AWS AppConfig

用户指南

AWS AppConfig: 用户指南

Table of Contents

什么是 AWS AppConfig？

使用 AWS AppConfig， AWS Systems Manager，以创建、管理和快速部署应用配置。 AWS AppConfig 支持对任何规模的应用进行控制部署，包括内置验证检查和监控。您可以将 AWS AppConfig 与 EC2 实例上托管的应用程序、AWS Lambda、容器、移动应用程序或 IoT 设备一起使用。

在部署应用配置时防止错误，尤其是 一个简单的类型可能导致n 意外停电， AWS AppConfig 包括验证器。验证程序提供语法或语义检查，以确保要部署的配置正常工作。要验证应用程序配置数据，请需要提供针对该配置运行的架构或 Lambda 函数。只有在配置数据有效时，才能执行配置部署或更新。

在配置部署期间，AWS AppConfig 监控应用程序以确保部署成功。如果系统遇到错误，AWS AppConfig 将回滚更改以最大限度减少对应用程序用户的影响。您可以为每个应用程序或环境配置一个包含部署条件的部署策略，包括速度、烘焙时间和要监控的警报。与错误监控类似，如果部署触发一个警报，则 AWS AppConfig 自动回滚到以前的版本。

AWS AppConfig 支持多个使用案例。下面是一些示例：

- 应用程序调整: 使用 AWS AppConfig 仔细介绍只能使用生产流量测试的应用程序的更改。
- 功能切换: 使用 AWS AppConfig 要开启需要及时部署的新功能，例如产品发布或公告。
- 允许列表: 使用 AWS AppConfig 允许优质订户访问已付内容。
- 运营问题: 使用 AWS AppConfig 在依赖性或其他外部因素影响系统时，减少应用对应用的压力。

我的组织如何从 AWS AppConfig 获益？

AWS AppConfig 具有以下优点。

- 快速部署一组目标的变化

  通过从中心位置部署配置更改，AWS AppConfig 简化了批量管理应用程序的过程。AWS AppConfig 支持 Systems Manager Parameter Store、Systems Manager (SSM) 文档和 Amazon S3 中存储的配置。您可以将 AWS AppConfig 与 EC2 实例上托管的应用程序、AWS Lambda、容器、移动应用程序或 IoT 设备一起使用。
- 减少配置更改中的错误

  AWS AppConfig 允许您创建规则以验证配置，从而减少应用程序停机时间。无法部署无效的配置。AWS AppConfig 提供了两种方法以验证配置。
  - 对于语法验证，您可以使用 JSON 架构。AWS AppConfig 使用 JSON 架构验证配置，以确保配置更改符合应用程序要求。
  - 对于语义验证，您可以调用 AWS Lambda 函数以在部署之前运行配置。
- 更新应用程序而不会发生中断

  AWS AppConfig 在运行时将配置更改部署到目标，而无需执行繁重的生成过程或停止使用目标。
- 控制在应用程序中部署更改

  在将配置更改部署到目标时，AppConfig 允许您使用部署策略以最大限度降低风险。您可以使用部署策略的速率控制，以确定希望应用程序目标接收配置更改的速度。

支持哪些类型的目标？

您可以将 AWS AppConfig 与 EC2 实例上托管的应用程序、AWS Lambda、容器、移动应用程序或 IoT 设备一起使用。目标不需要配置 Systems Manager SSM Agent 或其他 Systems Manager 功能所需的 AWS Identity and Access Management (IAM) 实例配置文件。这意味着，AWS AppConfig 适用于非托管实例。

使用 AWS AppConfig 是否需要支付费用？

是 有关更多信息，请参阅 AWS Systems Manager 定价。

我是否必须更改应用程序才能使用 AWS AppConfig？

是 您必须使用 GetConfiguration API 操作配置应用程序以轮询新的配置更新。在新配置或更新的配置准备就绪后，AWS AppConfig 将配置文件部署到部署策略中的每个目标。

AWS AppConfig 是如何工作的？

概括来说，可通过三个过程使用 AWS AppConfig。

1. 配置 AWS AppConfig 以与应用程序一起使用。
2. 允许应用程序代码定期检查并从 AWS AppConfig 接收配置数据。
3. 部署新配置或更新的配置。

配置 AWS AppConfig 以与应用程序一起使用。

要配置 AWS AppConfig 以与应用程序一起使用，您需要设置三种类型的资源。

Resource	Details (详细信息)。
应用程序	AWS AppConfig 中的应用程序是为客户提供功能的代码的逻辑单元。例如，应用程序可以是在 EC2 实例上运行的微服务、用户安装的移动应用程序、使用 Amazon API Gateway 和 AWS Lambda 的无服务器应用程序或您代表其他人运行的任何系统。
Environment	对于每个应用程序，您可以定义一个或多个环境。环境是 AWS AppConfig 应用程序的逻辑部署组，例如 `Beta` 或 `Production` 环境中的应用程序。您也可以为应用程序子组件定义环境，例如应用程序的 `Web`、`Mobile` 和 `Back-end` 组件。您可以为每个环境配置 Amazon CloudWatch 警报。系统在部署配置期间监控警报。如果触发警报，系统将回滚配置。
配置文件	A 配置文件 启用 AWS AppConfig 在其存储位置访问配置。您可以按以下格式和位置存储配置： • AWS AppConfig 托管配置存储中的 YAML、JSON 或文本文档 • Amazon Simple Storage Service (Amazon S3) 存储桶中的对象 • Systems Manager 文档存储中的文档 • Parameter Store 中的参数

Resource	Details (详细信息)。
	配置配置文件还可以包括可选验证器，以确保配置数据的协同和语义正确。 AWS AppConfig 启动部署时，使用验证器执行检查。在对配置目标进行任何更改之前，如果检测到任何错误，部署将会停止。

允许应用程序代码定期检查并从 AWS AppConfig 接收配置数据

通过使用 GetConfiguration API 操作，AWS AppConfig 以受控方式提供动态配置更新。您必须配置应用程序代码以定期调用该 API 操作。在调用时，代码发送以下信息：

- AWS AppConfig 应用程序、环境和配置文件的 ID。
- 唯一的应用程序实例标识符，称为客户端 ID。
- 应用程序代码已知的上一配置版本。

在部署新配置时（这意味着，具有新版本的配置），AWS AppConfig 响应 GetConfiguration 请求并返回新的配置数据。

部署新配置或更新的配置。

AWS AppConfig 允许您按最适合应用程序使用案例的方式部署配置。您可以在几秒钟内部署更改，也可以缓慢部署更改以评估更改的影响。帮助您控制部署的 AWS AppConfig 资源称为部署策略。部署策略包含以下信息：

- 部署的总持续时间 (DeploymentDurationInMinutes)。
- 在每个间隔内接收部署的配置的目标百分比 (GrowthFactor)。
- 在将部署视为完成并且不再符合自动回滚条件之前，AWS AppConfig 监控警报所花的时间。(FinalBakeTimeInMinutes)。

您可以使用涵盖常见方案的内置部署策略，也可以创建自己的策略。在创建或选择部署策略后，您可以开始部署。在开始部署时，将调用 StartDeployment API 操作。该调用包括应用程序、环境和配置文件的 ID 以及（可选）要部署的配置数据版本。该调用还包括要使用的部署策略的 ID，该策略确定如何部署配置数据。

什么是 AWS AppConfig 的服务配额？

Resource	配额
AWS AppConfig 应用程序数	100
部署策略数	20
每个 AWS AppConfig 应用程序的环境数	20
每个 AWS AppConfig 应用程序的配置文件数	100

Note

有关存储 AWS AppConfig 配置的服务的配额的信息，请参阅 关于配置存储配额和限制 (p. 11)。

开始使用 AWS AppConfig

以下主题介绍了开始使用 AWS AppConfig 的重要任务。

主题

配置权限 AWS AppConfig

AWS AppConfig 使用以下 API 操作。

AWS AppConfig 资源	API 操作
AWS AppConfig 应用程序数	CreateApplication、UpdateApplication、DeleteApplication、GetAp 有关更多信息，请参阅步骤 1. 创建 AWS AppConfig 应用 (p. 9)
环境	CreateEnvironment、UpdateEnvironment、DeleteEnvironment、G 有关更多信息，请参阅步骤 2. 创建 环境 (p. 9)
配置	GetConfiguration 有关更多信息，请参阅步骤 3 创建配置和配置文件 (p. 10)
配置文件	CreateConfigurationProfile、UpdateConfigurationProfile、DeleteC 有关更多信息，请参阅步骤 3 创建配置和配置文件 (p. 10)
部署策略数	CreateDeploymentStrategy、UpdateDeploymentStrategy、Delete 有关更多信息，请参阅步骤 4. 创建部署策略 (p. 20)
部署。	StartDeployment、StopDeployment、ListDeployments 有关更多信息，请参阅步骤 5. 部署配置 (p. 23)

我们建议您创建限制性的 IAM 权限策略，以便为用户、组和角色授予所需的最少权限以在 AWS AppConfig 中执行所需的操作。

例如，您可以创建一个只读 IAM 权限策略，以仅包含 AWS AppConfig 使用的 `Get` 和 `List` API 操作，如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetDocument",
        "ssm:ListDocuments",
        "appconfig:ListApplications",
        "appconfig:GetApplication",
        "appconfig:ListEnvironments",
        "appconfig:GetEnvironment",
        "appconfig:ListConfigurationProfiles",
        "appconfig:GetConfigurationProfile",
        "appconfig:ListDeploymentStrategies",
        "appconfig:GetDeploymentStrategy",
        "appconfig:GetConfiguration",
        "appconfig:ListDeployments"

      ],
      "Resource": "*"
    }
  ]
}
```

Important

仅限受信任的用户访问 StartDeployment 和 StopDeployment API 操作，这些用户了解将新配置部署到目标的责任和后果。

有关创建和编辑的更多信息 IAM 政策，请参阅 创建 IAM 政策 在 IAM 用户指南. 有关如何将此策略分配到 IAM 组的信息，请参阅将策略附加到 IAM 组。

为 IAM 用户账户配置使用 AWS AppConfig 的权限

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Users (用户)。
3. 在列表中，选择一个名称。
4. 选择 Permissions 选项卡。
5. 在页面右侧，在 Permission policies (权限策略) 下，选择 Add inline policy (添加内联策略)。
6. 选择 JSON 选项卡。
7. 将默认内容替换为自定义权限策略。
8. 选择查看策略。
9. 在查看策略页面上，对于名称，输入内联策略的名称。例如：。**AWS AppConfig-<*action*>-Access**.
10. 选择 Create policy (创建策略)。

（可选）配置基于 CloudWatch 警报的回滚权限

您可以配置 AWS AppConfig 回滚至一个或多个配置的先前版本 Amazon CloudWatch 警报。在配置部署以响应 CloudWatch 警报时，您可以指定 AWS Identity and Access Management (IAM) 角色。AWS AppConfig 需要使用该角色，以便它可以监控 CloudWatch 警报，即使在当前 AWS 账户中未创建这些警报。

使用以下过程可创建 IAM 角色，该角色允许 AWS AppConfig 基于 CloudWatch 警报回滚。本节包括以下过程：

1. 步骤 1. 根据基于 CloudWatch 警报 (p. 6)
2. 步骤 2. 创建 IAM 基于 CloudWatch 警报 (p. 6)

3. 步骤 3 添加信托关系 (p. 7)

步骤 1. 根据基于 CloudWatch 警报

可以使用以下过程创建一个 IAM 策略，以便为 AWS AppConfig 授予权限以调用 `DescribeAlarms` API 操作。

创建基于 CloudWatch 警报回滚的 IAM 权限策略

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择策略，然后选择创建策略。
3. 在创建策略页面上，选择 JSON 选项卡。
4. 将 JSON 选项卡上的默认内容替换以下权限策略，然后选择 Review。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:DescribeAlarms"
                ],
                "Resource": "*"
            }
        ]
    }
```

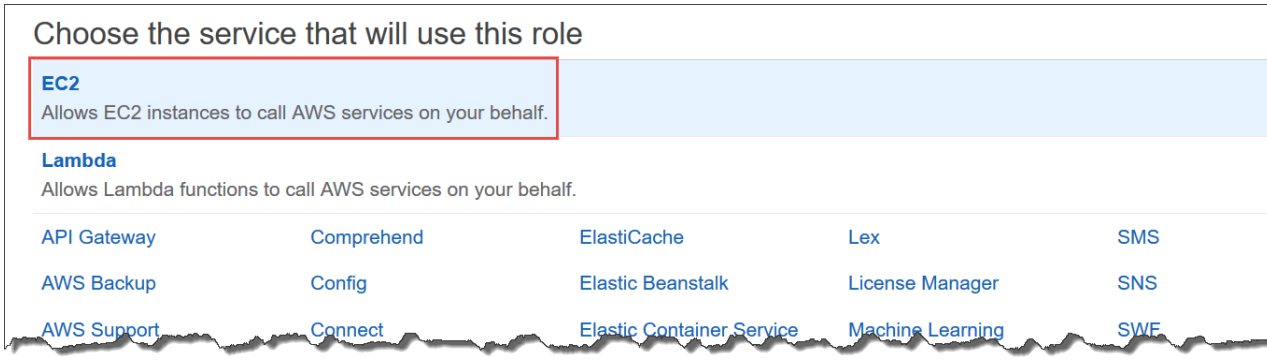
5. 在检查页面上，在角色名称字段中输入 **`SSMCloudWatchAlarmDiscoveryRole`**。
6. 选择 Create policy (创建策略)。系统将让您返回到 Policies 页面。

步骤 2. 创建 IAM 基于 CloudWatch 警报

使用以下过程创建 IAM 角色并向其分配您在上一过程中创建的策略。

创建基于 CloudWatch 警报回滚的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Roles (角色)，然后选择 Create Role (创建角色)。
3. 在选择受信任实体的类型下，选择 AWS 服务。
4. 立即在 选择将使用此角色的服务，选择 EC2，然后选择 下一步: Permissions (权限)



5. 在 Attached permissions policy 页面上，搜索 SSMCloudWatchAlarmDiscoveryRole。

6. 选择此策略，然后选择 下一步: 标签
7. 输入此角色的标签，然后选择 下一步: 审核
8. 在 Create role 页面上的 Role name 字段中，输入一个名称，然后选择 Create role。
9. 在 Roles 页面上，选择您刚刚创建的角色。此时将打开摘要页面。

步骤 3 添加信托关系

使用以下过程将您刚刚创建的角色配置为信任 AWS AppConfig。

为 AWS AppConfig 添加信任关系

1. 在刚刚创建的角色的摘要页面上，选择信任关系选项卡，然后选择编辑信任关系。
2. 编辑策略以仅包含“appconfig.amazonaws.com”，如以下示例中所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. 选择 Update Trust Policy。

AWS AppConfig 与 CodePipeline

AWS AppConfig 是针对 AWS CodePipeline （ CodePipeline ）。 CodePipeline 是一项全面管理的持续交付服务，帮助您实现快速可靠的应用程序和基础架构更新的发布管道自动化。 CodePipeline 根据您定义的发布模式，每次发生代码更改时，都会自动执行发布流程的构建、测试和部署阶段。有关更多信息，请参阅什么是 AWS CodePipeline 。

集成 AWS AppConfig 带有 CodePipeline 提供以下优势:

- 使用的客户 CodePipeline 要管理编排，现在有一种轻轻的方法来部署对其应用程序的配置更改，而无需部署其整个代码库。
- 希望使用的客户 AWS AppConfig 管理配置部署，但是是因为 AWS AppConfig 不支持其当前代码或配置存储，现在有其他选项。 CodePipeline 支持 AWS CodeCommit、Github和Bitbucket (等等) 。

整合工作原理

您开始设置和配置 CodePipeline. 这包括将配置添加到A CodePipeline-supportedcodestore。接下来，您可以设置 AWS AppConfig 执行以下任务。

- 创建应用程序
- 创建一个环境。
- 创建 AWS CodePipeline 配置文件.
- 选择预定义的部署策略或创建自己的.

完成这些任务后，您将在 CodePipeline 指定 AWS AppConfig 作为 部署提供商. 然后您可以更改配置并将其上传到 CodePipeline 代码商店。上传新配置会自动触发新部署 CodePipeline. 部署完成后，您可以验证更改。有关创建指定管道的信息 AWS AppConfig 作为部署提供商，请参阅 教程: 创建使用的管道 AWS AppConfig 作为部署提供商 在 AWS CodePipeline 用户指南.

使用 AWS AppConfig

本部分包括描述如何使用的主题 AWS AppConfig 为主机或目标创建和部署配置的功能。

主题

步骤 1. 创建 AWS AppConfig 应用

AWS AppConfig 中的应用程序是为客户提供功能的代码的逻辑单元。例如，应用程序可以是在 EC2 实例上运行的微服务、用户安装的移动应用程序、使用 Amazon API Gateway 和 AWS Lambda 的无服务器应用程序或您代表其他人运行的任何系统。

可以使用以下过程通过 AWS Systems Manager 控制台创建 AWS AppConfig 应用程序。

创建应用程序

1. Open the AWS Systems Manager console at https://console.amazonaws.cn/systems-manager/appconfig/.
2. 在导航窗格中，选择 AWS AppConfig。
3. 在 Applications (应用程序) 选项卡上，选择 Create application (创建应用程序)。
4. 对于 Name (名称)，请输入应用程序的名称。
5. 对于 Description (描述)，请输入有关应用程序的信息。
6. 在 Tags (标签) 部分中，输入一个键和可选的值。您最多可以为一个资源指定 50 个标签。
7. 选择 Create application (创建应用程序)。

AWS AppConfig 将创建应用程序，然后显示 Environments (环境) 选项卡。继续执行步骤 2. 创建 环境 (p. 9)。您可以开始执行指示“在 Environments (环境) 选项卡上...”的过程。

步骤 2. 创建 环境

对于每个 AWS AppConfig 应用程序，您定义一个或多个环境。环境是 AppConfig 目标的逻辑部署组，例如 `Beta` 或 `Production` 环境中的应用程序。您也可以为应用程序子组件定义环境，例如应用程序的 `Web`、`Mobile` 和 `Back-end` 组件。您可以为每个环境配置 Amazon CloudWatch 警报。系统在部署配置期间监控警报。如果触发警报，系统将回滚配置。

开始前的准备工作

如果要允许 AWS AppConfig 回滚配置以响应 CloudWatch 警报，则必须配置一个 AWS Identity and Access Management (IAM) 角色并为其授予允许 AWS AppConfig 响应 CloudWatch 警报的权限。您可以在以下过程中选择该角色。有关更多信息，请参阅 （可选）配置基于 CloudWatch 警报的回滚权限 (p. 5)。)

可以使用以下过程通过 AWS Systems Manager 控制台创建 AWS AppConfig 环境。

创建环境

1. Open the AWS Systems Manager console at https://console.amazonaws.cn/systems-manager/appconfig/.
2. 在导航窗格中，选择 AWS AppConfig。
3. 在 Applications (应用程序) 选项卡上，选择您在步骤 1. 创建 AWS AppConfig 应用 (p. 9)中创建的应用程序，然后选择 View details (查看详细信息)。
4. 在 Environments (环境) 选项卡上，选择 Create environment (创建环境)。
5. 对于 Name (名称)，请输入环境的名称。
6. 对于 Description (描述)，请输入有关环境的信息。
7. 在 Monitors (监控器) 部分中，如果您希望 AWS AppConfig 在触发警报时回滚配置，请选择 Enable rollback on CloudWatch Alarms (在触发 CloudWatch 警报时启用回滚)。
8. 在 IAM role (IAM 角色) 列表中，选择具有在触发警报时回滚配置的权限的 IAM 角色。
9. 在 CloudWatch alarms (CloudWatch 警报) 列表中，选择一个或多个要监控的警报。
10. 在 Tags (标签) 部分中，输入一个键和可选的值。您最多可以为一个资源指定 50 个标签。
11. 选择 Create environment (创建环境)。

AWS AppConfig 将创建环境，然后显示 Environment details (环境详细信息) 页面。继续执行步骤 3 创建配置和配置文件 (p. 10)。

步骤 3 创建配置和配置文件

配置 是一组影响应用程序行为的设置。例如，您可以创建和部署配置，这些配置谨慎地引入对应用程序的更改，或者启用要求及时部署的新功能（如产品发布或公告）。以下是访问列表配置的一个非常简单的示例。

```
{
    "AccessList": [
        {
            "user_name": "Mateo_Jackson"
        },
        {
            "user_name": "Jane_Doe"
        }
    ]
}
```

A 配置文件 启用 AWS AppConfig 从源位置访问配置。您可以按以下格式和位置存储配置：

- AWS AppConfig 托管配置存储中的 YAML、JSON 或文本文档
- Amazon Simple Storage Service (Amazon S3) 存储桶中的对象
- Systems Manager 文档存储中的文档
- Parameter Store 中的参数
- 任何 集成源操作 支持者 AWS CodePipeline

配置文件包含以下信息。

- 存储配置的URI位置。
- 提供对配置的访问权限的 AWS Identity and Access Management (IAM) 角色。

- 配置数据的验证程序。您可以使用 JSON 架构或 AWS Lambda 函数验证配置文件。配置文件最多可以具有两个验证程序。

对于存储在 AWS AppConfig 托管配置存储或 SSM 文档中的配置，您可以在创建配置文件时使用 Systems Manager 控制台创建配置。本主题稍后介绍了该过程。

对于存储在 SSM 参数或 S3 中的配置，您必须先创建参数或对象，然后将其添加到 Parameter Store 或 S3。创建参数或对象后，您可以使用本主题中的过程创建配置文件。有关在中创建参数的信息 Parameter Store，参见 创建系统管理器参数 在 AWS Systems Manager 用户指南.

关于配置存储配额和限制

AWS AppConfig 支持的配置存储具有以下配额和限制。

	AWS AppConfig 托管配置存储	S3	Parameter Store	文档存储	AWS CodePipeline
配置大小限制	64 KB	1MB 由 AWS AppConfig 而不是 S3 强制实施	4 KB（免费套餐）/8 KB（高级参数）	64 KB	1MB 执行人 AWS AppConfig，不是 CodePipeline
资源存储限制	1GB	无限制。	10000 个参数（免费套餐）/100000 个参数（高级参数）	500 个文档	受每个应用程序的配置文件数量限制（每个应用程序的100个配置文件）
服务器端加密	是	否	否	否	是
AWS CloudFormation	是 .support	不用于创建或更新数据	是	否	是
验证创建或更新 API 操作	不支持	不支持	支持正则表达式	所有放置和更新 API 操作都需要 JSON 架构	不支持
定价	免费	参见 Amazon S3 定价	参见 AWS Systems Manager 定价	免费	参见 AWS CodePipeline 定价

关于 AWS AppConfig 托管配置存储

AWS AppConfig 包括内部或托管配置存储。配置必须为 64 KB 或更小。与其他配置存储选项相比，AWS AppConfig 托管配置存储提供了以下好处。

- 您无需设置和配置其他服务，如 Amazon Simple Storage Service (Amazon S3) 或 Parameter Store。
- 您无需配置 AWS Identity and Access Management (IAM) 权限即可使用配置存储。
- 您可以按 YAML、JSON 或文本文档格式存储配置。
- 使用存储不产生任何费用。
- 您可以创建配置并在创建配置文件时将其添加到存储。

创建配置和配置文件

使用以下过程通过 AWS Systems Manager 控制台创建 AWS AppConfig 配置文件和（可选）配置。

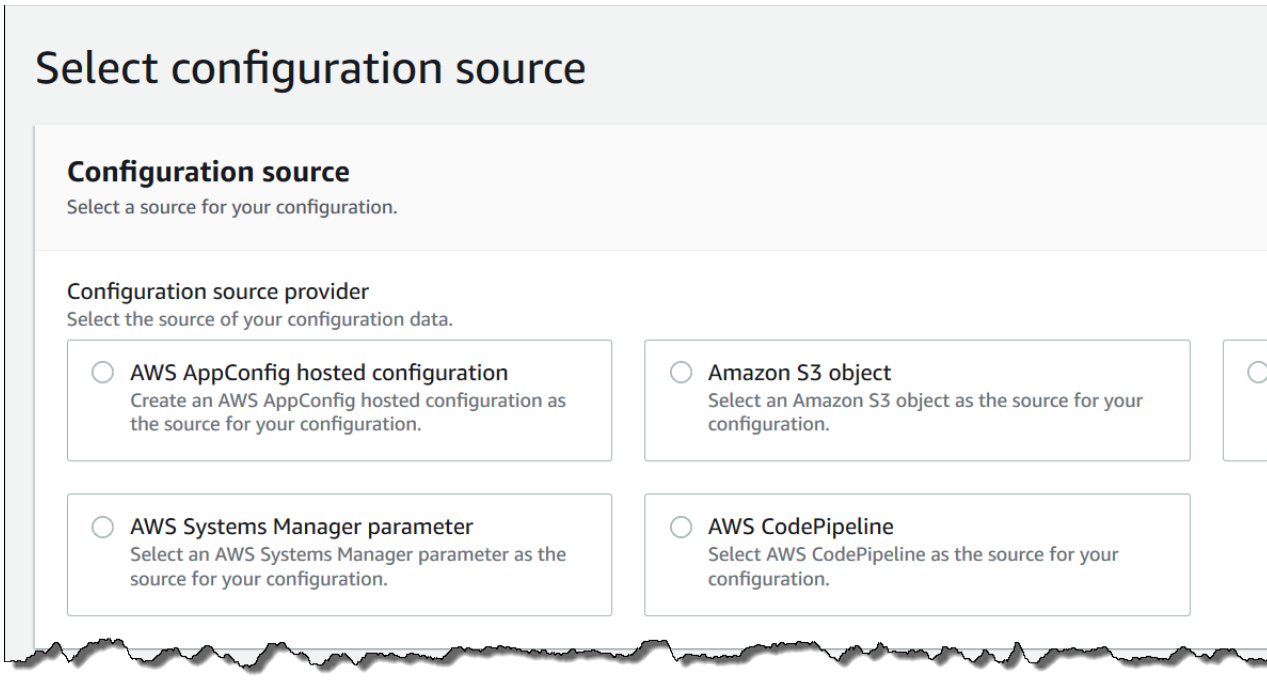
开始前的准备工作

在完成本节中的过程之前，请阅读以下相关内容。

- 以下过程要求您指定 IAM 服务角色，以便 AWS AppConfig 能够访问您选择的配置存储中的配置数据。如果您使用 AWS AppConfig 托管配置存储，则不需要此角色。如果您选择 S3、Parameter Store 或 Systems Manager 文档存储，则必须选择现有 IAM 角色，或选择让系统自动为您创建角色的选项。有关该角色的更多信息，请参阅 关于配置文件 IAM 角色 (p. 15)。
- 如果要为 S3 中存储的配置创建配置文件，则必须配置权限。有关使用 S3 作为配置存储的权限和其他要求的更多信息，请参阅 关于存储在 Amazon S3 中的配置 (p. 16)。
- 如果您要使用验证程序，请查看使用验证程序的详细信息和要求。有关更多信息，请参阅 关于验证程序 (p. 19)。)

创建配置文件

1. Open the AWS Systems Manager console at https://console.amazonaws.cn/systems-manager/appconfig/.
2. 在 Applications (应用程序) 选项卡上，选择您在 Create an AWS AppConfig configuration (创建 AppConfig 配置) (p. 9) 中创建的应用程序，然后选择 Configuration profiles (配置文件) 选项卡。
3. 选择 Create configuration profile (创建配置文件)。
4. 对于 Name (名称)，请输入配置文件的名称。
5. 对于 Description (描述)，请输入有关配置文件的信息。
6. 在 Tags (标签) 部分中，输入一个键和可选的值。您最多可以为一个资源指定 50 个标签。
7. 在 Select configuration source (选择配置源) 页面上，选择一个选项。



8. • 如果您选择了 AWS AppConfig hosted configuration (AWS AppConfig 托管配置)，请选择 YAML、JSON 或 Text (文本)，然后在字段中输入您的配置。选择 Next (下一步)，然后转到此过程中的步骤 10。

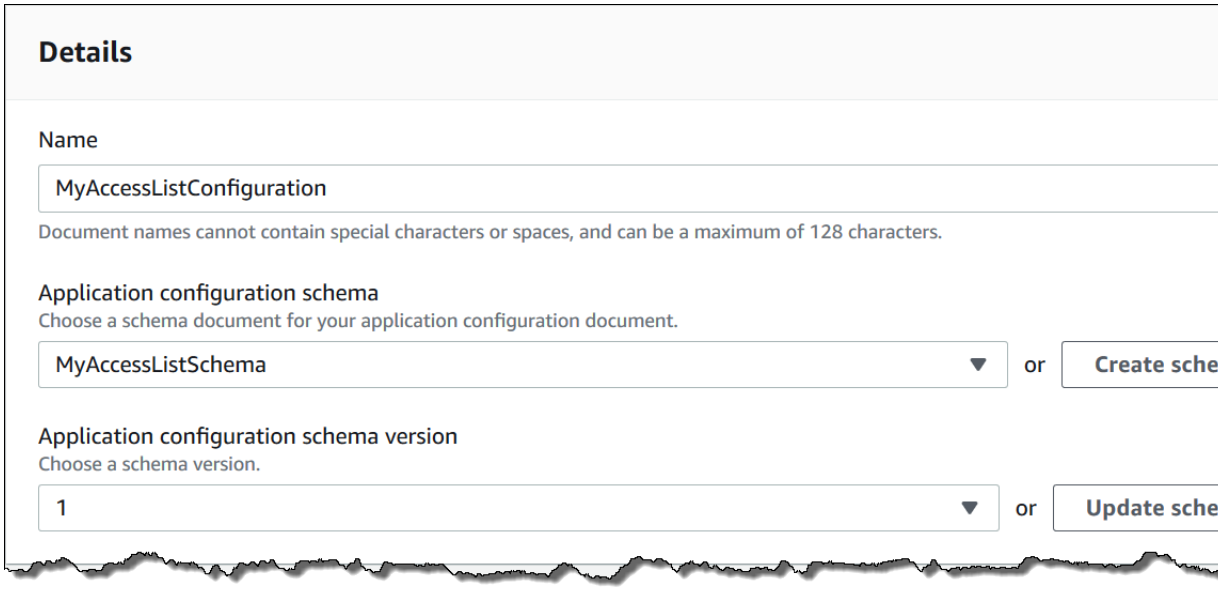
- 如果您选择了 Amazon S3 object (Amazon S3 对象)，则输入对象 URI。选择 Next (下一步)。
- 如果您选择了 AWS Systems Manager parameter (AWS Systems Manager 参数)，则从列表中选择参数的名称。选择 Next (下一步)。
- 如果您选择 AWS CodePipeline，然后选择 下一步 并转到本程序中的步骤10。
- 如果您选择了 AWS Systems Manager Document (AWS Systems Manager 文档)，请完成以下步骤。

a. 在 文件源 部分，选择 已保存文档 或 新文档.

b. 如果选择 Saved document (已保存的文档)，则从列表中选择 SSM 文档。如果选择 New document (新文档)，则会显示 Details (详细信息) 和 Content (内容) 部分。

c. 在 Details (详细信息) 部分中，输入新应用程序配置的名称。

d. 对于 Application configuration schema (应用程序配置架构) 部分，使用列表选择 JSON 架构或选择 Create schema (创建架构)。如果选择 Create schema (创建架构)，Systems Manager 将打开 Create schema (创建架构) 页面。在 Content (内容) 部分中输入架构详细信息，然后选择 Create schema (创建架构)。



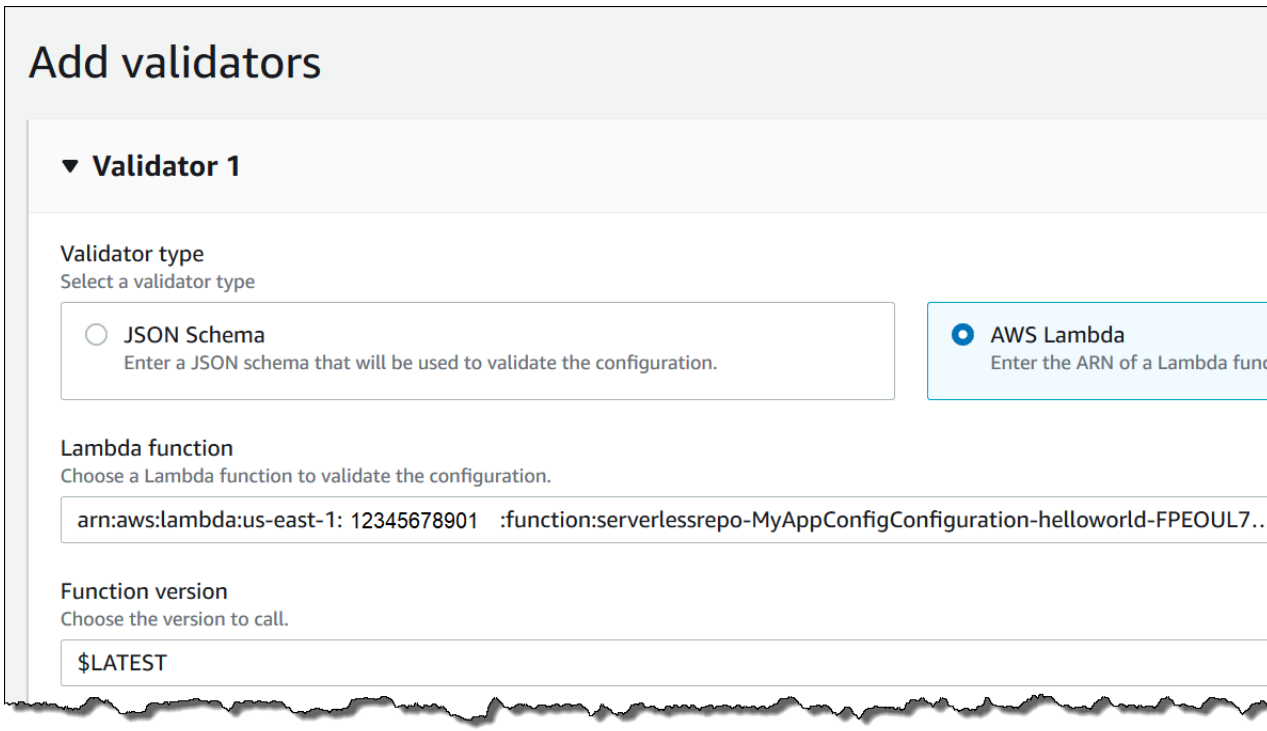
e. 对于 Application configuration schema version (应用程序配置架构版本)，从列表中选择版本，或选择 Update schema (更新架构) 以编辑架构并创建新版本。

f. 在 Content (内容) 部分中，选择 YAML 或 JSON，然后在字段中输入配置数据。

g. 选择 Next (下一步)。

9. 在 Service role (服务角色) 部分中，选择 New service role (新服务角色) 以让 AWS AppConfig 创建提供配置数据访问权限的 IAM 角色。AWS AppConfig 根据您以前输入的名称自动填充 Role name (角色名称) 字段。或者，要选择已位于 IAM 中的角色，请选择 Existing service role (现有服务角色)。使用 Role ARN (角色 ARN) 列表选择角色。

10. 在 Add validators (添加验证程序) 页面上，选择 JSON Schema (JSON 架构) 或 AWS Lambda。如果选择 JSON Schema (JSON 架构)，请在字段中输入 JSON 架构。如果您选择 AWS Lambda，请从列表中选择函数 Amazon 资源名称 (ARN) 和版本。



Important

在可以将配置添加到系统之前，SSM 文档中存储的配置数据必须对照关联的 JSON 架构进行验证。SSM 参数不需要验证方法，但我们建议您通过使用 AWS Lambda 为新的或更新的 SSM 参数配置创建验证检查。

11. 选择 Create configuration profile (创建配置文件)。

Important

如果您为 AWS CodePipeline，在创建部署策略后，如下一节所述，您必须在 CodePipeline 指定 AWS AppConfig 作为 部署提供商. 有关创建指定管道的信息 AWS AppConfig 作为部署提供商，请参阅 教程: 创建使用的管道 AWS AppConfig 作为部署提供商 在 AWS CodePipeline 用户指南.

继续执行步骤 4. 创建部署策略 (p. 20)。

关于配置文件 IAM 角色

您可以使用 AWS AppConfig 创建提供配置数据访问权限的 IAM 角色，如以下过程中所述。或者，您可以自行创建 IAM 角色并从列表中选择它。如果使用 AWS AppConfig 创建角色，系统将根据所选的配置源类型创建角色并指定以下权限策略之一。

配置源是 SSM 文档

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetDocument"
            ],
            "Resource": [
                "arn:aws:ssm:AWS-Region:account-number:document/document-name"
            ]
        }
    ]
}
```

配置源是 Parameter Store 参数

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": [
                "Arn:aws:ssm:AWS-Region:account-number:parameter/parameter-name"
            ]
        }
    ]
    }
```

如果使用 AWS AppConfig 创建角色，系统还会为角色创建以下信任关系。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

关于存储在 Amazon S3 中的配置

您可以在 Amazon Simple Storage Service (Amazon S3) 存储桶中存储配置。在创建配置文件时，将指定存储桶中单个 S3 对象的 URI。您还将指定 AWS Identity and Access Management (IAM) 角色的 Amazon 资源名称 (ARN)，该角色向 AWS AppConfig 授予获取对象的权限。在为 Amazon S3 对象创建配置文件之前，请注意以下限制。

限制	Details (详细信息)。
Size	存储为 S3 对象的配置的最大大小可以为 1 MB。
Object encryption	配置文件无法以加密的 S3 对象为目标。
存储类	AWS AppConfig 支持以下S3存储类: `STANDARD`， `INTELLIGENT_TIERING`， `REDUCED_REDUNDANCY`， `STANDARD_IA`，和 `ONEZONE_IA`。不支持以下类别: 所有S3冰川类别（`GLACIER` 和 `DEEP_ARCHIVE`）。
版本控制	AWS AppConfig 要求 S3 对象使用版本控制。

配置存储为 Amazon S3 对象的配置的权限

在为存储为 S3 对象的配置创建配置文件时，您必须为 IAM 角色指定 ARN，该角色向 AWS AppConfig 授予获取对象的权限。该角色必须包括以下权限。

对 S3 对象的访问权限

- s3:GetObject
- s3:GetObjectVersion

列出 S3 存储桶的权限

s3:ListAllMyBuckets

对用于存储对象的 S3 存储桶的访问权限

- s3:GetBucketLocation
- s3:GetBucketVersioning
- s3:ListBucket
- s3:ListBucketVersions

完成以下过程可创建一个角色，该角色使 AWS AppConfig 能够获取存储在 S3 对象中的配置。

创建用于访问 S3 对象的 IAM 策略

使用以下过程可创建一个 IAM 策略，该策略使 AWS AppConfig 能够获取存储在 S3 对象中的配置。

创建用于访问 S3 对象的 IAM 策略

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择策略，然后选择创建策略。
3. 在创建策略页面上，选择 JSON 选项卡。
4. 使用有关 S3 存储桶和配置对象的信息更新以下示例策略。然后将策略粘贴到 JSON 选项卡上的文本字段中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::my-bucket/my-configurations/my-configuration.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

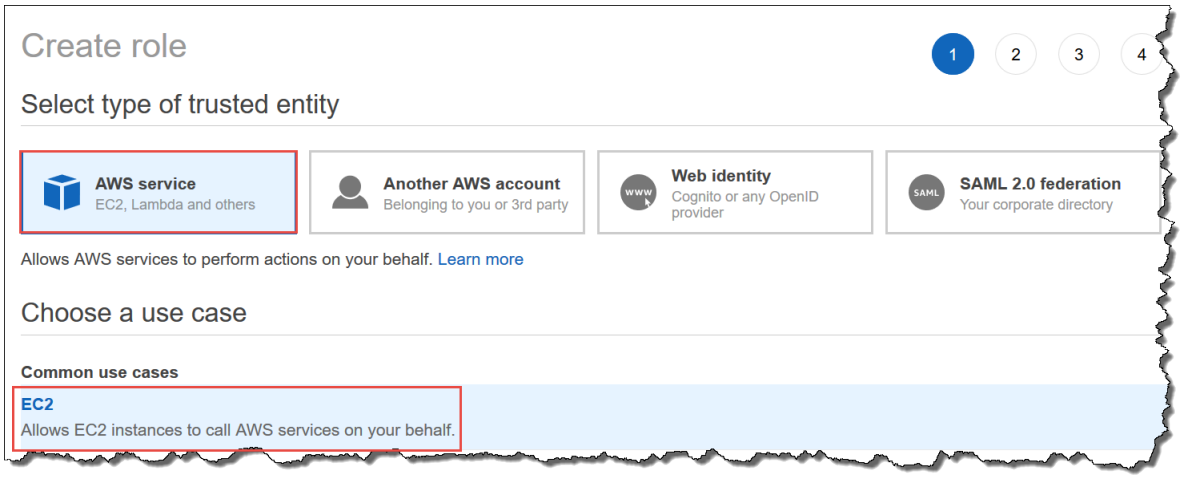
5. 选择查看策略。
6. 在 Review policy (查看策略) 页面上，在 Name (名称) 框中键入名称，然后键入描述。
7. 选择 Create policy (创建策略)。系统将让您返回到角色页。

创建用于访问 S3 对象的 IAM 角色

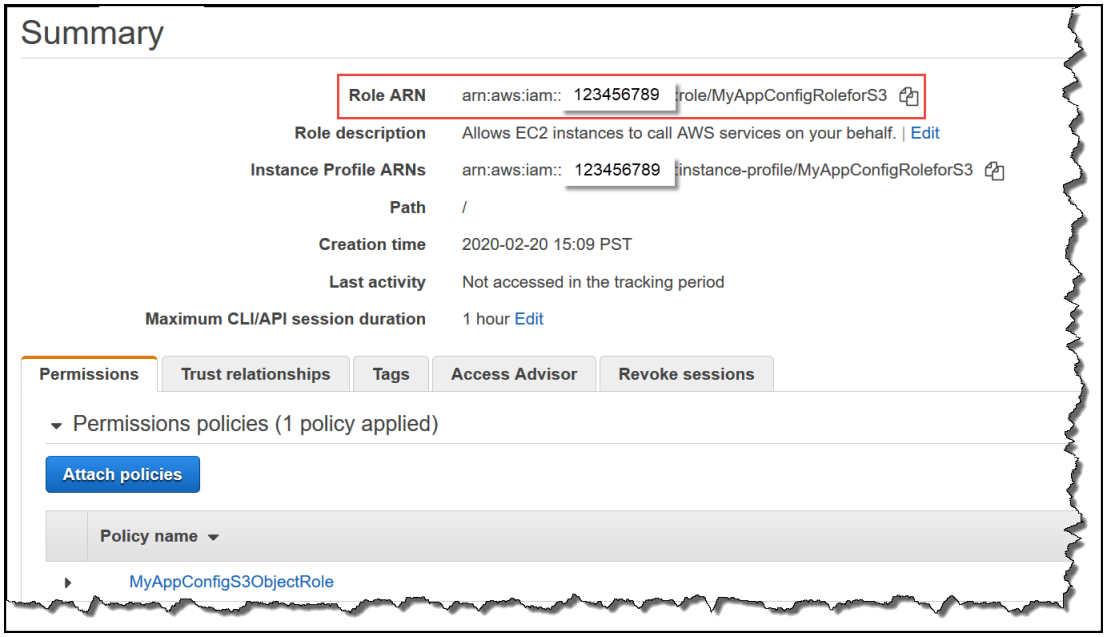
使用以下过程可创建一个 IAM 角色，该角色使 AWS AppConfig 能够获取存储在 S3 对象中的配置。

创建用于访问 Amazon S3 对象的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Roles (角色)，然后选择 Create Role (创建角色)。
3. 在 Select type of trusted entity (选择受信任实体的类型) 部分中，选择 AWS service (AWS 服务)。



4. 在 选择使用案例 章节， 常用用例，选择 EC2，然后选择 下一步: Permissions (权限)
5. 在 Attach permissions policy (附加权限策略) 页面上的搜索框中，输入您在上一过程中创建的策略的名称。
6. 选择政策，然后选择 下一步: 标签
7. 在 Add tags (optional) (添加标签(可选)) 页面上，输入密钥和可选值，然后选择 Next: Review (下一步: 审核)。
8. 在 Review (审核) 页面上，在 Role name (角色名称) 字段中键入名称，然后键入描述。
9. 选择创建角色。系统将让您返回到角色页。
10. 在角色页面中，选择刚刚创建的角色以打开摘要页面。记下角色名称和角色 ARN。在本主题的后面部分中创建配置文件时，您将指定角色 ARN。



创建信任关系

使用以下过程将您刚刚创建的角色配置为信任 AWS AppConfig。

添加信任关系

1. 在刚刚创建的角色的摘要页面上，选择信任关系选项卡，然后选择编辑信任关系。
2. 删除 "ec2.amazonaws.com" 并添加 "appconfig.amazonaws.com"，如以下示例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. 选择 Update Trust Policy。

关于验证程序

创建配置和配置文件时，最多可以指定两个验证程序。验证程序可确保您的配置数据在语法和语义上是正确的。您可以在 JSON 架构中或作为 AWS Lambda 函数创建验证程序。

> **Important**
>
> 在可以将配置添加到系统之前，SSM 文档中存储的配置数据必须对照关联的 JSON 架构进行验证。SSM 参数不需要验证方法，但我们建议您通过使用 AWS Lambda 为新的或更新的 SSM 参数配置创建验证检查。

JSON 架构验证程序

如果在 SSM 文档中创建配置，则必须为该配置指定或创建 JSON 架构。JSON 架构定义每个应用程序配置设置允许的属性。JSON 架构的作用类似于一组规则，用于确保新配置设置或更新的配置设置符合应用程序所需的最佳实践。以下是 示例。

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "$id$",
  "description": "BasicFeatureToggle-1",
  "type": "object",
  "additionalProperties": false,
  "patternProperties": {
      "[^\\s]+$": {
          "type": "boolean"
      }
  },
  "minProperties": 1
}
```

使用本主题中的过程创建配置时，AWS AppConfig 验证配置是否符合架构要求。如果不符合要求，Systems Manager 将返回验证错误。

> **Note**
>
> AWS AppConfig 对于内联架构支持 JSON 架构版本 4.X。如果您的应用程序配置需要不同的 JSON 架构版本，则必须创建 Lambda 验证程序。

AWS Lambda 验证程序

Lambda 函数验证程序必须配置了以下事件架构。AWS AppConfig 使用该架构调用 Lambda 函数。content 是 base64 编码的字符串，而 URI 是字符串。

```
{
   "content":Base64EncodedByteString,
   "uri":"The uri of the configuration"
}
```

AWS AppConfig 验证是否已在响应中设置 Lambda `X-Amz-Function-Error` 标头。如果函数引发异常，则 Lambda 设置此标头。有关 `X-Amz-Function-Error`，参见 错误处理和自动重试 AWS Lambda 在 AWS Lambda Developer Guide.

这里是一个成功验证的 Lambda 响应代码的简单示例。

```
import json
def handler(event, context):
     #Add your validation logic here
     print("We passed!")
```

以下是验证不成功的 Lambda 响应代码的简单示例。

```
def handler(event, context):
     #Add your validation logic here
     raise Exception("Failure!")
```

下面是另一个示例，仅在配置参数是质数时才验证。

```
function isPrime(value) {
    if (value < 2) {
        return false;
    }

    for (i = 2; i < value; i++) {
        if (value % i === 0) {
            return false;
        }
    }

    return true;
}

exports.handler = async function(event, context) {
    console.log('EVENT: ' + JSON.stringify(event, null, 2));
    const input = parseInt(Buffer.from(event.content, 'base64').toString('ascii'));
    const prime = isPrime(input);
    console.log('RESULT: ' + input + (prime ? ' is' : ' is not') + ' prime');
    if (!prime) {
        throw input + "is not prime";
    }
}
```

AWS AppConfig 在调用 `StartDeployment` 和 `ValidateConfigurationActivity` API 操作时调用您的验证 Lambda。您必须提供 appconfig.amazonaws.com 权限才能调用 Lambda。有关更多信息，请参阅为函数授予 AWS 服务的访问权限。

步骤 4. 创建部署策略

一个 AWS AppConfig 部署策略定义了配置部署的以下重要方面。

设置	Description

Deployment type (部署类型)	部署类型定义配置部署或推出方式。AWS AppConfig 支持线性和指数部署类型。
	• Linear (线性) 对于此类型， AWS AppConfig 按照在部署时间均匀分布的增长因子的增量进行部署。例如，使用步骤百分比 20 的线性部署最初使配置可用于 20% 的目标。在部署时间的五分之一过后，系统将百分比更新为 40%。这将一直持续到 100% 的目标设置为接收部署的配置。
	• 指数: 对于此类型， AWS AppConfig 使用以下公式按顺序处理部署: `G*(2^N)`。在这种配方中， `G` 是用户指定的步骤百分比 `N` 为配置部署到所有目标之前的步骤数。例如，如果将增长系数指定为 2，则系统将按如下方式推出配置：
	`2*(2^0)`
	`2*(2^1)`
	`2*(2^2)`
	以数字表示，部署的推出情况如下：2% 的目标、4% 的目标、8% 的目标，并持续到将配置部署到所有目标为止。

步骤百分比（增长系数）	该设置指定在部署的每个步骤中作为目标的调用方百分比。 Note 在开发工具包和 AWS AppConfig API 参考中，`step percentage` 称为 `growth factor`。
Deployment time (部署时间)	该设置指定 AWS AppConfig 部署到主机所花的时间。这不是超时值。这是一个按间隔处理部署的时段。 >
Bake time (烘焙时间)	该设置指定在执行下一个部署步骤或将部署视为完成之前，AWS AppConfig 监控 Amazon CloudWatch 警报所花的时间。如果在此期间触发了警报，AWS AppConfig 将回滚部署。您必须为 AWS AppConfig 配置权限以根据 CloudWatch 警报回滚。有关更多信息，请参阅 （可选）配置基于 CloudWatch 警报的回滚权限 (p. 5)。）

预定义的部署策略

AWS AppConfig 包括预定义的部署策略，以帮助您快速部署配置。您可以在部署配置时选择以下选项之一，而不是创建自己的策略。

部署策略	Description
AppConfig.AllAtOnce	快速：

部署策略	Description
	此策略会立即将配置部署到所有目标。系统监视 Amazon CloudWatch 警报达 10 分钟。如果此时未收到任何警报，则部署已完成。如果在此期间触发了警报，AppConfig 将回滚部署。
AppConfig.Linear50PercentEvery30Seconds	测试/演示： 此策略每 30 秒将配置部署到所有目标的一半，以进行一分钟部署。系统监视 Amazon CloudWatch 警报达 1 分钟。如果此时未收到任何警报，则部署已完成。如果在此期间触发了警报，AppConfig 将回滚部署。 我们建议仅将此策略用于测试或演示目的，因为它持续时间和烘焙时间短。
AppConfig.Canary10Percent20Minutes	AWS 推荐： 此策略在 20 分钟内使用 10% 的增长系数以指数方式处理部署。系统监视 Amazon CloudWatch 警报达 10 分钟。如果此时未收到任何警报，则部署已完成。如果在此期间触发了警报，AppConfig 将回滚部署。 我们建议将此策略用于生产部署，因为它与配置部署的 AWS 最佳实践保持一致。

创建部署策略

您最多可以创建 20 个部署策略。在部署配置时，您可以选择最适合应用程序和环境的部署策略。

可以使用以下过程通过 AWS Systems Manager 控制台创建 AWS AppConfig 部署策略。

创建部署策略

1. Open the AWS Systems Manager console at https://console.amazonaws.cn/systems-manager/appconfig/.
2. 在导航窗格中，选择 AWS AppConfig。
3. 选择 Deployment Strategies (部署策略) 选项卡，然后选择 Create deployment strategy (创建部署策略)。
4. 对于 Name (名称)，请输入部署策略的名称。
5. 对于 Description (描述)，请输入有关部署策略的信息。
6. 对于 Deployment type (部署类型)，选择类型。
7. 对于 Step percentage (步骤百分比)，请选择在部署的每个步骤中作为目标的调用方百分比。
8. 对于 Deployment time (部署时间)，请输入部署的总持续时间（以分钟或小时为单位）。
9. 对于 Bake time (烘焙时间)，请输入在执行部署的下一步或将部署视为完成之前监控 Amazon CloudWatch 警报所花的总时间（以分钟或小时为单位）。
10. 在 Tags (标签) 部分中，输入一个键和可选的值。您最多可以为一个资源指定 50 个标签。
11. 选择 Create deployment strategy (创建部署策略)。

Important

如果您为 AWS CodePipeline，然后您必须在 CodePipeline 指定 AWS AppConfig 作为 部署提供商. 您不需要执行 步骤 5. 部署配置 (p. 23). 但是，您必须配置客户端接收应用程序配置更新，详见 步骤 6. 接收配置 (p. 23). 有关创建指定管道的信息 AWS AppConfig 作为部署提供商，请参阅 教程: 创建使用的管道 AWS AppConfig 作为部署提供商 在 AWS CodePipeline 用户指南.

继续执行步骤 5. 部署配置 (p. 23)。

步骤 5. 部署配置

开始部署 AWS AppConfig 呼叫 开始部署 API操作。该调用包括 AWS AppConfig 应用程序、环境和配置文件的 ID 以及（可选）要部署的配置数据版本。该调用还包括要使用的部署策略的 ID，该策略确定如何部署配置数据。

AWS AppConfig 监控分发到所有主机的过程并报告状态。如果分发失败，AWS AppConfig 将回滚配置。

部署配置

可以使用以下过程通过 AWS Systems Manager 控制台部署 AWS AppConfig 配置。

使用控制台部署配置

1. Open the AWS Systems Manager console at https://console.amazonaws.cn/systems-manager/appconfig/.
2. 在导航窗格中，选择 AWS AppConfig。
3. 在 Applications (应用程序) 选项卡上，选择一个应用程序，然后选择 View details (查看详细信息)。
4. 在 Environments (环境) 选项卡上，选择一个环境，然后选择 View details (查看详细信息)。
5. 选择开始部署。
6. 对于 Configuration (配置)，请从列表中选择一个配置。
7. 根据配置的来源，使用 Document version (文档版本) 或 Parameter version (参数版本) 列表选择要部署的版本。
8. 对于 Deployment strategy (部署策略)，请从列表中选择一个策略。
9. 对于 Deployment description (部署描述)，请输入描述。
10. 在 Tags (标签) 部分中，输入一个键和可选的值。您最多可以为一个资源指定 50 个标签。
11. 选择开始部署。

步骤 6. 接收配置

您必须将客户端配置为与 GetConfiguration API 操作集成以接收配置更新。您可以使用 AWS 开发工具包进行集成。以下 AWS CLI 命令说明了如何接收配置。该调用包括 AWS AppConfig 应用程序、环境和配置文件的 ID 以及唯一的客户端 ID。配置内容将保存到输出文件名中。

Note

以下命令中的 `client-id` 参数是用户指定的唯一 ID，用于标识配置的客户端。此ID启用 AWS AppConfig 按照部署策略中定义的间隔部署配置。

```
aws appconfig get-configuration \
    --application application_name_or_ID \
    --environment environment_name_or_ID \
```

```
    --configuration configuration_profile_name_or_ID \
    --client-id client_ID \
output_filename
```

系统使用以下格式的信息进行响应。

```
{
   "ConfigurationVersion":"configuration version",
   "ContentType":"content type"
}
```

Important

注意以下关于 GetConfiguration API操作:

- TheThethe GetConfiguration 响应包括 Content 显示配置数据的部分。TheThethe Content 仅当系统发现新的或更新的配置数据时，才会显示部分。如果系统未找到新的或更新的配置数据，则 Content 未返回部分（Null）。
- AWS AppConfig 使用 ClientConfigurationVersion 参数的值来标识客户端上的配置版本。如果您没有在每次调用 GetConfiguration 时发送 ClientConfigurationVersion，则您的客户端会收到当前配置。您的客户端每次收到配置时，您都需要付费。
- 为避免超额费用，我们建议您将 ClientConfigurationVersion 每次拜访 GetConfiguration。此值必须保存在您的客户端。对 GetConfiguration 的后续调用必须通过使用 ClientConfigurationVersion 参数传递此值，如下所示。

在随后的配置更新轮询期间发送 ConfigurationVersion 类似于 HTTP ETag 概念。

```
aws appconfig get-configuration \
    --application application_name_or_ID \
    --environment environment_name_or_ID \
    --configuration configuration_profile_name_or_ID \
    --client-configuration-version previous_configuration_version_value \
    --client-id client_ID \
output_filename
```

Note

我们建议您根据预算、配置部署的预期频率以及配置目标数以调整 GetConfiguration API 调用的轮询频率。

AWS AppConfig 用户指南文档历史记录

下表列出了自 AWS AppConfig 上一次发布以来对文档所做的重要更改。

当前API版本: 2019-10-09

update-history-change	update-history-description	update-history-date
推出 AWS AppConfig 用户指南 (p. 25)	使用 AWS AppConfig， AWS Systems Manager，以创建、管理和快速部署应用配置。 AWS AppConfig 支持对任何规模的应用进行控制部署，包括内置验证检查和监控。您可以将 AWS AppConfig 与 EC2 实例上托管的应用程序、AWS Lambda、容器、移动应用程序或 IoT 设备一起使用。	July 31, 2020

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 AWS 词汇表。

如果我们为英文版本指南提供翻译，那么如果存在任何冲突，将以英文版本指南为准。在提供翻译时使用机器翻译。