扩缩组

要在不等待自动扩缩组中的实例终止的情况下删除该组，请使用 `--force-delete` 选项。

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --force-delete
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的删除 EC2 自动扩展基础设施。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteAutoScalingGroup中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteAutoScalingGroup中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- 有关 API 的详细信息，请参阅适用DeleteAutoScalingGroup于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
```

```
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DeleteAutoScalingGroup中的。

PowerShell

用于 PowerShell

示例 1：如果指定的自动扩缩组没有正在运行的实例，则此示例将删除该组。在操作继续之前，系统会提示您进行确认。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

示例 3：此示例删除指定的自动扩缩组并终止该组包含的所有正在运行的实例。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteAutoScalingGroup中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

将自动扩缩组的最小大小更新为零，终止该组中的所有实例，然后删除该组。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
```

```
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- 有关 API 的详细信息，请参阅适用DeleteAutoScalingGroup于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
 Error> {
    client
```

```
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DeleteAutoScalingGroup于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteLaunchConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteLaunchConfiguration。

CLI

Amazon CLI

删除启动配置

此示例删除指定的启动配置。

```
aws autoscaling delete-launch-configuration \
    --launch-configuration-name my-launch-config
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的删除 EC2 自动扩展基础设施。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteLaunchConfiguration中的。

PowerShell

用于 PowerShell

示例 1：如果指定的启动配置未附加到自动扩缩组，则此示例将删除该配置。在操作继续之前，系统会提示您进行确认。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteLaunchConfiguration中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteLifecycleHook** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteLifecycleHook。

CLI

Amazon CLI

删除生命周期挂钩

此示例删除指定的生命周期挂钩。

```
aws autoscaling delete-lifecycle-hook \
```

```
    --lifecycle-hook-name my-lifecycle-hook \
    --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteLifecycleHook中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定自动扩缩组的指定生命周期挂钩。在操作继续之前，系统会提示您进行确认。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteLifecycleHook中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteNotificationConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteNotificationConfiguration。

CLI

Amazon CLI

删除 Auto Scaling 通知

此示例删除指定自动扩缩组的指定通知。

```
aws autoscaling delete-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的删除通知配置。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteNotificationConfiguration中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的通知操作。在操作继续之前，系统会提示您进行确认。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteNotificationConfiguration中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeletePolicy** 与 CLI 配合使用

以下代码示例演示如何使用 DeletePolicy。

CLI

Amazon CLI

删除扩缩策略

此示例删除指定的扩展策略。

```
aws autoscaling delete-policy \
    --auto-scaling-group-name my-asg \
    --policy-name alb1000-target-tracking-scaling-policy
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeletePolicy中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定自动扩缩组的指定策略。在操作继续之前，系统会提示您进行确认。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeletePolicy中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteScheduledAction** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteScheduledAction。

CLI

Amazon CLI

从自动扩缩组中删除计划的操作

此示例从指定的自动扩缩组删除指定的计划操作。

```
aws autoscaling delete-scheduled-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteScheduledAction中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定自动扩缩组的指定计划操作。在操作继续之前，系统会提示您进行确认。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteScheduledAction中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteTags** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteTags。

CLI

Amazon CLI

从自动扩缩组中删除标签

此示例从指定的自动扩缩组删除指定的标签。

```
aws autoscaling delete-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 A EC2 uto Scaling 组和实例添加标签。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteTags中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定的自动扩缩组删除指定的标签。在操作继续之前，系统会提示您进行确认。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

示例 3：对于 Powershell 版本 2，必须使用 New-Object 创建 Tag 参数的标签。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DeleteTags中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeAccountLimits** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeAccountLimits。

CLI

Amazon CLI

描述您的 Amazon A EC2 uto Scaling 账户限制

此示例描述了您 Amazon 账户的 Amazon A EC2 uto Scaling 限制。

```
aws autoscaling describe-account-limits
```

输出：

```
{
    "NumberOfLaunchConfigurations": 5,
    "MaxNumberOfLaunchConfigurations": 100,
    "NumberOfAutoScalingGroups": 3,
    "MaxNumberOfAutoScalingGroups": 20
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 服务配额。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAccountLimits中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您的 Amazon 账户的 Auto Scaling 资源限制。

```
Get-ASAccountLimit
```

输出：

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeAccountLimits中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeAdjustmentTypes** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeAdjustmentTypes。

CLI

Amazon CLI

描述可用的扩缩调整类型

此示例描述可用的调整类型。

```
aws autoscaling describe-adjustment-types
```

输出：

```
{
    "AdjustmentTypes": [
        {
            "AdjustmentType": "ChangeInCapacity"
        },
        {
            "AdjustmentType": "ExactCapacity"
        },
        {
            "AdjustmentType": "PercentChangeInCapacity"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的缩放调整类型。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAdjustmentTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例描述 Auto Scaling 所支持的调整类型。

```
Get-ASAdjustmentType
```

输出：

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeAdjustmentTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAutoScalingGroups`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAutoScalingGroups`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

CLI

Amazon CLI

示例 1：描述指定的自动扩缩组

此示例将描述指定的自动扩缩组。

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-names my-asg
```

输出：

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-1234567890abcde12"
            },
            "MinSize": 0,
            "MaxSize": 1,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-06905f55584de02da",
                    "InstanceType": "t2.micro",
                    "AvailabilityZone": "us-west-2a",
```

```
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "ProtectedFromScaleIn": false,
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcde12"
                    }
                }
            ],
            "CreatedTime": "2023-10-28T02:39:22.152Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":"arn",
            "TrafficSources": []
        }
    ]
}
```

示例 2：描述前 100 个指定的自动扩缩组

此示例将描述指定的自动扩缩组。它允许您指定最多 100 个组名称。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 100 \
    --auto-scaling-group-names "group1" "group2" "group3" "group4"
```

有关输出示例，请参阅示例 1。

示例 3：描述指定区域中的自动扩缩组

此示例将描述指定区域中的自动扩缩组（最多 75 个组）。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 75 \
    --region us-east-1
```

有关输出示例，请参阅示例 1。

示例 4：描述指定数量的自动扩缩组

要返回特定数量的自动扩缩组，请使用 `--max-items` 选项。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 1
```

有关输出示例，请参阅示例 1。

如果输出包含 `NextToken` 字段，则可描述更多组。要获取其他组，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-auto-scaling-groups \
    --starting-token Z3M3LMPEXAMPLE
```

有关输出示例，请参阅示例 1。

示例 5：描述使用启动配置的自动扩缩组

此示例使用 `--query` 选项描述使用启动配置的自动扩缩组。

```
aws autoscaling describe-auto-scaling-groups \
    --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

输出：

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
        "LaunchConfigurationName": "my-lc",
        "MinSize": 0,
        "MaxSize": 1,
        "DesiredCapacity": 1,
        "DefaultCooldown": 300,
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
```

```
        ],
        "LoadBalancerNames": [],
        "TargetGroupARNs": [],
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": 0,
        "Instances": [
            {
                "InstanceId": "i-088c57934a6449037",
                "InstanceType": "t2.micro",
                "AvailabilityZone": "us-west-2c",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc",
                "ProtectedFromScaleIn": false
            }
        ],
        "CreatedTime": "2023-10-28T02:39:22.152Z",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN":"arn",
        "TrafficSources": []
    }
]
```

有关更多信息，请参阅《Amazon 命令行界面用户指南》中的筛选 Amazon CLI 输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAutoScalingGroups中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
```

```
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
 group.autoScalingGroupName());
                System.out.println("The group ARN is " +
 group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考[DescribeAutoScalingGroups](#)中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DescribeAutoScalingGroups中的。

PowerShell

用于 PowerShell

示例 1：此示例列出自动扩缩组的名称。

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

输出：

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

示例 2：此示例描述指定的自动扩缩组。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

输出：

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

示例 3：此示例描述指定的两个自动扩缩组。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

示例 4：此示例描述指定自动扩缩组的 Auto Scaling 实例。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

示例 5：此示例描述所有自动扩缩组。

```
Get-ASAutoScalingGroup
```

示例 6：此示例描述了指定 LaunchTemplate 的 Auto Scaling 组。此示例假设“实例购买选项”设置为“遵照启动模板”。如果此选项设置为 “合并购买选项和实例类型”，则 LaunchTemplate 可以使用 “进行访问MixedInstancesPolicy。 LaunchTemplate“财产。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

输出：

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeAutoScalingGroups中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.
```

```
        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingGroups于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAutoScalingInstances`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAutoScalingInstances`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
```

```
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
_amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考[DescribeAutoScalingInstances](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 [Amazon 代码示例存储库](#)中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);
```

```
        if (outcome.IsSuccess()) {
            const
 Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
 &instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DescribeAutoScalingInstances中的。

CLI

Amazon CLI

示例 1：描述一个或多个实例

此示例将描述指定的实例。

```
aws autoscaling describe-auto-scaling-instances \
    --instance-ids i-06905f55584de02da
```

输出：

```
{
    "AutoScalingInstances": [
        {
            "InstanceId": "i-06905f55584de02da",
            "InstanceType": "t2.micro",
            "AutoScalingGroupName": "my-asg",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "HEALTHY",
            "ProtectedFromScaleIn": false,
```

```
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-1234567890abcde12",
                "LaunchTemplateName": "my-launch-template",
                "Version": "1"
            }
        }
    ]
}
```

示例 2：描述一个或多个实例

此示例使用 `--max-items` 选项来指定通过此调用返回多少个实例。

```
aws autoscaling describe-auto-scaling-instances \
    --max-items 1
```

如果输出包含 NextToken 字段，可返回更多实例。要获取其他实例，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-auto-scaling-instances \
    --starting-token Z3M3LMPEXAMPLE
```

有关输出示例，请参阅示例 1。

示例 3：描述使用启动配置的实例

此示例使用 `--query` 选项描述使用启动配置的实例。

```
aws autoscaling describe-auto-scaling-instances \
    --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

输出：

```
[
    {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AutoScalingGroupName": "my-asg",
        "AvailabilityZone": "us-west-2c",
        "LifecycleState": "InService",
        "HealthStatus": "HEALTHY",
        "LaunchConfigurationName": "my-lc",
```

```
        "ProtectedFromScaleIn": false
    }
]
```

有关更多信息，请参阅《Amazon 命令行界面用户指南》中的筛选 Amazon CLI 输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAutoScalingInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
                   .build();

           DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
           List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
           for (AutoScalingInstanceDetails instance : instances) {
               System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeAutoScalingInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingInstances于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DescribeAutoScalingInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例列出了您的 IDs Auto Scaling 实例。

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

输出：

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

示例 2：此示例描述指定的 Auto Scaling 实例。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

输出：

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

示例 3：此示例描述指定的两个 Auto Scaling 实例。

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

示例 4：此示例描述指定自动扩缩组的 Auto Scaling 实例。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

示例 5：此示例描述所有 Auto Scaling 实例。

```
Get-ASAutoScalingInstance
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeAutoScalingInstances中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingInstances于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
 instances", &err))
    }
```

- 有关 API 的详细信息，请参阅适用DescribeAutoScalingInstances于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `DescribeAutoScalingNotificationTypes` 与 CLI 配合使用

以下代码示例演示如何使用 `DescribeAutoScalingNotificationTypes`。

CLI

Amazon CLI

描述可用的通知类型

此示例描述可用的通知类型。

```
aws autoscaling describe-auto-scaling-notification-types
```

输出：

```
{
    "AutoScalingNotificationTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCH",
        "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
        "autoscaling:EC2_INSTANCE_TERMINATE",
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
        "autoscaling:TEST_NOTIFICATION"
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Auto Scaling 群组缩放时获取 Amazon EC2 SNS 通知。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAutoScalingNotificationTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例列出 Auto Scaling 支持的通知类型。

```
Get-ASAutoScalingNotificationType
```

输出：

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeAutoScalingNotificationTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLaunchConfigurations** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeLaunchConfigurations。

CLI

Amazon CLI

示例 1：描述指定的启动配置

此示例描述指定的启动配置。

```
aws autoscaling describe-launch-configurations \
    --launch-configuration-names my-launch-config
```

输出：

```
{
    "LaunchConfigurations": [
        {
```

```
            "LaunchConfigurationName": "my-launch-config",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
            "ImageId": "ami-0528a5175983e7f28",
            "KeyName": "my-key-pair-uswest2",
            "SecurityGroups": [
                "sg-05eaec502fcdadc2e"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "",
            "InstanceType": "t2.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "SnapshotId": "snap-06c1606ba5ca274b1",
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "DeleteOnTermination": true,
                        "Encrypted": false
                    }
                }
            ],
            "InstanceMonitoring": {
                "Enabled": true
            },
            "CreatedTime": "2020-10-28T02:39:22.321Z",
            "EbsOptimized": false,
            "AssociatePublicIpAddress": true,
            "MetadataOptions": {
                "HttpTokens": "required",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "disabled"
            }
        }
    ]
}
```

示例 2：描述指定数量的启动配置

要返回特定数量的启动配置，请使用 `--max-items` 选项。

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

如果输出包含 NextToken 字段，则可返回更多启动配置。要获取其他启动配置，请在后续调用中使用此字段的值和 --starting-token 选项，如下所示。

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeLaunchConfigurations中的。

PowerShell

用于 PowerShell

示例 1：此示例列出启动配置的名称。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

输出：

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

示例 2：此示例描述指定的启动配置。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

输出：

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
```

```
CreatedTime                 : 12/12/2014 3:22:08 PM
EbsOptimized                : False
IamInstanceProfile          :
ImageId                     : ami-043a5034
InstanceMonitoring          : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                : t2.micro
KernelId                    :
KeyName                     :
LaunchConfigurationARN      : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                              e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName     : my-lc-1
PlacementTenancy            :
RamdiskId                   :
SecurityGroups              : {sg-67ef0308}
SpotPrice                   :
UserData                    :
```

示例 3：此示例描述指定的两种启动配置。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

示例 4：此示例描述所有启动配置。

```
Get-ASLaunchConfiguration
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeLaunchConfigurations中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLifecycleHookTypes** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeLifecycleHookTypes。

CLI

Amazon CLI

描述可用的生命周期挂钩类型

此示例描述可用的生命周期挂钩类型。

```
aws autoscaling describe-lifecycle-hook-types
```

输出：

```
{
    "LifecycleHookTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCHING",
        "autoscaling:EC2_INSTANCE_TERMINATING"
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeLifecycleHookTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例列出 Auto Scaling 支持的生命周期挂钩类型。

```
Get-ASLifecycleHookType
```

输出：

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeLifecycleHookTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLifecycleHooks** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeLifecycleHooks。

CLI

Amazon CLI

描述您的生命周期挂钩

此示例描述指定自动扩缩组的生命周期挂钩。

```
aws autoscaling describe-lifecycle-hooks \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "LifecycleHooks": [
        {
            "GlobalTimeout": 3000,
            "HeartbeatTimeout": 30,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-hook",
            "DefaultResult": "ABANDON",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"
        },
        {
            "GlobalTimeout": 6000,
            "HeartbeatTimeout": 60,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-termination-hook",
            "DefaultResult": "CONTINUE",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeLifecycleHooks中的。

PowerShell

用于 PowerShell

示例 1：此示例描述指定的生命周期挂钩。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

输出：

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

示例 2：此示例描述指定自动扩缩组的所有生命周期挂钩。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

示例 3：此示例描述所有自动扩缩组的所有生命周期挂钩。

```
Get-ASLifecycleHook
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeLifecycleHooks中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLoadBalancers** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeLoadBalancers。

CLI

Amazon CLI

描述自动扩缩组的经典负载均衡器

此示例描述指定自动扩缩组的经典负载均衡器。

```
aws autoscaling describe-load-balancers \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "LoadBalancers": [
        {
            "State": "Added",
            "LoadBalancerName": "my-load-balancer"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeLoadBalancers中的。

PowerShell

用于 PowerShell

示例 1：此示例描述指定自动扩缩组的负载均衡器。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

输出：

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeLoadBalancers中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeMetricCollectionTypes** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeMetricCollectionTypes。

CLI

Amazon CLI

描述可用的指标收集类型

此示例描述可用的指标收集类型。

```
aws autoscaling describe-metric-collection-types
```

输出：

```
{
    "Metrics": [
        {
            "Metric": "GroupMinSize"
        },
        {
            "Metric": "GroupMaxSize"
        },
        {
            "Metric": "GroupDesiredCapacity"
        },
        {
            "Metric": "GroupInServiceInstances"
        },
        {
            "Metric": "GroupInServiceCapacity"
        },
        {
            "Metric": "GroupPendingInstances"
        },
        {
            "Metric": "GroupPendingCapacity"
        },
        {
            "Metric": "GroupTerminatingInstances"
        },
        {
            "Metric": "GroupTerminatingCapacity"
        },
        {
            "Metric": "GroupStandbyInstances"
```

```
        },
        {
            "Metric": "GroupStandbyCapacity"
        },
        {
            "Metric": "GroupTotalInstances"
        },
        {
            "Metric": "GroupTotalCapacity"
        }
    ],
    "Granularities": [
        {
            "Granularity": "1Minute"
        }
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 A EC2 uto Scaling 组指标。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeMetricCollectionTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例列出 Auto Scaling 支持的指标收集类型。

```
(Get-ASMetricCollectionType).Metrics
```

输出：

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

示例 2：此示例列出相应的粒度。

```
(Get-ASMetricCollectionType).Granularities
```

输出：

```
Granularity
-----------
1Minute
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeMetricCollectionTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeNotificationConfigurations** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeNotificationConfigurations。

CLI

Amazon CLI

示例 1：描述指定组的通知配置

此示例描述指定自动扩缩组的通知配置。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
```

```
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Auto Scaling 群组缩放时获取 Amazon EC2 SNS 通知。

示例 1：描述指定数量的通知配置

要返回特定数量的通知配置，请使用 max-items 参数。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-auto-scaling-group \
    --max-items 1
```

输出：

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

如果输出包含 NextToken 字段，可返回更多通知配置。要获取其他通知配置，请在后续调用中使用此字段的值和 starting-token 选项，如下所示。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg \
```

```
    --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Auto Scaling 群组缩放时获取 Amazon EC2 SNS 通知。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeNotificationConfigurations中的。

PowerShell

用于 PowerShell

示例 1：此示例描述与指定自动扩缩组关联的通知操作。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

输出：

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

示例 2：此示例描述与所有自动扩缩组关联的通知操作。

```
Get-ASNotificationConfiguration
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeNotificationConfigurations中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribePolicies** 与 CLI 配合使用

以下代码示例演示如何使用 DescribePolicies。

CLI

Amazon CLI

示例 1：描述指定组的扩缩策略

此示例描述指定自动扩缩组的扩展策略。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "alb1000-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
```

```
                    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
                },
                "TargetValue": 1000.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        },
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "cpu40-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 40.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
```

```
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的动态扩展。

示例 2：描述指定名称的扩展策略

要返回特定的扩展策略，请使用 `--policy-names` 选项。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --policy-names cpu40-target-tracking-scaling-policy
```

有关输出示例，请参阅示例 1。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的动态扩展。

示例 3：描述多种扩展策略

要返回特定数量的策略，请使用 `--max-items` 选项。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

有关输出示例，请参阅示例 1。

如果输出包含 `NextToken` 字段，则请在后续调用中使用此字段的值和 `--starting-token` 选项获取其他策略。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-
token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的动态扩展。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribePolicies中的。

PowerShell

用于 PowerShell

示例 1：此示例描述指定自动扩缩组的所有策略。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

输出：

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

示例 2：此示例描述指定自动扩缩组的指定策略。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

示例 3：此示例描述所有自动扩缩组的所有策略。

```
Get-ASPolicy
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribePolicies中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeScalingActivities`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeScalingActivities`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DescribeScalingActivities中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
```

```
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DescribeScalingActivities中的。

CLI

Amazon CLI

示例 1：描述指定组的扩展活动

此示例描述指定自动扩缩组的扩展活动。

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
```

```
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的验证 Aut EC2 o Scaling 组的扩展活动。

示例 2：描述已删除组的扩展活动

要在删除自动扩缩组后描述扩展活动，请添加 `--include-deleted-groups` 选项。

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg \
    --include-deleted-groups
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "Description": "Launching a new EC2 instance.  Status Reason: Your
 Spot request price of 0.001 is lower than the minimum required Spot request
 fulfillment price of 0.0031. Launching EC2 instance failed.",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of
 AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
 capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
 response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
```

```
            "StatusMessage": "Your Spot request price of 0.001 is lower than
 the minimum required Spot request fulfillment price of 0.0031. Launching EC2
 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 疑难解答。

示例 3：描述指定数量的扩展活动

要返回特定数量的活动，请使用 `--max-items` 选项。

```
aws autoscaling describe-scaling-activities \
    --max-items 1
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
```

```
        }
    ]
}
```

如果输出包含 NextToken 字段，可返回更多活动。要获取其他活动，请在后续调用中使用此字段的值和 --starting-token 选项，如下所示。

```
aws autoscaling describe-scaling-activities \
    --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的验证 Aut EC2 o Scaling 组的扩展活动。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeScalingActivities中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                   .autoScalingGroupName(groupName)
                   .maxRecords(10)
                   .build();

           DescribeScalingActivitiesResponse response = autoScalingClient
                   .describeScalingActivities(scalingActivitiesRequest);
           List<Activity> activities = response.activities();
           for (Activity activity : activities) {
               System.out.println("The activity Id is " +
activity.activityId());
```

```
                System.out.println("The activity details are " +
 activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeScalingActivities中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeScalingActivities于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function describeScalingActivities($autoScalingGroupName)
    {
        return $this->autoScalingClient->describeScalingActivities([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DescribeScalingActivities中的。

PowerShell

用于 PowerShell

示例 1：此示例描述指定自动扩缩组过去六周的扩缩活动。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

输出：

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
```

```
                        between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                         from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                        en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

示例 2：此示例描述指定的扩缩活动。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

示例 3：此示例描述所有自动扩缩组过去六周的扩缩活动。

```
Get-ASScalingActivity
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeScalingActivities中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
 Any]]:
        """
        Gets information about scaling activities for the group. Scaling
 activities
        are things like instances stopping or starting in response to user
 requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
 the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling
 activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator =
 paginator.paginate(AutoScalingGroupName=group_name)
```

```
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
        else:
            return activities
```

- 有关 API 的详细信息，请参阅适用DescribeScalingActivities于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
```

```
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- 有关 API 的详细信息，请参阅适用DescribeScalingActivities于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeScalingProcessTypes** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeScalingProcessTypes。

CLI

Amazon CLI

描述可用的进程类型

此示例描述可用的流程类型。

```
aws autoscaling describe-scaling-process-types
```

输出：

```
{
    "Processes": [
```

```
        {
            "ProcessName": "AZRebalance"
        },
        {
            "ProcessName": "AddToLoadBalancer"
        },
        {
            "ProcessName": "AlarmNotification"
        },
        {
            "ProcessName": "HealthCheck"
        },
        {
            "ProcessName": "InstanceRefresh"
        },
        {
            "ProcessName": "Launch"
        },
        {
            "ProcessName": "ReplaceUnhealthy"
        },
        {
            "ProcessName": "ScheduledActions"
        },
        {
            "ProcessName": "Terminate"
        }
    ]
}
```

有关更多信息，请参阅 A mazon A EC2 uto Scaling 用户指南中的暂停和恢复扩展流程。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeScalingProcessTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例列出 Auto Scaling 支持的进程类型。

```
Get-ASScalingProcessType
```

输出：

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeScalingProcessTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeScheduledActions** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeScheduledActions。

CLI

Amazon CLI

示例 1：描述所有计划操作

此示例描述所有计划操作。

```
aws autoscaling describe-scheduled-actions
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
```

```
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 2：描述指定组的计划操作

要描述特定自动扩缩组的计划操作，请使用 `--auto-scaling-group-name` 选项。

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 3：描述指定的计划操作

要描述特定的计划操作，请使用 `--scheduled-action-names` 选项。

```
aws autoscaling describe-scheduled-actions \
    --scheduled-action-names my-recurring-action
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 4：描述具有指定开始时间的计划操作

要描述在特定时间开始的计划操作，请使用 `--start-time` 选项。

```
aws autoscaling describe-scheduled-actions \
    --start-time "2023-12-01T04:00:00Z"
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 5：描述在指定时间结束的计划操作

要描述在特定时间结束的计划操作，请使用 `--end-time` 选项。

```
aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
```

```
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 6：描述指定数量的计划操作

要返回特定数量的计划操作，请使用 `--max-items` 选项。

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

输出：

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

如果输出包含 NextToken 字段，可返回更多计划操作。要获取其他计划操作，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeScheduledActions中的。

PowerShell

用于 PowerShell

示例 1：此示例描述指定自动扩缩组的计划扩缩操作。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

输出：

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

示例 2：此示例描述指定的计划扩缩操作。

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

示例 3：此示例描述在指定时间开始的计划扩缩操作。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

示例 4：此示例描述在指定时间结束的计划扩缩操作。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

示例 5：此示例描述所有自动扩缩组的计划扩缩操作。

```
Get-ASScheduledAction
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考[DescribeScheduledActions](#)中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅[将此服务与 Amazon SDK 配合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeTags** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeTags。

CLI

Amazon CLI

描述所有标签

此示例描述所有标签。

```
aws autoscaling describe-tags
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "Research",
            "Key": "Dept"
        },
```

```
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "WebServer",
            "Key": "Role"
        }
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 A EC2 uto Scaling 组和实例添加标签。

示例 2：描述指定组的标签

要描述特定自动扩缩组的标签，请使用 `--filters` 选项。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 A EC2 uto Scaling 组和实例添加标签。

示例 3：描述指定数量的标签

要返回特定数量的标签，请使用 `--max-items` 选项。

```
aws autoscaling describe-tags \
    --max-items 1
```

如果输出包含 `NextToken` 字段，则可返回更多标签。要获取其他标签，请在后续调用中使用此字段的值和 `--starting-token` 选项，如下所示。

```
aws autoscaling describe-tags \
    --filters Name=auto-scaling-group,Values=my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 A EC2 uto Scaling 组和实例添加标签。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeTags中的。

PowerShell

用于 PowerShell

示例 1：此示例描述键值为“myTag”或“myTag2”的标签。筛选器名称的可能值为 "、auto-scaling-group 'key'、'value' 和 'propagate-at-launch'。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

输出：

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 为过滤器参数创建过滤器。

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

示例 3：此示例描述所有自动扩缩组的所有标签。

```
Get-ASTag
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeTags中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeTerminationPolicyTypes** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeTerminationPolicyTypes。

CLI

Amazon CLI

描述可用的终止策略类型

此示例描述可用的终止策略类型。

```
aws autoscaling describe-termination-policy-types
```

输出：

```
{
    "TerminationPolicyTypes": [
        "AllocationStrategy",
        "ClosestToNextInstanceHour",
        "Default",
        "NewestInstance",
        "OldestInstance",
        "OldestLaunchConfiguration",
        "OldestLaunchTemplate"
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的控制哪些 A EC2 uto Scaling 实例在缩容期间终止。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeTerminationPolicyTypes中的。

PowerShell

用于 PowerShell

示例 1：此示例列出 Auto Scaling 支持的终止策略。

```
Get-ASTerminationPolicyType
```

输出：

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DescribeTerminationPolicyTypes中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DetachInstances** 与 CLI 配合使用

以下代码示例演示如何使用 DetachInstances。

CLI

Amazon CLI

将实例与自动扩缩组分离

此示例将指定实例与指定的自动扩缩组分离。

```
aws autoscaling detach-instances \
    --instance-ids i-030017cfa84b20135 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
            "AutoScalingGroupName": "my-asg",
            "Description": "Detaching EC2 instance: i-030017cfa84b20135",
```

```
            "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was
 detached in response to a user request, shrinking the capacity from 2 to 1.",
            "StartTime": "2020-04-12T15:02:16.179Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DetachInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的实例与指定的自动扩缩组分离，并减少所需容量以使 Auto Scaling 不启动替换实例。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

示例 2：此示例将在不减少所需容量的情况下将指定的实例与指定的自动扩缩组分离。Auto Scaling 会启动一个替换实例。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DetachInstances中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DetachLoadBalancers** 与 CLI 配合使用

以下代码示例演示如何使用 DetachLoadBalancers。

CLI

Amazon CLI

将经典负载均衡器与自动扩缩组分离

此示例将指定的经典负载均衡器与指定的自动扩缩组分离。

```
aws autoscaling detach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的将负载均衡器附加到您的 A EC2 uto Scaling 组。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DetachLoadBalancers中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的负载均衡器与指定的自动扩缩组分离。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DetachLoadBalancers中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisableMetricsCollection`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DisableMetricsCollection`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
```

```
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考DisableMetricsCollection中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);
```

```
        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DisableMetricsCollection中的。

CLI

Amazon CLI

禁用自动扩缩组指标收集

此示例禁用指定自动扩缩组的 GroupDesiredCapacity 指标的收集。

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Aut EC2 o S caling 组和实例的监控 CloudWatch 指标。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DisableMetricsCollection中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DisableMetricsCollection中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- 有关 API 的详细信息，请参阅适用DisableMetricsCollection于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
```

```
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考DisableMetricsCollection中的。

PowerShell

用于 PowerShell

示例 1：此示例禁用对指定自动扩缩组指定指标的监控。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

示例 2：此示例禁用对指定自动扩缩组所有指标的监控。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考DisableMetricsCollection中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
```

```
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.

        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
'{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- 有关 API 的详细信息，请参阅适用DisableMetricsCollection于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- 有关 API 的详细信息，请参阅适用DisableMetricsCollection于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`EnableMetricsCollection`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `EnableMetricsCollection`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考EnableMetricsCollection中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考EnableMetricsCollection中的。

CLI

Amazon CLI

示例 1：启用自动扩缩组的指标收集

此示例启用指定自动扩缩组的数据收集。

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Aut EC2 o S caling 组和实例的监控 CloudWatch 指标。

示例 2：收集自动扩缩组指定指标的相关数据

要收集特定指标的相关数据，请使用 `--metrics` 选项。

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity --granularity "1Minute"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Aut EC2 o S caling 组和实例的监控 CloudWatch 指标。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考EnableMetricsCollection中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考EnableMetricsCollection中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- 有关 API 的详细信息，请参阅适用EnableMetricsCollection于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考EnableMetricsCollection中的。

PowerShell

用于 PowerShell

示例 1：此示例启用对指定自动扩缩组指定指标的监控。

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

示例 2：此示例启用对指定自动扩缩组所有指标的监控。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考[EnableMetricsCollection](#)中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
 Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
 activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
 collection.
```

```
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
                logger.error(
                    f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- 有关 API 的详细信息，请参阅适用EnableMetricsCollection于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- 有关 API 的详细信息，请参阅适用EnableMetricsCollection于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `EnterStandby` 与 CLI 配合使用

以下代码示例演示如何使用 `EnterStandby`。

CLI

Amazon CLI

将实例移入备用模式

此示例将指定实例置于备用模式。这对于更新当前正在使用的实例或对其进行问题排查非常有用。

```
aws autoscaling enter-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
 moved to standby in response to a user request, shrinking the capacity from 1 to
 0.",
            "StartTime": "2020-10-31T20:31:00.949Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 实例生命周期。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考EnterStandby中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定实例置于备用模式并减少所需容量，以使 Auto Scaling 不启动替换实例。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

示例 2：此示例在不减少所需容量的情况下将指定实例置于备用模式。Auto Scaling 会启动一个替换实例。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考EnterStandby中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ExecutePolicy** 与 CLI 配合使用

以下代码示例演示如何使用 ExecutePolicy。

CLI

Amazon CLI

执行扩缩策略

此示例将对指定的自动扩缩组执行名为 my-step-scale-out-policy 的扩展策略。

```
aws autoscaling execute-policy \
    --auto-scaling-group-name my-asg \
    --policy-name  my-step-scale-out-policy \
    --metric-value 95 \
    --breach-threshold 80
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的步骤和简单扩展策略。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ExecutePolicy中的。

PowerShell

用于 PowerShell

示例 1：此示例对指定的自动扩缩组执行指定的策略。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

示例 2：此示例在等待冷却时间结束后，对指定的自动扩缩组执行指定的策略。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考ExecutePolicy中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ExitStandby** 与 CLI 配合使用

以下代码示例演示如何使用 ExitStandby。

CLI

Amazon CLI

将实例移出备用模式

此示例将指定的实例移出备用模式。

```
aws autoscaling exit-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
 moved out of standby in response to a user request, increasing the capacity from
 0 to 1.",
            "StartTime": "2020-10-31T20:32:50.222Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的临时从 Aut EC2 o Sc aling 组中移除实例。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ExitStandby中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的实例移出备用模式。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

输出：

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考ExitStandby中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutLifecycleHook** 与 CLI 配合使用

以下代码示例演示如何使用 PutLifecycleHook。

CLI

Amazon CLI

示例 1：创建生命周期挂钩

此示例创建一个生命周期挂钩，该挂钩将在任何新启动的实例上调用，超时时间为 4800 秒。这对于在用户数据脚本完成之前保持实例处于等待状态或使用调用 Lambd Amazon a 函数非常有用。 EventBridge

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --heartbeat-timeout 4800
```

此命令不生成任何输出。如果已存在同名的生命周期挂钩，则该挂钩将被新的生命周期挂钩覆盖。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

示例 2：发送 Amazon SNS 电子邮件以通知您实例状态转换

此示例创建一个包含 Amazon SNS 主题和 IAM 角色的生命周期挂钩，用于在实例启动时接收通知。

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

示例 3：向 Amazon SQS 队列发布消息

此示例创建一个生命周期挂钩，该挂钩将包含元数据的消息发布到指定的 Amazon SQS 队列。

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
```

```
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \
    --role-arn arn:aws:iam::123456789012:role/my-notification-role \
    --notification-metadata "SQS message metadata"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考PutLifecycleHook中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的生命周期挂钩添加到指定的自动扩缩组。

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考PutLifecycleHook中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutNotificationConfiguration** 与 CLI 配合使用

以下代码示例演示如何使用 PutNotificationConfiguration。

CLI

Amazon CLI

添加通知

此示例将指定的通知添加到指定的自动扩缩组。

```
aws autoscaling put-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --notification-type autoscaling:TEST_NOTIFICATION
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Auto Scaling 群组缩放时获取 Amazon EC2 SNS 通知。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考PutNotificationConfiguration中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的 Auto Scaling 组配置为在启动 EC2 实例时向指定的 SNS 主题发送通知。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

示例 2：此示例将指定的 Auto Scaling 组配置为在启动或终止 EC2 实例时向指定的 SNS 主题发送通知。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考PutNotificationConfiguration中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutScalingPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 PutScalingPolicy。

CLI

Amazon CLI

向自动扩缩组添加目标跟踪扩缩策略

以下 put-scaling-policy 示例将目标跟踪扩展策略应用到指定的自动扩缩组。输出包含代表您创建的两个 CloudWatch 警报的 ARNs 和名称。如果已存在同名扩缩策略，则该扩展策略将被新扩展策略覆盖。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

config.json 的内容：

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
          "PredefinedMetricType": "ALBRequestCountPerTarget",
          "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
     }
}
```

输出：

```
{
     "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
     "Alarms": [
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
             "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
         },
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
```

```
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
          }
      ]
 }
```

有关更多示例，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon 命令行界面 (Amazon CLI) 扩展策略示例。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考PutScalingPolicy中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的策略添加到指定的自动扩缩组。指定的调整类型决定了如何解释 ScalingAdjustment 参数。使用 “ChangeInCapacity” 时，正值将按指定数量的实例增加容量，负值则按指定的实例数量减少容量。

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

输出：

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考PutScalingPolicy中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutScheduledUpdateGroupAction** 与 CLI 配合使用

以下代码示例演示如何使用 PutScheduledUpdateGroupAction。

CLI

Amazon CLI

示例 1：向自动扩缩组添加计划操作

此示例将指定的计划操作添加到指定的自动扩缩组。

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action \
    --start-time "2023-05-12T08:00:00Z" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

此命令不生成任何输出。如果已存在同名的计划操作，则该计划操作将被新计划操作覆盖。

有关更多示例，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

示例 2：指定周期性计划

此示例创建一个计划操作，以按计划在每年一月、六月和十二月的第一天 00:30 执行的周期性计划进行扩展。

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action \
    --recurrence "30 0 1 1,6,12 *" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

此命令不生成任何输出。如果已存在同名的计划操作，则该计划操作将被新计划操作覆盖。

有关更多示例，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考PutScheduledUpdateGroupAction中的。

PowerShell

用于 PowerShell

示例 1：此示例创建或更新一次性计划操作，以在指定的开始时间更改所需容量。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考PutScheduledUpdateGroupAction中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **RecordLifecycleActionHeartbeat** 与 CLI 配合使用

以下代码示例演示如何使用 RecordLifecycleActionHeartbeat。

CLI

Amazon CLI

记录生命周期操作心跳

此示例记录生命周期操作心跳，以使实例保持待处理状态。

```
aws autoscaling record-lifecycle-action-heartbeat \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考RecordLifecycleActionHeartbeat中的。

PowerShell

用于 PowerShell

示例 1：此示例记录指定生命周期操作的心跳。这会使实例保持待处理状态，直到您完成自定义操作。

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考RecordLifecycleActionHeartbeat中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ResumeProcesses** 与 CLI 配合使用

以下代码示例演示如何使用 `ResumeProcesses`。

CLI

Amazon CLI

恢复暂停的进程

此示例恢复指定自动扩缩组的指定暂停扩展流程。

```
aws autoscaling resume-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

此命令不生成任何输出。

有关更多信息，请参阅 A mazon A EC2 uto Scaling 用户指南中的暂停和恢复扩展流程。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ResumeProcesses中的。

PowerShell

用于 PowerShell

示例 1：此示例恢复指定自动扩缩组的指定 Auto Scaling 进程。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

示例 2：此示例恢复指定自动扩缩组的所有暂停 Auto Scaling 进程。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考ResumeProcesses中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetDesiredCapacity`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetDesiredCapacity`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Set the desired capacity of an Auto Scaling group.
/// </summary>
```

```
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考SetDesiredCapacity中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考SetDesiredCapacity中的。

CLI

Amazon CLI

为自动扩缩组设置所需容量

此示例为指定的自动扩缩组设置所需容量。

```
aws autoscaling set-desired-capacity \
    --auto-scaling-group-name my-asg \
    --desired-capacity 2 \
    --honor-cooldown
```

如果成功，该命令将返回到提示符。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetDesiredCapacity中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SetDesiredCapacity中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- 有关 API 的详细信息，请参阅适用SetDesiredCapacity于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考SetDesiredCapacity中的。

PowerShell

用于 PowerShell

示例 1：此示例设置指定自动扩缩组的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

示例 2：此示例设置指定自动扩缩组的大小，并等待冷却时间结束后再扩缩到新大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考SetDesiredCapacity中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
 keep the
```

```
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- 有关 API 的详细信息，请参阅适用SetDesiredCapacity于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity})").as_str(),
               &err,
           ));
       }
       Ok(())
   }
```

- 有关 API 的详细信息，请参阅适用SetDesiredCapacity于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SetInstanceHealth** 与 CLI 配合使用

以下代码示例演示如何使用 SetInstanceHealth。

CLI

Amazon CLI

设置实例的运行状况

此示例将指定实例的运行状况设置为 Unhealthy。

```
aws autoscaling set-instance-health \
    --instance-id i-061c63c5eb45f0416 \
    --health-status Unhealthy
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetInstanceHealth中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定实例的状态设置为“运行状况不佳”，使其停止服务。Auto Scaling 会终止并替换该实例。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

示例 2：此示例将指定实例的状态设置为“运行状况正常”，使其保持服务状态。未遵守自动扩缩组的任何运行状况检查宽限期。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考SetInstanceHealth中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SetInstanceProtection** 与 CLI 配合使用

以下代码示例演示如何使用 SetInstanceProtection。

CLI

Amazon CLI

示例 1：启用实例的实例保护设置

此示例启用对指定实例的实例保护。

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg --protected-from-scale-in
```

此命令不生成任何输出。

示例 2：禁用实例的实例保护设置

此示例禁用对指定实例的实例保护。

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --no-protected-from-scale-in
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetInstanceProtection中的。

PowerShell

用于 PowerShell

示例 1：此示例启用指定实例的实例保护。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

示例 2：此示例禁用指定实例的实例保护。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考SetInstanceProtection中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 SuspendProcesses 与 CLI 配合使用

以下代码示例演示如何使用 SuspendProcesses。

CLI

Amazon CLI

暂停 Auto Scaling 进程

此示例暂停指定自动扩缩组的指定扩展流程。

```
aws autoscaling suspend-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

此命令不生成任何输出。

有关更多信息，请参阅 A mazon A EC2 uto Scaling 用户指南中的暂停和恢复扩展流程。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SuspendProcesses中的。

PowerShell

用于 PowerShell

示例 1：此示例暂停了指定自动扩缩组的指定 Auto Scaling 进程。

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

示例 2：此示例暂停了指定自动扩缩组的所有 Auto Scaling 进程。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考SuspendProcesses中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`TerminateInstanceInAutoScalingGroup`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `TerminateInstanceInAutoScalingGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
```

```
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考TerminateInstanceInAutoScalingGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);
```

```
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考TerminateInstanceInAutoScalingGroup中的。

CLI

Amazon CLI

终止自动扩缩组中的实例

此示例在不更新指定自动扩缩组大小的情况下终止该组中的指定实例。Amazon A EC2 uto Scaling 会在指定实例终止后启动替换实例。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
    --instance-id i-061c63c5eb45f0416 \
    --no-should-decrement-desired-capacity
```

输出：

```
{
    "Activities": [
        {
            "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
            "AutoScalingGroupName": "my-asg",
```

```
            "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
            "Cause": "",
            "StartTime": "2020-10-31T20:34:25.680Z",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考[TerminateInstanceInAutoScalingGroup](#)中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                   .instanceId(instanceId)
                   .shouldDecrementDesiredCapacity(false)
                   .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance " + instanceId);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考TerminateInstanceInAutoScalingGroup中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- 有关 API 的详细信息，请参阅适用TerminateInstanceInAutoScalingGroup于 K otlin 的 Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' =>
 $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
 $attempts < 5) {
                error_log("Cannot terminate an instance while it is still
 pending. Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考TerminateInstanceInAutoScalingGroup中的。

PowerShell

用于 PowerShell

示例 1：此示例终止了指定的实例，并减少其自动扩缩组的所需容量，以使 Auto Scaling 不启动替换实例。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

示例 2：此示例在不减少指定实例自动扩缩组所需容量的情况下终止该实例。Auto Scaling 会启动一个替换实例。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
```

```
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考[TerminateInstanceInAutoScalingGroup](#)中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
 capacity
                                  of the group. When passing True for this
 parameter,
```

```
                                    you can stop an instance without having a
replacement
                                    instance start when the desired capacity
threshold is
                                    crossed.
        :return: A dictionary containing details of the scaling activity that
occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- 有关 API 的详细信息，请参阅适用TerminateInstanceInAutoScalingGroup于 Python 的 Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
```

```
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- 有关 API 的详细信息，请参阅适用TerminateInstanceInAutoScalingGroup于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UpdateAutoScalingGroup`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UpdateAutoScalingGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 构建和管理弹性服务

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };
```

```
        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考UpdateAutoScalingGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考UpdateAutoScalingGroup中的。

CLI

Amazon CLI

示例 1：更新自动扩缩组的大小限制

该示例更新指定的自动扩缩组，该组的最小大小为 1，最大大小为 10。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --min-size 2 \
    --max-size 10
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 Aut EC2 o S caling 组设置容量限制。

示例 2：添加 Elastic Load Balancing 运行状况检查并指定要使用的可用区和子网

此示例更新指定的自动扩缩组以添加 Elastic Load Balancing 运行状况检查。此命令还会--vpc-zone-identifier使用多个可用区 IDs 中的子网列表更新的值。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Elastic Load Balancing 和 Amazon A EC2 uto Scaling。

示例 3：更新置放群组和终止策略

此示例更新要使用的置放群组和终止策略。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --placement-group my-placement-group \
    --termination-policies "OldestInstance"
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 A EC2 uto Scaling 群组。

示例 4：使用最新版本的启动模板

此示例将指定的自动扩缩组更新为使用最新版本的指定启动模板。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的启动模板。

示例 5：使用特定版本的启动模板

此示例将指定的自动扩缩组更新为使用特定版本的启动模板，而不是最新或默认版本。

```
aws autoscaling update-auto-scaling-group \
```

```
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的启动模板。

示例 6：定义混合实例策略并启用容量再平衡

此示例将指定的自动扩缩组更新为使用混合实例策略并启用容量再平衡。此结构允许您指定具有竞价和按需容量的组，并针对不同的架构使用不同的启动模板。

```
aws autoscaling update-auto-scaling-group \
    --cli-input-json file://~/config.json
```

config.json 的内容：

```
{
    "AutoScalingGroupName": "my-asg",
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c6g.large",
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template-for-arm",
                        "Version": "$Latest"
                    }
                },
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                }
            ]
        },
```

```
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    }
}
```

此命令不生成任何输出。

有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的具有多种实例类型和购买选项的 A EC2 uto Scaling 群组。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考UpdateAutoScalingGroup中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
```

```
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考[UpdateAutoScalingGroup](#)中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 [Amazon 代码示例存储库](#)中进行设置和运行。

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }
```

```
    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- 有关 API 的详细信息，请参阅适用UpdateAutoScalingGroup于 K otlin 的Amazon SDK API 参考。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
```

```
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考[UpdateAutoScalingGroup](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例更新指定自动扩缩组的最小和最大大小。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

示例 2：此示例更新指定自动扩缩组的默认冷却时间。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

示例 3：此示例更新指定自动扩缩组的可用区。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

示例 4：此示例更新指定的自动扩缩组以使用 Elastic Load Balancing 运行状况检查。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考[UpdateAutoScalingGroup](#)中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
```

```
                    "The Auto Scaling group '%s' is currently in use and cannot
 be modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
 to update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- 有关 API 的详细信息，请参阅适用UpdateAutoScalingGroup于 Python 的Amazon SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
 Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
```

```
}
```

- 有关 API 的详细信息，请参阅适用UpdateAutoScalingGroup于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Auto Scaling 的场景 Amazon SDKs

以下代码示例向您展示了如何使用在 Auto Scaling 中实现常见场景 Amazon SDKs。这些场景向您展示如何通过调用 Auto Scaling 中的多个函数或与其他 Amazon Web Services 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用 Amazon SDK 构建和管理弹性服务

使用 Amazon SDK 构建和管理弹性服务

以下代码示例展示如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (Amazon EC2) 实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 HTTP 请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个 EC2 实例上运行 Python 网络服务器来处理 HTTP 请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 Amazon Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
       if (interactive)
           await DemoActionChoices();

       Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
       Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

       await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

       Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
       Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
       if (interactive)
           await DemoActionChoices();

       Console.WriteLine("Let's reinstate the recommendation service.\n");
       await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
       Console.WriteLine(
           "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
           "access the DynamoDB recommendation table.\n"
       );
       await _autoScalerWrapper.CreateInstanceProfileWithName(
           _autoScalerWrapper.BadCredsPolicyName,
           _autoScalerWrapper.BadCredsRoleName,
           _autoScalerWrapper.BadCredsProfileName,
           ssmOnlyPolicy,
           new List<string> { "AmazonSSMManagedInstanceCore" }
       );
       var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
       var badInstanceId = instances.First();
       var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
       Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                     "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

创建一个包含弹性负载均衡操作的类。

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
```

```
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
```

```
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
```

```
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

创建一个包含 Systems Manager 操作的类。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
```

```
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
```

```
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances

- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
```

```
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
                that were created for this demo.
                """);
```

```
        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
```

```
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
   }

   /**
    * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
    * terminated, it can no longer be accessed.
    */
   public void terminateInstance(String instanceId) {
       TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
               .builder()
               .instanceId(instanceId)
               .shouldDecrementDesiredCapacity(false)
               .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
       System.out.format("Terminated instance %s.", instanceId);
   }

   /**
    * Replaces the profile associated with a running instance. After the profile
is
    * replaced, the instance is rebooted to ensure that it uses the new profile.
    * When
    * the instance is ready, Systems Manager is used to restart the Python web
    * server.
    */
   public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
           throws InterruptedException {
       // Create an IAM instance profile specification.
       software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
               .builder()
               .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                             // name.
               .build();

       // Replace the IAM instance profile association for the EC2 instance.
       ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包含弹性负载均衡操作的类。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
   // Deletes a load balancer.
   public void deleteLoadBalancer(String lbName) {
       try {
           // Use a waiter to delete the Load Balancer.
           DescribeLoadBalancersResponse res = getLoadBalancerClient()
                   .describeLoadBalancers(describe -> describe.names(lbName));
           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                   .build();

           getLoadBalancerClient().deleteLoadBalancer(
                   builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancersDeleted(request);
           waiterResponse.matched().response().ifPresent(System.out::println);

       } catch (ElasticLoadBalancingV2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println(lbName + " was deleted.");
   }

   // Deletes the target group.
   public void deleteTargetGroup(String targetGroupName) {
       try {
           DescribeTargetGroupsResponse res = getLoadBalancerClient()
                   .describeTargetGroups(describe ->
describe.names(targetGroupName));
           getLoadBalancerClient()
                   .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
       } catch (ElasticLoadBalancingV2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println(targetGroupName + " was deleted.");
   }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
        return false;
```

```
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
```

```
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

创建一个包含 Systems Manager 操作的类。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
```

```
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
```

```
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

创建部署所有资源的步骤。

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
```

```
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
```

```
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
```

```
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
```

```
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
```

```
      ),
    }),
  );
}),
new ScenarioOutput(
  "createdInstanceRole",
  MESSAGES.createdInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioOutput(
  "attachingPolicyToRole",
  MESSAGES.attachingPolicyToRole
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
),
new ScenarioAction("attachPolicyToRole", async (state) => {
  const client = new IAMClient({});
  await client.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.instanceRoleName,
      PolicyArn: state.instancePolicyArn,
    }),
  );
}),
new ScenarioOutput(
  "attachedPolicyToRole",
  MESSAGES.attachedPolicyToRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioOutput(
  "creatingInstanceProfile",
  MESSAGES.creatingInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  ),
),
new ScenarioAction("createInstanceProfile", async (state) => {
  const client = new IAMClient({});
  const {
    InstanceProfile: { Arn },
  } = await client.send(
```

```
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
```

```
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
```

```
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
```

```
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
```

```
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
```

```
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
```

```
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
```

```
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

创建运行演示的步骤。

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
```

```
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
```

```
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
```

```
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
```

```
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
```

```
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
```

```
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
```

```
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
```

```
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
```

```
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
```

```
    }),
  );

  return InstanceProfile;
}
```

创建销毁所有资源的步骤。

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
```

```
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
```

```
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
```

```
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
```

```
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
```

```
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
```

```
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
```

```
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
```

```
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
```

```
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
```

```
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile

- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
 DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
```

```
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)
```

```
        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
```

```
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
```

```
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()
```

```
        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()
```

```
        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
```

```
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
```

```
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```

```
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
 definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
```

```
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
```

```
        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
```

```
                                    use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

```
    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
```

```
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
```

```
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
```

```
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
```

```
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
```

```
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
```

```
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
```

```
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
```

```
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

```
   def verify_inbound_port(
       self, vpc: Dict[str, Any], port: int, ip_address: str
   ) -> Tuple[Dict[str, Any], bool]:
       """
       Verify the default security group of the specified VPC allows ingress
from this
       computer. This can be done by allowing ingress from this computer's IP
       address. In some situations, such as connecting from a corporate network,
you
       must instead specify a prefix list ID. You can also temporarily open the
port to
       any IP address while running this example. If you do, be sure to remove
public
       access when you're done.

       :param vpc: The VPC used by this example.
       :param port: The port to verify.
       :param ip_address: This computer's IP address.
       :return: The default security group of the specified VPC, and a value
that indicates
                whether the specified port is open.
       """
       try:
           response = self.ec2_client.describe_security_groups(
               Filters=[
                   {"Name": "group-name", "Values": ["default"]},
                   {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
               ]
           )
           sec_group = response["SecurityGroups"][0]
           port_is_open = False
           log.info(f"Found default security group {sec_group['GroupId']}.")

           for ip_perm in sec_group["IpPermissions"]:
               if ip_perm.get("FromPort", 0) == port:
                   log.info(f"Found inbound rule: {ip_perm}")
                   for ip_range in ip_perm["IpRanges"]:
                       cidr = ip_range.get("CidrIp", "")
                       if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                           port_is_open = True
                   if ip_perm["PrefixListIds"]:
                       port_is_open = True
                   if not port_is_open:
```

```
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
```

```
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
```

```
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

创建一个包含弹性负载均衡操作的类。

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
```

```
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
```

```
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
```

```
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.
```

```
        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

```
   def create_listener(
       self,
       load_balancer_name: str,
       target_group: Dict[str, Any],
   ) -> Dict[str, Any]:
       """
       Creates a listener for the specified load balancer that forwards requests
to the
       specified target group.

       :param load_balancer_name: The name of the load balancer to create a
listener for.
       :param target_group: An existing target group that is added as a listener
to the
                            load balancer.
       :return: Data about the newly created listener.
       """
       try:
           # Retrieve the load balancer ARN
           load_balancer_response = self.elb_client.describe_load_balancers(
               Names=[load_balancer_name]
           )
           load_balancer_arn = load_balancer_response["LoadBalancers"][0][
               "LoadBalancerArn"
           ]

           # Create the listener
           response = self.elb_client.create_listener(
               LoadBalancerArn=load_balancer_arn,
               Protocol=target_group["Protocol"],
               Port=target_group["Port"],
               DefaultActions=[
                   {
                       "Type": "forward",
                       "TargetGroupArn": target_group["TargetGroupArn"],
                   }
               ],
           )
           log.info(
               f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
           )
           return response["Listeners"][0]
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
```

```
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```

```
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
```

```
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
```

```
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
```

```
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

创建一个包含 Systems Manager 操作的类。

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
```

```
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

对 Amazon A EC2 uto Scaling 中的问题进行故障排除

Amazon A EC2 uto Scaling 提供了具体的描述性错误来帮助您解决问题。可以从扩展活动的描述中发现错误消息。

主题

检索来自扩缩活动的错误消息

要从扩展活动描述中检索错误消息，请使用 describe-scaling-activities 命令。您拥有可追溯到 6 周的扩展活动记录。扩展活动按开始时间排序，首先列出最新的扩展活动。

> **Note**
>
> 扩展活动还会显示在 Amazon A EC2 uto Scaling 控制台的活动历史记录中，位于 Auto Scaling 组的 “活动” 选项卡上。

要查看特定 Auto Scaling 组的扩展活动，请使用以下命令。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

在下面的示例响应中，StatusCode 包含活动的当前状态，StatusMessage 包含错误消息。

```
{
    "Activities": [
        {
```

```
            "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e.  Status
 Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with
 token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed
 with ABANDON Result",
            "Cause": "At 2021-01-11T00:35:52Z a user request created an
 AutoScalingGroup changing the desired capacity from 0 to 1.  At 2021-01-11T00:35:53Z
 an instance was started in response to a difference between desired and actual
 capacity, increasing the capacity from 0 to 1.",
            "StartTime": "2021-01-11T00:35:55.542Z",
            "EndTime": "2021-01-11T01:06:31Z",
            "StatusCode": "Cancelled",
            "StatusMessage": "Instance failed to complete user's Lifecycle Action:
 Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned:
 Lifecycle Action Completed with ABANDON Result",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
     ...
    ]
}
```

有关输出中字段的描述，请参阅《Amazon A EC2 uto Scaling API 参考》中的 “活动”。

要查看已删除组的扩展活动

要在删除 Auto Scaling 组后查看伸缩活动，describe-scaling-activities请在命令中添加以下`--include-deleted-groups`选项。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-
deleted-groups
```

以下是示例响应，其中包含已删除组的扩展活动。

```
{
    "Activities": [
        {
```

```
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance.  Status Reason: Your Spot
 request price of 0.001 is lower than the minimum required Spot request fulfillment
 price of 0.0031. Launching EC2 instance failed.",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup
 constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3.
  At 2021-01-13T20:47:27Z an instance was started in response to a difference between
 desired and actual capacity, increasing the capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than the
 minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance
 failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
     ...
    ]
}
```

关闭扩缩活动

如果您需要在不受扩缩策略或计划操作干扰的情况下调查问题，则可以使用以下选项：

- 通过暂停 AlarmNotification 和 ScheduledActions 进程，防止所有动态扩缩策略和计划操作更改组的所需容量。有关更多信息，请参阅 暂停和恢复 Amazon A EC2 uto Scaling 流程。
- 禁用单个动态扩缩策略，以使其不会因负载变化而更改组的所需容量。有关更多信息，请参阅 禁用 Auto Scaling 组的扩缩策略。
- 通过禁用策略的横向缩减部分，将单个目标跟踪扩缩策略更新为仅横向扩展（增加容量）。这种方法可以防止组的所需容量缩小，但允许在负载增加时增加容量。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的目标跟踪扩展政策。
- 将您的预测性扩展策略更新为仅预测模式。在仅预测模式下，预测性扩展将继续生成预测，但不会自动增加容量。有关更多信息，请参阅 创建自动扩缩组的预测性扩展策略。

其他故障排除资源

以下页面提供了有关排除 Amazon A EC2 uto Scaling 问题的更多信息。

- 验证 Auto Scaling 组的扩缩活动
- 在 Amazon A EC2 uto Scaling 控制台中查看监控图表
- 自动扩缩组中实例的运行状况检查
- 生命周期钩子的注意事项和限制
- 在自动扩缩组中完成生命周期操作
- 使用 Amazon VPC 为 Auto Scaling 实例提供网络连接
- 临时从 Auto Scaling 组中移除实例
- 禁用 Auto Scaling 组的扩缩策略
- 暂停和恢复 Amazon A EC2 uto Scaling 流程
- 控制在横向缩减过程中要终止的 Auto Scaling 实例
- 删除 Auto Scaling 基础设施
- 自动扩缩资源和组的配额

以下 Amazon 资源也可能有所帮助：

- Amazon 知识中心中的 Amazon A EC2 uto Scaling 主题
- Amazon Re: Post 上的 Amazon A EC2 uto Scaling 问题
- Amazon A EC2 uto Scaling 在 Amazon 计算博客上发布的文章
- 《Amazon CloudFormation 用户指南》 CloudFormation 中的疑难解答

故障排除通常需要由专家或多个帮助者进行迭代查询和发现。如果您在尝试本节中的建议后仍然遇到问题，请联系 Amazon Web Services 支持（在 “支持” Amazon Web Services Management Console、“支持中心” 中）或使用 A mazon A EC2 uto Scaling 标签在 Amazon re: Pos t 上提问。

对 Amazon A EC2 uto Scaling 进行故障排除： EC2 实例启动失败

本页提供有关您的 EC2 实例启动失败、潜在原因以及您可以采取的解决问题的步骤的信息。

要检索错误消息，请参阅检索来自扩缩活动的错误消息。

当您的 EC2 实例启动失败时，您可能会收到以下一条或多条错误消息：

启动问题

当前不支持请求的配置。

原因：您的启动模板或启动配置中的某些选项可能与实例类型不兼容，或者您请求的 Amazon 区域或可用区域可能不支持实例配置。

解决方案：尝试其他实例配置。要搜索符合您要求的实例类型，请参阅亚马逊 EC2 用户指南中的查找亚马逊 EC2 实例类型。

有关解决该问题的更多指导，请检查以下内容：

- 确保您选择了实例类型支持的 AMI。例如，如果实例类型使用基于 ARM 的 Amazon Graviton 处理器而不是英特尔至强处理器，则需要兼容 ARM 的 AMI。有关选择兼容实例类型的更多信息，请参阅 Amazon EC2 用户指南中的更改实例类型的兼容性。
- 测试实例类型在所请求的区域和可用区中可用。最新一代实例类型可能尚未在给定区域或可用区域中可用。旧一代实例类型可能在给定区域或可用区中不可用。要搜索按位置（区域或可用区）提供的实例类型，请使用 describe-instance-type-offerings 命令。有关更多信息，请参阅亚马逊 EC2 用户指南中的查找亚马逊 EC2实例类型。

- 如果您使用专用实例或专用主机，请确保选择了作为专用实例或专用主机受支持的实例类型。

安全组 <该安全组的名称> 不存在。启动 EC2 实例失败。

原因：可能已删除启动模板或启动配置中指定的安全组。

解决方案：

1. 使用 describe-security-groups 命令获取与您的账户关联的安全组列表。
2. 从该列表中选择要使用的安全组。要创建安全组，请使用 create-security-group 命令。
3. 创建新的启动模板或启动配置。
4. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

密钥对 <与您的 EC2 实例关联的密钥对>不存在。启动 EC2 实例失败。

原因：可能已删除启动实例时使用的密钥对。

解决方案：

1. 使用 describe-key-pairs 命令获取可用的密钥对列表。
2. 从该列表中选择要使用的密钥对。要创建密钥对，请使用 create-key-pair 命令。
3. 创建新的启动模板或启动配置。
4. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

请求的实例类型（<实例类型>）在请求的可用区（<实例可用区>）中不受支持...

错误消息：您请求的<instance type>可用区 () 不支持您请求的实例类型 (<instance Availability Zone>)... 启动 EC2 实例失败。

原因：您的自动扩缩组中指定的可用区不支持您选择的实例类型。

解决方案：

1. 使用describe-instance-type-offerings命令验证哪些可用区支持您选择的实例类型，或者通过在 Amazon EC2 控制台中查看实例类型页面的网络窗格上的可用区值。

2. 使用update-auto-scaling-group命令更新或移除 Auto Scaling 组设置中任何不支持的区域的子网。有关更多信息，请参阅 添加可用区。

您的竞价请求价格 0.015 低于要求的最低竞价请求履行价格 0.0735...

原因：请求中的 Spot 最高价低于选定的实例类型的 Spot 价格。

解决方案：提交具有较高 Spot 最高价（可能是按需价格）的新请求。之前，您支付的 Spot 价格是基于出价的。今天，您支付当前 Spot 价格。通过将最高价格设置得更高，Amazon EC2 Spot 服务有更好的机会启动和维持所需的容量。

设备名称 <device name> 无效/设备名称上传无效。启动 EC2 实例失败。

原因 1：启动模板或启动配置中的块储存设备映射所包含的块储存设备名称可能无法使用或目前不受支持。

解决方案：

1. 验证哪些设备名称可用于您的特定实例配置。有关设备命名的更多详细信息，请参阅 Amazon EC2 用户指南中的 Linux 实例上的设备名称。
2. 手动创建不属于 Auto Scaling 组的 Amazon EC2 实例，然后调查问题。如果块储存设备命名配置与 Amazon Machine Image (AMI) 中的名称冲突，则实例将在启动期间失败。有关更多信息，请参阅 Amazon EC2 用户指南中的屏蔽设备映射。
3. 确认您的实例已成功启动后，使用 describe-volumes 命令查看如何向实例公开卷。
4. 使用卷描述中列出的设备名称创建新的启动模板或启动配置。
5. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

用于参数 virtualName 的值（<与实例存储设备相关联的名称>）无效... 启动 EC2 实例失败。

原因：与块储存设备相关联的虚拟名称的指定格式不正确。

解决方案：

1. 通过在 `virtualName` 参数中指定设备名称创建新的启动模板或启动配置。有关设备名称格式的信息，请参阅 Amazon EC2 用户指南中的 Linux 实例上的设备命名。
2. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

实例存储不支持 EBS 块储存设备映射。 AMIs

原因：启动模板或启动配置中指定的块储存设备映射在您的实例上不受支持。

解决方案：

1. 使用实例类型支持的块储存设备映射来创建新的启动模板或启动配置。有关更多信息，请参阅 Amazon EC2 用户指南中的屏蔽设备映射。
2. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

置放群组可能无法与类型为“<instance type>”的实例一起使用。启动 EC2 实例失败。

原因：您的集群置放群组包含无效实例类型。

解决方案：

1. 有关置放群组支持的有效实例类型的信息，请参阅 Amazon EC2 用户指南中的置放群组。
2. 按照置放群组中的详细说明创建新的置放群组。
3. 或者，也可以使用受支持的实例类型创建新的启动模板或启动配置。
4. 使用 update-auto-scaling-group 命令通过新的置放群组、启动模板或启动配置更新您的 Auto Scaling 组。

客户。 InternalError: 启动时出现客户端错误。

问题：Amazon A EC2 uto Scaling 尝试启动具有加密 EBS 卷的实例，但服务相关角色无权访问用于加密该卷的 Amazon KMS 客户托管密钥。有关更多信息，请参阅 使用加密卷所需的 Amazon KMS 密钥策略。

原因 1：您需要密钥策略，该策略向适当的服务相关角色授予使用客户托管密钥的权限。

解决方案 1：允许服务相关角色使用客户管理密钥，如下所示：

1. 确定将哪个服务相关角色用于此 Auto Scaling 组。
2. 更新客户托管密钥上的密钥策略并允许服务相关角色使用客户托管密钥。
3. 更新 Auto Scaling 组来使用服务相关角色。

有关允许服务相关角色使用客户托管密钥的密钥策略示例，请参阅 示例 1：允许访问客户托管密钥的关键策略部分。

原因 2：如果客户托管密钥和 Auto Scaling 组位于不同的 Amazon 账户中，则需要配置对客户托管密钥的跨账户访问权限，以便向适当的服务相关角色授予使用客户托管密钥的权限。

解决方案 2：允许外部账户中的服务相关角色使用本地账户中的客户托管密钥，如下所示：

1. 更新客户托管密钥上的密钥策略，以允许 Auto Scaling 组账户访问客户托管密钥。
2. 在 Auto Scaling 组账户中定义可以创建授权的 IAM 用户或角色。
3. 确定将哪个服务相关角色用于此 Auto Scaling 组。
4. 使用适当的服务相关角色作为被授权者委托人，创建对客户托管密钥的授权。
5. 更新 Auto Scaling 组来使用服务相关角色。

有关更多信息，请参阅 示例 2：允许跨账户访问客户托管密钥的关键策略部分。

解决方案 3：将同一 Amazon 账户中的客户托管密钥作为 Auto Scaling 组。

1. 使用属于与 Auto Scaling 组相同的账户中的另一个客户托管密钥，复制并重新加密快照。
2. 允许服务相关角色使用新的客户托管密钥。请参阅解决方案 1 的步骤。

我们目前在您请求的可用区中没有足够的 <实例类型> 容量。启动 EC2 实例失败。

错误消息：您请求的可用区（<请求的可用区>）中当前没有足够的 <实例类型> 容量。我们的系统将调配额外的容量。您当前可以通过不在请求中指定可用区，或者选择 <当前支持该实例类型的可用区列表> 来获取 <实例类型> 容量。启动 EC2 实例失败。

原因：目前，不支持请求的实例类型和可用区组合。

解决方案：要解决该问题，请尝试以下操作：

- 等待几分钟，让 Amazon A EC2 uto Scaling 在其他已启用的可用区域中找到该实例类型的容量。
- 将自动扩缩组扩展到其他可用区。有关更多信息，请参阅 添加可用区。
- 请遵循使用不同实例类型集的最佳实践，以便您不依赖于某一特定实例类型。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

所请求的预留没有足够的兼容容量和可用容量来满足此请求。启动 EC2实例失败。

原因 1：您已达到可通过 targeted 按需容量预留启动的实例数量限制。

解决方案 1：增加可通过 targeted 按需容量预留启动的实例数量，或者使用容量预留组，以便预留容量以外的任何容量都将作为常规按需容量启动。有关更多信息，请参阅 使用容量预留在特定可用区中预留容量 。

原因 2：您已达到可用某一容量块启动的实例数限制。

使用容量块，您将受到最初购买的容量值的限制。如果您遇到的启动次数超过预期并且可用容量耗尽，则会导致启动失败。终止实例在完全终止之前要经过漫长的清理过程。在此期间，无法重新使用它们。这也可能导致启动失败。有关更多信息，请参阅 使用 Capacity Blocks 适用于机器学习工作负载。

解决方案 2：要解决该问题，请尝试以下操作：

- 保持请求不变。如果容量块实例正在终止，则必须等待几分钟，让实例完成终止，容量才能再次可用。在容量可用之前，Amazon A EC2 uto Scaling 会继续自动提出启动请求。
- 请务必购买足够的容量来容纳峰值工作负载，这样就不会经常遇到此错误。

您的容量块预留 <reservation id> 尚未激活。启动 EC2 实例失败。

原因：指定的容量块尚未激活。

解决方案：按照推荐的容量块方法进行操作，并使用计划扩缩。这样做可以帮助您确保仅在预留处于活动状态时增加自动扩缩组的所需容量，并在预留结束之前减少所需容量。

没有与您的请求匹配的竞价容量。启动 EC2 实例失败。

原因：目前没有足够的备用容量来满足您的竞价型实例请求。

解决方案：要解决该问题，请尝试以下操作：

- 等待几分钟；容量可能经常转移。在容量可用之前，Amazon A EC2 uto Scaling 会继续自动提出启动请求。
- 将自动扩缩组扩展到其他可用区。有关更多信息，请参阅 添加可用区。
- 请遵循使用不同实例类型集的最佳实践，以便您不依赖于某一特定实例类型。有关更多信息，请参阅 Auto Scaling 组具有多个实例类型和购买选项。

已运行 <实例数量> 个实例。启动 EC2 实例失败。

原因：您已达到可在某一区域中启动的实例数限制。在您创建 Amazon 账户时，我们会为每个地区可以运行的实例数量设置默认限制。

解决方案：要解决该问题，请尝试以下操作：

- 如果您当前的限制不足以满足需求，您可以根据区域请求提高配额。有关更多信息，请参阅《亚马逊 EC2 用户指南》中的亚马逊 EC2 服务配额。
- 提交减少了实例数（可在后期阶段增加）的新请求。

对 Amazon A EC2 uto Scaling 进行故障排除：AMI 问题

本页提供与您相关的问题 AMIs、潜在原因以及为解决这些问题可以采取的步骤的信息。

要检索错误消息，请参阅检索来自扩缩活动的错误消息。

当您的 EC2 实例因您的 AMI 问题而无法启动时，您可能会收到以下一条或多条错误消息。

AMI 问题

- AMI ID <您的 AMI 的 ID> 不存在。启动 EC2 实例失败。
- AMI <AMI ID> 正在等待，无法运行。启动 EC2 实例失败。
- 设备名称 <device name> 无效。启动 EC2实例失败。
- 指定实例类型的架构 “arm64” 与指定 AMI 的架构 “x86_64” 不匹配... 启动实例失败。 EC2
- AMI“<AMI ID>”已禁用，无法运行。启动 EC2 实例失败。

> **Important**
>
> Amazon 支持通过修改 AMI 权限与其他 Amazon 账户私下共享 AMI。如果将 AMI 设为私有而不共享，则启动新实例时可能会导致授权错误。有关私密共享的更多信息 AMIs，请参阅 Amazon EC2 用户指南中的与特定 Amazon 账户共享 AMI。

AMI ID <您的 AMI 的 ID> 不存在。启动 EC2 实例失败。

- 原因：创建启动模板或启动配置后，可能已删除 AMI。

- 解决方案：
  1. 使用有效的 AMI 创建新的启动模板或启动配置。
  2. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

AMI <AMI ID> 正在等待，无法运行。启动 EC2 实例失败。

原因：您可能刚创建 AMI（通过获取运行实例的快照或任何其他方式），它可能还无法使用。

解决方案：必须等待您的 AMI 可用后，才能创建启动模板或启动配置。

设备名称 <device name> 无效。启动 EC2实例失败。

原因：将 EBS 卷连接到 EC2实例时，必须为该卷提供有效的设备名称。所选的 AMI 必须支持此设备名称。

解决方案：

1. 创建新的启动模板或启动配置并为您的 AMI 指定正确的设备名称。推荐的命名约定因 AMI 的虚拟化类型而异。有关更多信息，请参阅 Amazon EC2 用户指南中的设备名称。
2. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

指定实例类型的架构 “arm64” 与指定 AMI 的架构 “x86_64” 不匹配... 启动实例失败。 EC2

原因 1：如果 AMI 的架构与您的启动模板或启动配置中使用的实例类型不同，则当 Amazon A EC2 uto Scaling 尝试使用不兼容的实例配置启动实例时，您会收到错误。

解决方案 1：

1. 使用 desc ribe-images 命令验证您的 AMI 架构，或者通过在亚马逊 EC2 控制台中查看亚马逊系统映像 () AMIs 页面的详细信息窗格上的架构值，来验证 AM I 的架构。
2. 使用describe-instance-types命令查找与您的 AMI 具有相同架构的实例类型，或者通过在 Amazon EC2 控制台中查看 “实例类型” 屏幕上的 “架构” 列。有关选择兼容实例类型的更多信息，请参阅 Amazon EC2 用户指南中的更改实例类型的兼容性。
3. 使用与您的 AMI 具有相同架构的实例类型创建新的启动模板或启动配置。
4. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

原因 2：Amazon A EC2 uto Scaling 尝试启动在 Auto Scaling 组的混合实例策略中指定的实例类型，但该实例类型的架构与您的启动模板中指定的 AMI 不同。

解决方案 1：不要在您的混合实例策略中包含具有不同架构的实例类型。

1. 使用 desc ribe-images 命令验证您的 AMI 架构，或者通过在亚马逊 EC2 控制台中查看亚马逊系统映像 () AMIs 页面的详细信息窗格上的架构值，来验证 AM I 的架构。
2. 使用describe-instance-types命令或在 Amazon EC2 控制台中查看实例类型屏幕上的 “架构” 列，验证您打算包含在混合实例策略中的每种实例类型的架构。有关选择兼容实例类型的更多信息，请参阅 Amazon EC2 用户指南中的更改实例类型的兼容性。
3. 使用update-auto-scaling-group命令从 Auto Scaling 组中更新或移除不兼容的实例类型。

解决方案 2：要在同一自动扩缩组中同时启动 Arm (Graviton2) 和 x86_64（英特尔）实例，您必须分别使用兼容 ARM 的 AMI 和兼容英特尔 x86 的 AMI 所支持的启动模板来匹配混合实例策略中的实例类型。

1. 使用 desc ribe-images 命令在现有启动模板中验证 AMI 的架构，或者在亚马逊 EC2 控制台中查看 Amazon 系统映像 () AMIs 页面详细信息窗格上的架构值。
2. 使用与您打算使用的其他架构相匹配的 AMI 来创建新的启动模板。
3. 更新您的 Auto Scaling 组以覆盖现有启动模板，并使用update-auto-scaling-group命令为每种兼容的实例类型指定新的启动模板。有关更多信息，请参阅 为实例类型使作不同的启动模板。

AMI“<AMI ID>”已禁用，无法运行。启动 EC2 实例失败。

原因：您正试图从已禁用的 AMI 启动实例。有关更多信息，请参阅 Amazon EC2 用户指南中的禁用 AMI。

解决方案：

1. 创建新的启动模板或启动配置，并指定未禁用的 AMI。
2. 使用 update-auto-scaling-group 命令通过新的启动模板或启动配置更新您的 Auto Scaling 组。

Amazon A EC2 uto Scaling 疑难解答：负载均衡器问题

本页提供与 Auto Scaling 组相关联的负载均衡器所导致的问题有关的信息、可能原因，以及可用来解决这些问题的步骤。

要检索错误消息，请参阅检索来自扩缩活动的错误消息。

当您的 EC2 实例由于与 Auto Scaling 组关联的负载均衡器出现问题而无法启动时，您可能会收到以下一条或多条错误消息。

负载均衡器问题

- 一个或多个目标组未找到。验证负载均衡器配置失败。
- 找不到负载均衡器 <your load balancer>。验证负载均衡器配置失败。
- 名为 <负载均衡器名称> 的活动负载均衡器不存在。更新负载均衡器配置失败。
- EC2 实例<instance ID>不在 VPC 中。更新负载均衡器配置失败。

一个或多个目标组未找到。验证负载均衡器配置失败。

问题：当您的 Auto Scaling 组启动实例时，Amazon A EC2 uto Scaling 会尝试验证与 Auto Scaling 组关联的 Elastic Load Balancing 资源是否存在。当找不到目标组时，扩展活动会失败，您会收到 `One or more target groups not found. Validating load balancer configuration failed.` 错误消息。

原因 1：关联到您的自动扩缩组的目标组已被删除。

解决方案 1：您可以创建一个没有目标组的新 Auto Scaling 组，也可以使用 Amazon Auto Scaling 控制台或 detach-load-balancer-target-groups 命令将未使用的目标组从 A EC2 uto Scaling 组中移除。

原因 2：目标组存在，但是创建自动扩缩组时，在尝试指定目标组 ARN 时出现问题。资源的创建顺序不正确。

解决方案 2：创建新的自动扩缩组，并在最后指定目标组名称。

找不到负载均衡器 <your load balancer>。验证负载均衡器配置失败。

问题：当您的 Auto Scaling 组启动实例时，Amazon A EC2 uto Scaling 会尝试验证与 Auto Scaling 组关联的 Elastic Load Balancing 资源是否存在。当找不到经典负载均衡器时，扩展活动会失败，您会收到 `Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed.` 错误消息。

原因 1：已删除经典负载均衡器。

解决方案 1：您可以创建一个没有负载均衡器的新 Auto Scaling 组，也可以使用 Amazon Auto Scaling 控制台或detach-load-balancers命令将未使用的负载均衡器从 A EC2 uto Scaling 组中移除。

原因 2：经典负载均衡器存在，但是创建自动扩缩组时，在尝试指定负载均衡器名称时出现问题。资源的创建顺序不正确。

解决方案 2：创建新的 Auto Scaling 组，并在最后指定负载均衡器名称。

名为 <负载均衡器名称> 的活动负载均衡器不存在。更新负载均衡器配置失败。

原因：可能已删除指定的负载均衡器。

解决方案：可以创建新的负载均衡器，然后创建新的 Auto Scaling 组，也可以创建无负载均衡器的新 Auto Scaling 组。

EC2 实例<instance ID>不在 VPC 中。更新负载均衡器配置失败。

原因：VPC 中不存在指定的实例。

解决方案：可以删除与实例相关联的负载均衡器，或者创建新的 Auto Scaling 组。

对 Amazon A EC2 uto Scaling 进行故障排除：启动模板

使用以下信息可帮助您诊断和修复在尝试为自动扩缩组指定启动模板时可能遇到的常见问题。

无法启动实例

如果您无法使用已指定的启动模板启动任何实例，请检查以下一般故障排除内容：对 Amazon A EC2 uto Scaling 进行故障排除： EC2 实例启动失败。

您必须使用有效的完整启动模板（无效值）

问题：当您尝试为某个自动扩缩组指定启动模板时，收到 `You must use a valid fully-formed launch template` 错误。您可能遇到此错误，因为只有在使用启动模板的自动扩缩组被创建或更新时，才会对启动模板中的值进行验证。

原因 1：如果您收到`You must use a valid fully-formed launch template`错误，则存在一些问题会导致 Amazon A EC2 uto Scaling 认为启动模板的某些内容无效。这是一个通用错误，可能由几种不同的原因导致。

解决方案 1：请尝试以下步骤进行故障排除：

1. 请注意错误消息的第二部分以查找更多信息。在遇到 `You must use a valid fully-formed launch template` 错误后，请参阅更具体的错误消息，该消息标识您需要解决的问题。
2. 如果您无法找到原因，请使用 run-instances 命令测试您的启动模板。使用 `--dry-run` 选项，如以下示例中所示。这使您可以重现问题并提供有关问题原因的见解。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

3. 如果值无效，请验证指定的资源是否存在且正确。例如，当您指定 Amazon EC2 密钥对时，该资源必须存在于您的账户以及您创建或更新 Auto Scaling 组的区域中。
4. 如果缺少预期信息，请验证您的设置并根据需要调整启动模板。
5. 进行更改后，使用 `--dry-run` 选项重新运行 run-instances 命令，以验证您的启动模板是否使用有效的值。

有关更多信息，请参阅 为 Auto Scaling 组创建启动模板。

您没有权限使用启动模板（权限不足）

问题：当您尝试为某个自动扩缩组指定启动模板时，收到 `You are not authorized to use launch template` 错误。

原因 1：如果您尝试使用启动模板，但您使用的 IAM 凭证没有足够的权限，则会收到一条错误，显示您没有权限使用启动模板。

解决方案 1：要解决该问题，请尝试以下操作：

- 验证您用于发出请求的 IAM 凭证是否有权调用所需的 EC2 API 操作，包括`ec2:RunInstances`操作。如果在启动模板中指定了任何标签，您还必须拥有使用 `ec2:CreateTags` 操作的权限。
- 也可以验证您用于发出请求的 IAM 凭证被分配了 `AmazonEC2FullAccess` 策略。该 Amazon 托管策略授予对所有亚马逊 EC2 资源和相关服务的完全访问权限，包括 Amazon A EC2 uto Scaling 和 Elastic Load Balancing。 CloudWatch

有关使用启动模板所需权限的更多信息，包括 IAM 策略示例，请参阅 A mazon EC2 用户指南中的使用 IAM 权限控制启动模板的访问权限。有关其他示例 IAM policies，请参阅 在 Auto Scaling 群组中控制亚马逊 EC2 启动模板的使用情况。

原因 2：如果您尝试使用指定实例配置文件的启动模板，则必须具有传递与实例配置文件关联的 IAM 角色的 IAM 权限。

解决方案 2：验证您用于发出请求的 IAM 证书是否具有将指定角色传递给 Amazon A EC2 uto Scaling 服务的正确`iam:PassRole`权限。有关更多信息和示例 IAM policy，请参阅 适用于在 Amazon EC2 实例上运行的应用程序的 IAM 角色。有关与实例配置文件相关的更多疑难解答主题，请参阅 IAM 用户指南中的 Amazon EC2 和 IAM 故障排除。

原因 3：如果您尝试使用在另一个 AMI 中指定了 AMI 的启动模板 Amazon Web Services 账户，并且该 AMI 是私有的，不与 Amazon Web Services 账户 您正在使用的模板共享，则会收到一条错误消息，提示您无权使用该启动模板。

解决方案 3：验证 AMI 的权限是否包括您正在使用的账户。有关更多信息，请参阅《亚马逊 EC2 用户指南》 Amazon Web Services 账户中的与特定用户共享 AMI。

相关信息

下列相关资源在您使用此服务的过程中会有所帮助。

资源	描述
中国入门中的 Amazon A EC2 uto S Amazon caling	中国的 Amazon A EC2 uto Scaling 文档以 Amazon 公开文档为基础。此文档中描述的服务或功能可能因区域而异。要查看适用于中国地区的差异，请参阅中国入门 Amazon 中的 EC2 Amazon Auto Scaling 部分。
亚马逊 A EC2 uto Scaling API 参考	每个 API 操作的文档都说明了请求参数和 XML 响应，并提供了特定语言的 SDK 参考主题链接。
Amazon CLI 命令参考中的autoscaling	描述可用于处理 Auto Scaling 组的 Amazon CLI 命令。
Amazon Tools for PowerShell Cmdlet 参考资料	这些 Amazon 工具 PowerShell 使您能够通过 PowerShell 命令行编写 Amazon 资源操作脚本。
使用 Amazon CloudFormation创建 Auto Scaling 组	使用资源，您无需手动操作即可构建、建模和管理 Auto Scaling 群组。
中国@@ 亚马逊网络服务入门中的终端节点和 ARNs 中国亚马逊网络服务入门	有关 Amazon A EC2 uto Scaling 区域和终端节点的信息。
在亚马逊 EC2 用户指南中创建 AMI	了解怎样从一个自定义的实例中创建一个 Amazon Machine Image (AMI)。
在亚马逊 EC2 用户指南中@@ 连接您的 Linux 实例	了解如何连接到您启动的 Linux 实例。
在亚马逊 EC2 用户指南中@@ 连接到你的 Windows 实例	了解如何连接到您启动的 Windows 实例。

资源	描述
在《亚马逊 CloudWatch 用户指南》中@@ 创建账单警报以监控您的预估 Amazon 费用	了解如何使用监控您的预估费用 CloudWatch。
Application Auto Scaling 用户指南	了解如何为亚马逊以外的亚马逊 Web Services 的可扩展资源配置自动扩展 EC2。

文档历史记录

下表描述了从 2018 年 7 月开始对 Amazon A EC2 uto Scaling 文档进行的重要补充。如需对此文档更新的通知，您可以订阅 RSS 源。

变更	说明	日期
高分辨率指标	目标跟踪现在支持具有秒级数据点的高分辨率 CloudWatch 指标，这些数据点的发布间隔小于一分钟。有关更多信息，请参阅使用高分辨率指标创建目标跟踪策略以加快响应速度。	2024 年 11 月 22 日
安全 IAM 更新	AutoScalingService RolePolicy托管策略现在向 Resource Groups 授予额外权限`resource-groups:ListGroupResources` 。	2024 年 11 月 20 日
性能保护	在 Auto Scaling 组中使用基于属性的实例类型选择时，您现在可以启用性能保护以确保所选实例类型与指定的性能基准相似或超过指定的性能基准。有关更多信息，请参阅使用基于属性的实例类型选择创建混合实例组。	2024 年 11 月 20 日
容量预留首选项	现在，您可以优先启动容量预留中的实例。有关更多信息，请参阅容量预留。	2024 年 11 月 20 日
可用区转移	现在，您可以使用区域切换功能从可用区域中的应用程序损坏中恢复过来。有关更多信	2024 年 11 月 18 日

	息，请参阅 Auto Scaling 组的区域偏移。	
可用区分布	现在，您可以为 Auto Scaling 组选择可用区域分配。有关更多信息，请参阅 Auto Scaling 组可用区域分布。	2024 年 11 月 7 日
安全 IAM 更新	AutoScalingServiceRolePolicy托管策略现在向 Amazon EC2（`ec2:GetSecurityGroupsForVpc` 和`ec2:GetInstanceTypesFromInstanceRequirements` ）授予额外权限。	2024 年 2 月 29 日
额外支持温水池休眠 Amazon Web Services 区域	现在，您可以在另外两个区域的温池中休眠实例： Amazon GovCloud （美国东部）和 Amazon GovCloud （美国西部）。有关温池的更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的适用于 Amazon A EC2 uto Scaling 的温池。	2024 年 2 月 26 日
额外支持温水池休眠 Amazon Web Services 区域	您现在可以在另外两个区域的暖池中休眠实例：欧洲（苏黎世）和中东（阿联酋）。有关温池的更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的适用于 Amazon A EC2 uto Scaling 的温池。	2024 年 2 月 21 日

支持跨账户参数使用	现在，您可以使用 Amazon Web Services 账户 与 Amazon A EC2 uto Scaling 共享的 Amazon Systems Manager 参数。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的在启动模板中使用 Amazon Systems Manager 参数代替 AMI IDs 。	2024 年 2 月 21 日
新的竞价价格保护选项	现在，当您使用基于属性的实例类型选择时，您可以将竞价型实例的价格保护阈值定义为按需价格的百分比。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的价格保护。	2024 年 1 月 29 日
实例维护策略	现在，您可以使用实例维护策略来定义在导致实例被替换的事件(包括实例刷新)发生期间终止现有实例之前还是之后启动实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的实例维护政策。	2023 年 11 月 15 日
适用于 ML 的容量块	现在，您可以在创建启动模板时指定容量块预留 ID，从而将实例启动到容量块中。通过容量块允许您在未来某个日期预留 GPU 实例，从而支持您的短期机器学习（ML）工作负载。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的将容量块用于机器学习工作负载。	2023 年 10 月 31 日

新的实例刷新功能	现在，您可以将实例刷新配置为将其状态设置为失败，并且在检测到指定的 CloudWatch 警报已进入ALARM状态时可以选择回滚。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用回滚撤消更改。	2023 年 7 月 31 日
指南更改	指南中增加了关于将按需型实例启动到容量预留中的新主题。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用按需容量预留在特定可用区域中预留容量。	2023 年 7 月 28 日
指南更改	指南中添加了一个关于将 Amazon CloudFormation 堆栈从启动配置迁移到启动模板的新主题。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的将Amazon CloudFormation 堆栈从启动配置迁移到启动模板。	2023 年 4 月 18 日

支持新的 API 操作	此版本添加了三个新的 API 操作：AttachTrafficSources 、DetachTrafficSources 、和 DescribeTrafficSources 。此外，新的字段 TrafficSources 已添加至 DescribeAutoScalingGroups 操作的结果中。DescribeScalingActivities 操作结果中添加了新的活动状态 WaitingForConnectionDraining 。Amazon A EC2 uto Scaling 还为VPC_LATTICE CreateAutoScalingGroup UpdateAutoScalingGroup 、和DescribeAutoScalingGroups 操作中的HealthCheckType 字段支持新值。有关更多信息，请参阅 Amazon A EC2 uto Scaling API 参考。	2023 年 3 月 31 日
支持 Amazon VPC Lattice	这是适用于 Amazon A EC2 uto Scaling 的 VPC Lattice 的正式发布版本。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的使用 VPC Lattice 目标组将流量路由到您的 A EC2 uto Scaling 组。	2023 年 3 月 31 日

指南更改	带有使用 Elastic Load Balancing Amazon CLI 示例的部分现在包括新的和更新的示例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用 Amazon Command Line Interface (Amazon CLI) 使用 Elastic Load Balancing 的示例。	2023 年 3 月 31 日
Support 还支持预测性扩展 Amazon Web Services 区域	现在，您可以在中东（阿联酋）和 Amazon GovCloud（美国东部）地区创建预测性扩展策略。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。	2023 年 3 月 16 日
新的实例刷新功能	现在，您可以选择终止或忽略待机实例并替换或忽略横向缩减保护实例，而不必等待它们变为可替换实例。您也可以回滚因实例刷新失败而产生的更改。作为本次更新的一部分，文档已扩展到包括回滚实例刷新、取消实例刷新以及了解实例刷新可配置参数的默认值等主题。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的根据实例刷新替换 A EC2 uto Scaling 实例。	2023 年 2 月 10 日

支持使用 Amazon Systems Manager 参数作为 AMI ID	您现在可以在启动模板中使用 Systems Manager 参数代替 AMI ID。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的在启动模板中使用 Amazon Systems Manager 参数代替 AMI IDs 。	2023 年 1 月 19 日
预测性扩展建议	现在，您可以从 Amazon A EC2 uto Scaling 控制台获取有关评估和选择正确的预测性扩展策略的建议。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的评估您的预测性扩展策略。	2023 年 1 月 18 日
预测式扩展预测	预测式扩展生成的预测现在每六小时更新一次，而不是每天更新一次。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。	2023 年 1 月 6 日
Support 支持 CloudWatch 公制数学	创建目标跟踪扩展策略时，您现在可使用指标数学。使用指标数学，您可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的时间序列。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的使用指标数学为 Amazon A EC2 uto Scaling 创建目标跟踪扩展策略。	2022 年 12 月 8 日

IAM 服务相关角色权限更新	该AutoScalingServiceRolePolicy 政策现在向 Amazon A EC2 uto Scaling 授予了额外权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling Amazon 托管策略。	2022 年 12 月 6 日
新的竞价型分配策略	您现在可以使用价格和容量优化分配策略从中断可能性最小、价格尽可能最低的竞价型池中请求竞价型实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的分配策略。	2022 年 11 月 10 日
在亚太地区（雅加达）区域支持预测性扩缩	您现在可以在亚太地区（雅加达）区域创建预测性扩缩策略。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。	2022 年 10 月 13 日
支持在控股台中将自定义指标用于预测性扩缩	现在，您可以在通过 Amazon A EC2 uto Scaling 控制台创建预测性扩展策略时使用自定义指标。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。	2022 年 10 月 13 日

CloudWatch 监控预测性扩展指标	现在，您可以使用访问监控数据以进行预测性扩展 CloudWatch。从而可以使用指标数学来创建可显示预测数据准确性的新时间序列。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南 CloudWatch中的使用监控预测性扩展指标。	2022 年 7 月 7 日
在亚太地区（大阪）区域支持预测性扩缩	您现在可以在亚太地区（大阪）区域创建预测性扩缩策略。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。	2022 年 7 月 6 日
在更多区域支持暖池休眠	您现在可以在另外四个区域的暖池中进行实例休眠：非洲（开普敦）、亚太地区（雅加达）、亚太地区（大阪）和欧洲地区（米兰）。有关温池的更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的适用于 Amazon A EC2 uto Scaling 的温池。	2022 年 7 月 5 日
运行状况检查更新	在执行运行状况检查时，Amazon A EC2 uto Scaling 现在可以帮助您最大限度地减少由于临时问题或配置错误的运行状况检查而可能发生的停机时间。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon Aut EC2 o Sc aling 如何最大限度地减少停机时间。	2022 年 5 月 21 日

原定设置实例预热	您现在可以统一自动扩缩组的所有预热和冷却设置，并通过启用默认实例预热来优化持续扩缩的扩缩策略的性能。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的为 Auto Scaling 组设置默认实例预热。	2022 年 4 月 19 日
指南更改	本指南中增加了关于与其他 Amazon 服务集成的新章节。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的与 Amazon A EC2 uto Scaling 集成的Amazon 服务。	2022 年 3 月 29 日
IAM 服务相关角色权限更新	现在，该`AutoScalingServiceRolePolicy` 策略向 Amazon A EC2 uto Scaling 授予了额外的读取权限。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling Amazon 托管策略。	2022 年 3 月 28 日
实例元数据提供目标生命周期状态	您可以从实例元数据检索 Auto Scaling 实例的目标生命周期状态。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的通过实例元数据检索目标生命周期状态。	2022 年 3 月 24 日

对新的暖池功能的支持	现在，您可以将暖池中的实例休眠以停止实例，而无需删除其内存内容 (RAM)。现在，您还可以在横向缩减时使实例返回到暖池，而不是总是终止未来需要的实例容量。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的适用于 Amazon A EC2 uto Scaling 的温池。	2022 年 2 月 24 日
指南更改	Amazon A EC2 uto Scaling 控制台已更新，增加了更多选项，可帮助您在启用跳过匹配并指定所需配置的情况下开始实例刷新。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的开始或取消实例刷新（控制台）。	2022 年 2 月 3 日
预测性扩展策略的自定义指标	您现在可以选择是否在创建预测性扩展策略时使用自定义指标。您还可以使用指标数学来进一步自定义策略中包含的指标。有关更多信息，请参阅 使用自定义指标进行高级预测性扩展配置。	2021 年 11 月 24 日
新的按需分配策略	现在，创建使用混合实例策略的 Auto Scaling 组时，您可以选择是否根据价格（最低价格的实例类型优先）启动按需型实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的分配策略。	2021 年 10 月 27 日

基于属性的实例类型选择	Amazon A EC2 uto Scaling 增加了对基于属性的实例类型选择的支持。您可以将实例要求表达为一组属性，例如 vCPU、内存和存储，而不是手动选择实例类型。有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的使用基于属性的实例类型选择创建 A EC2 uto Scaling 组。	2021 年 10 月 27 日
支持按标签筛选组	现在，当您使用 `describe-auto-scaling-groups` 命令检索有关 Auto Scaling 组的信息时，可以使用标签筛选条件筛选 Auto Scaling 组。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的使用标签筛选 A EC2 uto Scaling 组。	2021 年 10 月 14 日
指南更改	Amazon A EC2 uto Scaling 控制台已更新，可帮助您使用创建自定义终止策略 Amazon Lambda。控制台文档已相应地进行了修订。有关更多信息，请参阅使用不同的终止策略（控制台）。	2021 年 10 月 14 日
支持将启动配置复制到启动模板	现在，您可以从 Amazon A EC2 uto Scaling 控制台将一个 Amazon 区域中的所有启动配置复制到新的启动模板中。	2021 年 8 月 9 日

扩展实例刷新功能	现在，您可以在通过将所需的配置添加到 `start-instance-refresh` 命令来替换实例时包含更新（例如新版本的启动模板）。您也可以通过启用跳过匹配来跳过替换已包含所需配置的实例。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的根据实例刷新替换 A EC2 uto Scaling 实例。	2021 年 8 月 5 日
支持自定义终止策略	现在，您可以使用创建自定义终止政策 Amazon Lambda。有关更多信息，请参阅使用 Lambda 创建自定义终止策略。用于指定终止策略的文档进行了相应的更新。	2021 年 7 月 29 日
指南更改	Amazon A EC2 uto Scaling 控制台已更新和增强，增加了其他功能，可帮助您创建指定时区的计划操作。适用于计划扩展的文档已进行相应的修订。	2021 年 6 月 3 日
启动配置中的 gp3 卷	您现在可以在块储存设备映射中为启动配置指定 gp3 卷。	2021 年 6 月 2 日

支持预测性扩展	现在，您可以使用预测性扩展，通过扩展策略主动扩展您的 Amazon A EC2 uto Scaling 群组。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 预测性扩展。通过此更新，AutoScalingServiceRolePolicy托管策略现在包括调用 `cloudwatch:GetMetricData` API 操作的权限。	2021 年 5 月 19 日
指南更改	现在，您可以从中访问生命周期挂钩的示例模板 GitHub。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 生命周期挂钩。	2021 年 4 月 9 日
支持温水池	现在，您可以通过向 Auto Scaling 组添加温水池来平衡首次启动时间较长的应用程序的性能（最大限度地减少冷启动）和成本（停止过度配置实例容量）。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的适用于 Amazon A EC2 uto Scaling 的温池。	2021 年 4 月 8 日
支持检查点	现在，您可以将检查点添加到实例刷新中，以分阶段替换实例并在特定时间点对您的实例执行验证。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的向实例刷新添加检查点。	2021 年 3 月 18 日

指南更改	改进了用于处理 Amazon EventBridge A EC2 uto Scaling 事件和生命周期挂钩的文档。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的搭配使用 Amazon Auto Scalin g EventBridge EC2 和教程：配置调用 Lambda 函数的生命周期挂钩。	2021 年 3 月 18 日
支持本地时区	现在，您可以在本地时区中通过将 `--time-zone` 选项添加到 put-scheduled-update-group-action 命令来创建重复计划操作。如果您的时区遵守夏令时 (DST)，则重复操作会自动调整夏令时。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的计划扩展。	2021 年 3 月 9 日
扩展混合实例策略的功能	现在，当您使用混合实例策略时，您可以为竞价型容量设置实例类型的优先级。Amazon A EC2 uto Scaling 尝试尽最大努力完成优先级，但首先会针对容量进行优化。有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的具有多种实例类型和购买选项的 A EC2 uto Scaling 群组。	2021 年 3 月 8 日

已删除组的扩展活动	现在，您可以通过将 `--include-deleted-groups` 选项添加到 describe-scaling-activities 命令来查看已删除 Auto Scaling 组的扩展活动。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 疑难解答。	2021 年 2 月 23 日
控制台改进	现在，您可以从 Amazon A EC2 uto Scaling 控制台创建和连接应用程序负载均衡器或网络负载均衡器。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的创建和连接新的应用程序负载均衡器或网络负载均衡器（控制台）。	2020 年 11 月 24 日
多个网络接口	现在，您可以为指定多个网络接口的 Auto Scaling 组配置启动模板。有关更多信息，请参阅 VPC 中的网络接口。	2020 年 11 月 23 日
多个启动模板	多个启动模板现在可以与 Auto Scaling 组一起使用。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的为实例类型指定不同的启动模板。	2020 年 11 月 19 日

Gateway Load Balancer	更新了指南，展示了如何将网关负载均衡器连接到 Auto Scaling 组，以确保自动注册由 Amazon A EC2 uto Scaling 启动的设备实例并从负载均衡器中注销。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的弹性负载平衡类型和将负载均衡器附加到 A EC2 uto Scaling 组。	2020 年 11 月 10 日
最大实例生命周期	现在，您可以将最大实例生命周期减少到一天（86,400 秒）。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的基于最长实例生命周期替换 A EC2 uto Scaling 实例。	2020 年 11 月 9 日
容量再平衡	您可以将 Auto Scaling 组配置为在亚马逊 EC2 发出再平衡建议时启动替换竞价型实例。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 容量再平衡。	2020 年 11 月 4 日
实例元数据服务版本 2	使用启动配置时，您可能需要使用实例元数据服务版本 2，这是一种面向会话的方法，用于请求实例元数据。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的配置实例元数据选项。	2020 年 7 月 28 日

指南更改	A mazon EC2 Auto Scaling 用户指南的 “控制哪些 Auto Scaling 实例在缩容期间终止”、“监控您的 Auto Scaling 实例和组”、“启动模板” 和 “启动配置” 部分中进行了各种改进和新的控制台程序。	2020 年 7 月 28 日
实例刷新	在更改配置时启动实例刷新，可以更新 Auto Scaling 组中的所有实例。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的根据实例刷新替换 A EC2 uto Scaling 实例。	2020 年 6 月 16 日
指南更改	《Amazon A uto Scaling 用户指南》的 “根据最长实例寿命替换 A uto Scaling 实例”、“使用多种实例类型和购买选项替换 Auto S caling 实例”、“基于 Amazon SQS 进行扩展” 以及为 Amazon Auto Scalin g EC2 组和实例添加标签部分进行了各种改进。	2020 年 5 月 6 日
指南更改	IAM 文档的各种改进。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的启动模板支持和基于身份的 Amazon A EC2 uto Scaling 策略示例。	2020 年 3 月 4 日

禁用扩缩策略	您现在可以禁用和重新启用扩展策略。使用此功能，您可以在临时禁用扩展策略的同时保留配置详细信息，这样以后便可以再次启用该策略。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的禁用 A EC2 uto Scaling 组的扩展策略。	2020 年 2 月 18 日
增加了通知功能	现在，当您的 A EC2 uto Scaling 组由于缺少安全组或启动模板而无法扩展 Amazon Health Dashboard 时，Amazon Auto Scaling 会向您发送事件。有关更多信息，请参阅《亚马逊 A EC2 uto Scaling 用户指南》中的 Amazon A EC2 uto Scaling Amazon Health Dashboard 通知。	2020 年 2 月 12 日
指南更改	Amazon Auto Scaling 用户指南的 “A mazon A EC2 uto Scaling 如何与 IAM EC2 配合使用”、“Amazon Auto Scaling 基于身份的策略示例”、“用于加密卷的必需 CMK 密钥策略” 以及 “监控您的自动缩放实例和组” 部分进行了各种改进和更正。 EC2	2020 年 2 月 10 日

指南更改	改进了使用实例权重的 Auto Scaling 组的文档。了解如何在使用“容量单位”衡量所需容量时使用扩展策略。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的扩展策略的工作原理和扩展调整类型。	2020 年 2 月 6 日
新增“安全性”章节	Amazon A EC2 uto Scaling 用户指南作为本次更新的一部分，用户指南中的 “控制对您的 Amazon A EC2 uto Scaling 资源的访问权限” 一章已被一个新的、更有用的部分 “Amazon A EC2 uto Scaling 的身份和访问管理” 所取代。	2020 年 2 月 4 日
实例类型建议	Amazon Compute Optimizer 提供 Amazon EC2 实例建议，以帮助您提高性能、节省资金或两者兼而有之。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的获取实例类型的建议。	2019 年 12 月 3 日
专用主机和主机资源组	已更新演示如何创建指定主机资源组的启动模板的指南。这样，您可以使用启动模板创建 Auto Scaling 组，该模板指定要在专用主机上使用的 BYOL AMI。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 Aut EC2 o S caling 组创建启动模板。	2019 年 12 月 3 日

对 Amazon VPC 终端节点的支持	您现在可以在您的 VPC 和 Amazon A EC2 uto Scaling 之间建立私有连接。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的 Amazon A EC2 uto Scaling 和接口 VPC 终端节点。	2019 年 11 月 22 日
最大实例生命周期	现在，您可以通过指定实例可以使用的最长时间来自动替换实例。如果有任何实例接近此限制，Amazon A EC2 uto Scaling 会逐渐取代它们。有关更多信息，请参阅 Amazon A uto Scaling 用户指南中的基于最长实例生命周期替换 A EC2 uto Scaling 实例。	2019 年 11 月 19 日
实例权重	对于具有多个实例类型的 Auto Scaling 组，您现在可以选择指定每个实例类型对组容量贡献的容量单位数。有关更多信息，请参阅 Amazon Auto Scalin g 用户指南中的 Amazon A EC2 ut EC2 o Sc aling 的实例权重。	2019 年 11 月 19 日
最小实例类型数量	您不再需要为竞价型实例、按需型实例和预留实例组指定其他实例类型。对于所有 Auto Scaling 组，最小值现在为一个实例类型。有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的具有多种实例类型和购买选项的 A EC2 uto Scaling 群组。	2019 年 9 月 16 日

支持新的竞价型分配策略	Amazon A EC2 uto Scaling 现在支持 “容量优化” 的新竞价分配策略，该策略使用根据可用竞价容量进行最佳选择的竞价型实例池来满足您的请求。有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的具有多种实例类型和购买选项的 A EC2 uto Scaling 群组。	2019 年 8 月 12 日
指南更改	改进了服务相关角色中的 Amazon A EC2 uto Scaling 文档，以及用于加密卷主题的必需 CMK 密钥策略。	2019 年 8 月 1 日
支持标记增强功能	现在，Amazon A EC2 uto Scaling 会向亚马逊 EC2 实例添加标签，这是启动这些实例的同一 API 调用的一部分。有关更多信息，请参阅标记 Auto Scaling 组和实例。	2019 年 7 月 26 日
指南更改	改进了 “暂停和恢复扩展流程” 主题中的 Amazon A EC2 uto Scaling 文档。更新了客户托管策略示例，增加了允许用户仅将特定的自定义后缀服务相关角色传递给 Amazon A EC2 uto Scaling 的示例策略。	2019 年 6 月 13 日
支持新的 Amazon EBS 功能	在启动模板主题中增加了对新的 Amazon EBS 功能的支持。在从快照还原时更改 EBS 卷的加密状态。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的为 Aut EC2 o S caling 组创建启动模板。	2019 年 5 月 13 日

指南更改	改进了以下部分中的 Amazon A EC2 uto Scaling 文档：控制哪些自动缩放实例在缩容期间终止、自动缩放组、具有多种实例类型和购买选项的 Auto Scaling 组，以及适用于 Amazon A EC2 uto Scaling 的动态扩展。	2019 年 3 月 12 日
支持组合实例类型和购买选项	在单个 Auto Scaling 组中通过多个购买选项（竞价型、按需型和预留实例）和实例类型预置和自动扩展实例。有关更多信息，请参阅 A mazon Auto Scaling 用户指南中的具有多种实例类型和购买选项的 A EC2 uto Scaling 群组。	2018 年 11 月 13 日
更新了有关基于 Amazon SQS 扩展的主题	更新了指南，介绍如何使用自定义指标，根据 Amazon SQS 队列中的需求变化来扩展 Auto Scaling 组。有关更多信息，请参阅 Amazon A EC2 uto Scaling 用户指南中的基于亚马逊 SQS 进行扩展。	2018 年 7 月 26 日

下表描述了 2018 年 7 月之前 Amazon A EC2 uto Scaling 文档的重要更改。

特征	描述	发行日期
对目标跟踪扩展策略的支持	只需几个步骤就可为应用程序设置动态扩展。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的目标跟踪扩展政策。	2017 年 7 月 12 日

特征	描述	发行日期
支持资源级权限	创建 IAM 策略以控制资源级访问。有关更多信息，请参阅控制对您的 Amazon A EC2 uto Scaling 资源的访问权限。	2017 年 5 月 15 日
监控改进	Auto Scaling 组指标不再需要您启用详细监控。您现在可以启用组指标集合和从控制台中的 Monitoring (监控) 选项卡上查看指标图形。有关更多信息，请参阅使用 Amazon 监控您的 Auto Scaling 群组和实例 CloudWatch。	2016 年 8 月 18 日
支持 Application Load Balancer	将一个或多个目标组附加到新的或现有的 Auto Scaling 组。有关更多信息，请参阅将负载均衡器附加到 Auto Scaling 组。	2016 年 8 月 11 日
生命周期挂钩的事件	Amazon A EC2 uto Scaling 会在调用生命周期挂钩 EventBridge 时向其发送事件。有关更多信息，请参阅获取 EventBridge Auto Scaling 组缩放的时间。	2016 年 2 月 24 日
实例保护	在缩减规模时，阻止 Amazon A EC2 uto Scaling 选择要终止的特定实例。有关更多信息，请参阅实例保护。	2015 年 12 月 07 日
分步扩展策略	创建扩展策略，使您能根据警报的严重程度来扩展。有关更多信息，请参阅扩展策略类型。	2015 年 7 月 6 日
更新负载均衡器	将负载均衡器附加到现有 Auto Scaling 组或从中分离。有关更多信息，请参阅将负载均衡器附加到 Auto Scaling 组。	2015 年 6 月 11 日
Support ClassicLink	将 Auto Scaling 组中的 EC2经典实例链接到 VPC，从而使用私有 IP 地址在这些关联的 EC2经典实例与 VPC 中的实例之间实现通信。有关更多信息，请参阅将 EC2经典实例关联到 VPC。	2015 年 1 月 19 日
生命周期钩子	在您对新启动的实例或正在终止的实例执行操作时，请将这些实例保持在等待状态。有关更多信息，请参阅 Amazon A EC2 uto Scaling 生命周期挂钩。	2014 年 7 月 30 日

特征	描述	发行日期
分离实例	从 Auto Scaling 组分离实例。有关更多信息，请参阅从 Auto Scaling 组中分离 EC2 实例。	2014 年 7 月 30 日
将实例置于 Standby 状态	将处于 `InService` 状态的实例置于 `Standby` 状态。有关更多信息，请参阅从 Auto Scaling 组临时删除实例。	2014 年 7 月 30 日
管理标签	使用 Amazon Web Services Management Console管理您的 Auto Scaling 组。有关更多信息，请参阅标记 Auto Scaling 组和实例。	2014 年 5 月 01 日
支持专用实例	通过在创建启动配置时指定部署租期属性启动专用实例。有关更多信息，请参阅实例部署租期。	2014 年 4 月 23 日
创建群组或从 EC2 实例启动配置	使用 EC2 实例创建 Auto Scaling 组或启动配置。有关使用 EC2 实例创建启动配置的信息，请参阅使用 EC2 实例创建启动配置。有关使用实例创建 Auto Scaling 组的信息，请参阅使用 EC2 实例创建 Auto Scaling 组。 EC2	2014 年 1 月 02 日
附加实例	通过将 EC2 实例附加到现有 Auto Scaling 组，为该实例启用自动扩展。有关更多信息，请参阅将 EC2 实例附加到您的 Auto Scaling 组。	2014 年 1 月 02 日
查看账户限制	查看对您的账户的 Auto Scaling 资源限制。有关更多信息，请参阅 Auto Scaling 资源和组的配额。	2014 年 1 月 02 日
控制台支持 Amazon A EC2 uto Scaling	使用访问 Amazon A EC2 uto Scaling Amazon Web Services Management Console。有关更多信息，请参阅 Amazon A EC2 uto Scaling 入门。	2013 年 12 月 10 日
分配公有 IP 地址	将公有 IP 地址分配给启动至 VPC 的实例。有关更多信息，请参阅在 VPC 中启动 Auto Scaling 实例。	2013 年 9 月 19 日
实例终止策略	指定实例终止策略，让 Amazon A EC2 uto Scaling 在终止EC2实例时使用。有关更多信息，请参阅控制在扩展期间终止哪些 Auto Scaling 实例。	2012 年 9 月 17 日

特征	描述	发行日期
对 IAM 角色的支持	使用 IAM EC2 实例配置文件启动实例。您可以使用此功能为实例指定 IAM 角色，允许您的应用程序安全地访问其他亚马逊云科技服务。有关更多信息，请参阅使用 IAM 角色启动 Auto Scaling 实例。	2012 年 6 月 11 日
支持竞价型实例	使用启动配置启动竞价型实例。有关更多信息，请参阅为容错和灵活的应用程序请求竞价型实例。	2012 年 6 月 7 日
对组和实例进行标记	为 Auto Scaling 组添加标签，并指定该标签也适用于在创建标签后启动的 EC2 实例。有关更多信息，请参阅标记 Auto Scaling 组和实例。	2012 年 1 月 26 日
Amazon SNS 支持	每当 Amazon A EC2 uto Scaling 启动或终止 EC2实例时，使用 Amazon SNS 接收通知。有关更多信息，请参阅在 Auto Scaling 组扩展时获取 SNS 通知。 Amazon A EC2 uto Scaling 还添加了以下新功能： • 使用 cron 语法设置重复扩展活动的能力。有关更多信息，请参阅 `PutScheduledUpdateGroupAction` API 操作。 • 一项新的配置设置，允许您在不向负载均衡器中添加已启动的实例的情况下进行横向扩展（LoadBalancer）。有关更多信息，请参阅 `ProcessType` API 数据类型。 • `DeleteAutoScalingGroup` 操作中的`ForceDelete` 标志，告诉 Amazon A EC2 uto Scaling 删除与其关联的实例的 Auto Scaling 组，而不必先等待实例终止。有关更多信息，请参阅 `DeleteAutoScalingGroup` API 操作。	2011 年 7 月 20 日
计划扩展操作	增加了对计划扩展操作的支持。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的计划扩展。	2010 年 12 月 2 日
Amazon VPC 支持	添加了对 Amazon VPC 的支持。有关更多信息，请参阅在 VPC 中启动 Auto Scaling 实例。	2010 年 12 月 2 日

特征	描述	发行日期
支持 HPC 集群	增加了对于高性能计算 (HPC) 集群的支持。	2010 年 12 月 2 日
支持运行状况检查	增加了对 Amazon A EC2 uto Scaling 托管实例使用 Elastic Load Balancing EC2 运行状况检查的支持。有关更多信息，请参阅自动扩缩组中实例的运行状况检查。	2010 年 12 月 2 日
Support 支持 CloudWatch 警报	删除了旧的触发机制，并重新设计了 Amazon A EC2 uto Scaling 以使用 CloudWatch 警报功能。有关更多信息，请参阅 Amazon A EC2 uto Scaling 的动态扩展。	2010 年 12 月 2 日
暂停和恢复扩缩	增加了对暂停和恢复扩展过程的支持。	2010 年 12 月 2 日
对 IAM 的支持	增加了对 IAM 的支持。有关更多信息，请参阅控制对您的 Amazon A EC2 uto Scaling 资源的访问权限。	2010 年 12 月 2 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。