Amazon Tools for PowerShell

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。

Table of Contents

什么是 Amazon Tools for PowerShell？

Amazon Tools for PowerShell 是根据 Amazon SDK for .NET 公开的功能构建的一组 PowerShell 模块。Amazon Tools for PowerShell使您可以从 PowerShell 命令行在 Amazon 资源上为操作编写脚本。

这些 cmdlet 提供了惯用的 PowerShell 体验来指定参数和处理结果，即使它们是使用各种不同的 Amazon 服务 HTTP 查询 API 实现的也是如此。例如，面向 Amazon Tools for PowerShell 的 cmdlet 支持 PowerShell 管道传输，即您可以将 PowerShell 对象以管道形式传入和传出 cmdlet。

Amazon Tools for PowerShell 可以灵活地使用它们来处理凭证，包括对 Amazon Identity and Access Management (IAM) 基础设施的支持。您可以将这些工具与 IAM 用户凭证、临时安全令牌和 IAM 角色一起使用。

Amazon Tools for PowerShell支持的服务和 Amazon 区域集与开发工具包支持的服务和区域集相同。您可以在运行 Windows、Linux 或 macOS 操作系统的计算机上安装Amazon Tools for PowerShell。

> Note
>
> Amazon Tools for PowerShell版本 4 是最新的主要版本，是对Amazon Tools for PowerShell版本 3.3 的向后兼容更新。它在维护现有 cmdlet 行为的同时添加了大量的改进功能。升级到新版本后，您的现有脚本应继续正常工作，但我们建议您在升级之前实施全面测试。有关版本 4 中更改的更多信息，请参阅从Amazon Tools for PowerShell版本 3.3 迁移到版本 4 (p. 16)。

Amazon Tools for PowerShell提供以下三个不同的程序包：

- `AWS.Tools` (p. 1)
- AWSPowerShell.NetCore (p. 2)
- AWSPowerShell (p. 2)

开发工具包主要版本的维护和支持

有关维护和支持开发工具包主要版本及其基础依赖关系的信息，请参阅Amazon开发工具包和工具引用指南中的以下内容：

- Amazon 开发工具包和工具维护策略
- Amazon 开发工具包和工具版本支持矩阵

`AWS.Tools` – Amazon Tools for PowerShell 的模块化版本的

这个Amazon Tools for PowerShell版本是针对在生产环境中运行 PowerShell 的任何计算机推荐的版本。因为它是模块化的，所以您只需要为所使用的服务下载和加载模块。这样可以减少下载时间和内存使用量，并且可以启用自动导入 `AWS.Tools` cmdlet，不过需要先手动调用 `Import-Module`。

这是Amazon Tools for PowerShell的最新版本，可在所有支持的操作系统上运行，包括 Windows、Linux 和 macOS。该程序包提供了一个安装模块 `AWS.Tools.Installer`、一个通用模块 `AWS.Tools.Common`，

并为每个 Amazon 服务提供了一个模块，例如：`AWS.Tools.EC2`、`AWS.Tools.IAM`、`AWS.Tools.S3` 等等。

该 `AWS.Tools.Installer` 模块提供 cmdlet，使您能够安装、更新和删除各个 Amazon 服务的模块。此模块中的 cmdlet 会自动确保您拥有支持所要使用模块所需的全部依赖模块。

`AWS.Tools.Common` 模块提供了用于配置和身份验证的 cmdlet，这些 cmdlet 不是服务特定的。要对 Amazon 服务使用 cmdlet，只需运行此命令即可。PowerShell 会自动导入 `AWS.Tools.Common` 模块以及要运行其 cmdlet 的 Amazon 服务的模块。如果您使用 `AWS.Tools.Installer` 模块来安装服务模块，则会自动安装此模块。

您可以在正在运行以下软件的计算机上安装此版本的Amazon Tools for PowerShell：

- 在 Windows、Linux 或 macOS 上运行 PowerShell Core 6.0 或更高版本。
- 在具有 .NET Framework 4.7.2 或更高版本的 Windows 上运行 Windows PowerShell 5.1 或更高版本。

在本指南中，仅当需要指定此版本时，我们才以其模块名称进行引用：*`AWS.Tools`*。

AWSPowerShell.NetCore – Amazon Tools for PowerShell 的单一模块版本

此版本由单个大模块组成，其中包含对所有 Amazon 服务的支持。您必须先手动导入此模块，然后才能使用该模块。

您可以在正在运行以下软件的计算机上安装此版本的Amazon Tools for PowerShell：

- 在 Windows、Linux 或 macOS 上运行 PowerShell Core 6.0 或更高版本。
- 在具有 .NET Framework 4.7.2 或更高版本的 Windows 上运行 Windows PowerShell 3.0 或更高版本。

在本指南中，仅当需要指定此版本时，我们才以其模块名称进行引用：AWSPowerShell.NetCore。

AWSPowerShell – 适用于 Windows PowerShell 的单模块版本

此版本的 Amazon Tools for PowerShell 与运行 Windows PowerShell 版本 2.0 到 5.1 的 Windows 计算机兼容，并且只能在这些计算机上安装。它与 PowerShell Core 6.0 或更高版本或任何其他操作系统（Linux 或 macOS）不兼容。此版本由单个大模块组成，其中包含对所有 Amazon 服务的支持。

在本指南中，仅当需要指定此版本时，我们才以其模块名称进行引用：AWSPowerShell。

如何使用本指南

本指南分为以下几个主要部分。

安装 Amazon Tools for PowerShell (p. 4)

此部分介绍如何安装Amazon Tools for PowerShell。它包括如何注册Amazon（如果您还没有账户），以及如何创建可用于运行 cmdlet 的 IAM 用户。

Amazon Tools for Windows PowerShell入门 (p. 22)

此部分介绍使用Amazon Tools for PowerShell的基础知识，例如指定凭证和 Amazon 区域、查找特定服务的 cmdlet 以及为 cmdlet 使用别名。

使用 Amazon Tools for PowerShell (p. 45)

此部分包含有关使用Amazon Tools for PowerShell执行一些最常见的 Amazon 任务的信息。

安装 Amazon Tools for PowerShell

要成功安装和使用Amazon Tools for PowerShell cmdlet，请参阅以下主题中的步骤。

主题

设置Amazon Tools for PowerShell的先决条件

要使用Amazon Tools for PowerShell，您必须首先完成以下步骤。

1. 注册 Amazon 账户。

   如果您没有 Amazon 账户，请参阅以下主题了解有关如何注册的完整说明：

   http://www.amazonaws.cn/premiumsupport/knowledge-center/create-and-activate-aws-account/

2. 创建 IAM 用户。

   注册您的账户后，您必须在 Amazon Identity and Access Management (IAM) 服务中创建用户。每个用户都有自己的凭证和权限。凭证用于对发出请求的用户进行身份验证。权限决定为该用户授权的 Amazon 资源和操作。

   创建用户不在本主题的范围之内。如果您不熟悉 Amazon，我们建议您先阅读以下内容：

   - 要了解用户凭证和管理凭证最佳实践，请参阅亚马逊云科技一般参考中的Amazon安全凭证。
   - 有关创建可用来运行 Amazon Tools for PowerShell 命令的具有“管理员”权限的用户的分步教程，请参阅 IAM 用户指南中的创建您的第一个 IAM 管理员用户和组。

3. 为您的 IAM 用户创建访问密钥。

   Amazon Tools for PowerShell要求使用适当的安全凭证发送每个 cmdlet。为此，您通常必须为需要使用Amazon Tools for PowerShell cmdlet 的每个用户创建访问密钥。访问密钥由访问密钥 ID 和秘密访问密钥组成。这些用于签署（对加密进行加密）您对 Amazon 服务发出的编程请求。如果没有访问密钥，您可以在 https://console.amazonaws.cn/iam/ 中使用 IAM 控制台进行创建。如Amazon安全凭证中所述，我们建议您使用 IAM 用户访问密钥，而非Amazon根账户访问密钥。IAM 让您可以安全地控制对您Amazon账户中的Amazon服务和资源的访问。

   与任何 Amazon 操作一样，创建访问密钥要求您拥有执行相关 IAM 操作的权限。有关更多信息，请参阅 IAM 用户指南中的用于管理 IAM 身份的权限。

   在 Amazon 控制台中为第一个用户创建访问密钥后，可以使用该用户及其访问密钥运行Amazon Tools for PowerShell cmdlet 来为其他用户创建访问密钥。以下示例演示如何使用 `New-IAMAccessKey` cmdlet 为 IAM 用户创建访问密钥和秘密密钥。

```
PS > New-IAMAccessKey -UserName alice

AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 9/4/19 12:46:18 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
```

```
UserName        : alice
```

将这些凭证保存在安全的位置。您需要它们以便稍后配置 Amazon Tools for PowerShell 凭证文件。有关更多信息，请参阅使用 Amazon 凭证 (p. 22)。

> **Important**
>
> 只有在创建访问密钥时才能看到秘密访问密钥（相当于密码）。您以后无法检索它。如果您丢失秘密密钥，则必须删除访问密钥/秘密密钥对并重新创建它们。

IAM 用户在任何时候都只能具有两个访问密钥。如果您尝试创建第三组凭证，则 `New-IAMAccessKey` cmdlet 会返回错误。要创建另一个访问密钥，必须首先删除现有两个中的一个。

您可以使用 `Remove-IAMAccessKey` cmdlet 为 IAM 用户删除一组凭证。您必须同时指定 `UserName` 和 `AccessKeyId`。

```
PS > Remove-IAMAccessKey -UserName alice -AccessKeyId -AccessKeyId AKIAIOSFODNN7EXAMPLE

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-IAMAccessKey (DeleteAccessKey)" on target
 "AKIAIOSFODNN7EXAMPLE".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

在 Windows 上安装 Amazon Tools for PowerShell

基于 Windows 的计算机可以运行以下任何 Amazon Tools for PowerShell 程序包选项：

- **`AWS.Tools`** (p. 6) - 的模块化版本。。Amazon Tools for PowerShell每个 Amazon 服务都由其自己的小模块以及共享的支持模块 `AWS.Tools.Common` 和 `AWS.Tools.Installer` 提供支持。
- AWSPowerShell.NetCore (p. 7) – Amazon Tools for PowerShell 的单一大模块版本。所有 Amazon 服务都由这个单一大模块支持。
- AWSPowerShell (p. 8) - 特定于传统 Windows 的单一大模块版本 Amazon Tools for PowerShell。所有 Amazon 服务都由这个单一大模块支持。

您选择的程序包取决于您正在运行的 Windows 的发行版和版本。

> **Note**
>
> Tools for Windows PowerShell（AWSPowerShell 模块）默认安装在所有基于 Windows 的 Amazon Machine Image (AMI) 上。

设置适用于Amazon Tools for PowerShell涉及本主题中所述的以下概括性任务。

1. 安装适合您的环境的Amazon Tools for PowerShell程序包选项。
2. 通过运行 `Get-ExecutionPolicy` cmdlet 验证是否已启用脚本执行。
3. 将Amazon Tools for PowerShell模块导入您的 PowerShell 会话。

Prerequisites

确保您满足 设置Amazon Tools for PowerShell的先决条件 (p. 4) 上列出的要求。

PowerShell 的更新版本，包括 PowerShell Core，可以从 Microsoft 下载，地址为 Microsoft 网站上的安装各种版本的 PowerShell。

在 Windows 上安装 AWS.Tools

您可以在运行 Windows 且具有 Windows PowerShell 5.1 或者 PowerShell Core 6.0 或更高版本的计算机上安装Amazon Tools for PowerShell的模块化版本。有关如何安装 PowerShell Core 的信息，请参阅 Microsoft 网站上的安装各种版本的 PowerShell。

您可以通过以下三种方式之一安装 AWS.Tools：

- 使用 AWS.Tools 模块中的 cmdlet。AWS.Tools.Installer 模块可以简化其他 AWS.Tools 模块的安装和更新。AWS.Tools.Installer 需要使用 PowerShellGet 并会自动下载和安装其更新版本。AWS.Tools.Installer 模块还会自动同步您的模块版本。当您安装或更新到一个模块的较新版本时，AWS.Tools.Installer 中的 cmdlet 会自动将所有其他 AWS.Tools 模块更新为同一版本。
- 从 AWS.Tools.zip 下载模块并将它们提取到其中一个模块文件夹中。您可以通过输出 $Env:PSModulePath 变量的值来查找模块文件夹。
- 使用 Install-Module cmdlet 从 PowerShell 库安装每个服务模块，如以下过程中所述。

使用模块安装 cmdlet 在 Windows 上安装 **AWS.Tools** 的步骤

1. 启动 PowerShell 会话。

   Note

   除非所处理的任务需要，否则我们建议您不要 以具有提升权限的管理员身份运行 PowerShell。这是因为此操作具有潜在的安全风险，并且不符合最低特权原则。

2. 要安装模块化 AWS.Tools 程序包，请运行以下命令。

```
PS > Install-Module -Name AWS.Tools.Installer

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this
 repository, change its InstallationPolicy value by running the Set-PSRepository
 cmdlet. Are you sure
 you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "N"): y
```

   如果您收到关于存储库“不受信任”的通知，系统会询问您是否仍要安装。输入 **y** 以允许 PowerShell 安装模块。为了避免在不信任存储库的情况下出现提示并安装模块，您可以使用 -Force 参数运行以下命令。

```
PS > Install-Module -Name AWS.Tools.Installer -Force
```

3. 现在，您可以使用 Amazon cmdlet，为要使用的每个 Install-AWSToolsModule 服务安装模块。例如，以下命令将安装 IAM 模块。此命令还会安装指定模块工作所需的任何依赖模块。例如，当您安装第一个 AWS.Tools 服务模块时，它还会安装 AWS.Tools.Common。这是所有 Amazon 服务模块所需的共享模块。它还会删除模块的较早版本，并将其他模块更新到相同的较新版本。

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
  Confirm
  Are you sure you want to perform this action?
  Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
 4.0.0.0".
  [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

  Installing module AWS.Tools.Common version 4.0.0.0
  Installing module AWS.Tools.EC2 version 4.0.0.0
```

```
Installing module AWS.Tools.Glacier version 4.0.0.0
Installing module AWS.Tools.S3 version 4.0.0.0

Uninstalling AWS.Tools version 3.3.618.0
Uninstalling module AWS.Tools.Glacier
Uninstalling module AWS.Tools.S3
Uninstalling module AWS.Tools.SimpleNotificationService
Uninstalling module AWS.Tools.SQS
Uninstalling module AWS.Tools.Common
```

Note

`Install-AWSToolsModule` cmdlet 从名为 `PSRepository` 的 `PSGallery` (https://www.powershellgallery.com/) 下载所有请求的模块，并将其视为可信来源。有关此 `Get-PSRepository -Name PSGallery` 的更多信息，请使用命令 `PSRepository`。

默认情况下，此命令将模块安装到 `$home\Documents\PowerShell\Modules` 文件夹中。若要为计算机的所有用户安装Amazon Tools for PowerShell，您必须在以管理员身份启动的 PowerShell 会话中运行以下命令。这会将模块安装到所有用户都可以访问的 `$env:ProgramFiles\PowerShell\Modules` 文件夹中。

```
PS > Install-AWSToolsModule AWS.Tools.IdentityManagement -Scope AllUsers
```

在 Windows 上安装 AWSPowerShell.NetCore

您可以在运行 Windows 且具有 PowerShell 版本 3 到 5.1 或 PowerShell Core 6.0 或更高版本的计算机上安装 AWSPowerShell.NetCore。有关如何安装 PowerShell Core 的信息，请参阅 Microsoft PowerShell 网站上的安装各种版本的 PowerShell。

您可以通过以下两种方式之一安装 AWSPowerShell.NetCore

- 从 AWSPowerShell.NetCore.zip 下载模块并将其提取到其中一个模块目录中。您可以通过输出 `$Env:PSModulePath` 变量的值来查找模块目录。
- 使用 `Install-Module` cmdlet 从 PowerShell 库进行安装，如以下过程中所述。

使用 Install-Module cmdlet 从 PowerShell 库安装 AWSPowerShell.NetCore 的步骤

要从 PowerShell 库安装 AWSPowerShell.NetCore，您的计算机必须运行 PowerShell 5.0 或更高版本，或者在 PowerShell 3 或更高版本上运行 PowerShellGet。运行以下命令。

```
PS > Install-Module -name AWSPowerShell.NetCore
```

如果您以管理员身份运行 PowerShell，则先前的命令将为计算机上的所有用户安装Amazon Tools for PowerShell。如果您以没有管理员权限的标准用户身份运行 PowerShell，则该命令仅针对当前用户安装 Amazon Tools for PowerShell。

要仅在当前用户具有管理员权限时为该用户安装，请按以下步骤带有 `-Scope CurrentUser` 参数集运行此命令。

```
PS > Install-Module -name AWSPowerShell.NetCore -Scope CurrentUser
```

虽然 PowerShell 3.0 或更高版本通常会在您首次在模块中运行 cmdlet 时将模块加载到 PowerShell 会话中，但 AWSPowerShell.NetCore 模块太大而无法支持此功能。相反，您必须通过运行以下命令将 AWSPowerShell.NetCore Core 模块显式加载到 PowerShell 会话中。

```
PS > Import-Module AWSPowerShell.NetCore
```

要将 AWSPowerShell.NetCore 模块自动加载到 PowerShell 会话中，请将该命令添加到您的 PowerShell 配置文件中。有关如何编辑您的 PowerShell 配置文件的更多信息，请参阅 PowerShell 文档中的关于配置文件。

在 Windows PowerShell 上安装 AWSPowerShell

您可以通过以下三种方式之一安装 Amazon Tools for Windows PowerShell：

- 从 AWSpWhell.zip 下载模块并将其提取到其中一个模块目录中。您可以通过输出 `$Env:PSModulePath` 变量的值来查找模块目录。
- 运行 Tools for Windows PowerShell 安装程序。这种安装 AWSPowerShell 的方法已弃用，建议您改用 `Install-Module`。
- 使用 `Install-Module` cmdlet 从 PowerShell 库进行安装，如以下过程中所述。

使用 Install-Module cmdlet 从 PowerShell 库安装 AWSPowerShell 的步骤

如果您的计算机运行 PowerShell 5.0 或更高版本，或者在 PowerShell 3 或更高版本上安装 PowerShellGet，则可以从 PowerShell 库安装 AWSPowerShell。您可以通过运行以下命令，从 Microsoft 的 PowerShell 库安装和更新 AWSPowerShell。

```
PS > Install-Module -Name AWSPowerShell
```

要将 AWSPowerShell 模块自动加载到 PowerShell 会话中，请将先前的 `import-module` cmdlet 添加到您的 PowerShell 配置文件中。有关如何编辑您的 PowerShell 配置文件的更多信息，请参阅 PowerShell 文档中的关于配置文件。

> **Note**
>
> Tools for Windows PowerShell 默认安装在所有基于 Windows 的 Amazon Machine Image (AMI) 上。

启用脚本执行

要加载Amazon Tools for PowerShell模块，必须启用 PowerShell 脚本执行。要启用脚本执行，请运行 `Set-ExecutionPolicy` cmdlet 以设置 `RemoteSigned` 策略。有关更多信息，请参阅 Microsoft Technet 网站上的关于执行策略。

> **Note**
>
> 此要求仅适用于运行 Windows 的计算机。`ExecutionPolicy` 安全限制不存在于其他操作系统上。

启用脚本执行

1. 需要管理员权限才能设置执行策略。如果您未以具有管理员权限的用户身份登录，请以管理员身份打开 PowerShell 会话。选择开始，然后选择所有程序。选择附件，然后选择 Windows PowerShell。右键单击 Windows PowerShell，然后在上下文菜单中选择以管理员身份运行。
2. 在命令提示符处，输入以下命令。

```
PS > Set-ExecutionPolicy RemoteSigned
```

> **Note**
>
> 在 64 位系统上，您必须单独对 32 位版本的 PowerShell (Windows PowerShell (x86)) 执行该操作。

如果您没有正确设置执行策略，则每当您尝试运行脚本（如您的配置文件）时，PowerShell 都会显示以下错误。

```
File C:\Users\username\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1 cannot
 be loaded because the execution
 of scripts is disabled on this system. Please see "get-help about_signing" for more
 details.
At line:1 char:2
+ . <<<<  'C:\Users\username\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'
    + CategoryInfo          : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException
```

Tools for Windows PowerShell 安装程序自动更新 PSModulePath 以包括其中包含 `AWSPowerShell` 模块的目录位置。

由于 `PSModulePath` 包含Amazon模块的目录的位置，因此 `Get-Module -ListAvailable` cmdlet 将显示该模块。

```
PS > Get-Module -ListAvailable

ModuleType Name                      ExportedCommands
---------- ----                      ----------------
Manifest   AppLocker                 {}
Manifest   BitsTransfer              {}
Manifest   PSDiagnostics             {}
Manifest   TroubleshootingPack       {}
Manifest   AWSPowerShell             {Update-EBApplicationVersion, Set-DPStatus, Remove-
IAMGroupPol...
```

Versioning

Amazon 定期发布Amazon Tools for PowerShell的新版本，以支持新的 Amazon 服务和功能。要确定您安装的工具版本，请运行 Get-AWSPowerShellVersion cmdlet。

```
PS > Get-AWSPowerShellVersion

Tools for PowerShell
Version 4.1.11.0
Copyright 2012-2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.7.0.12
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md

This software includes third party software subject to the following copyrights:
- Logging from log4net, Apache License
[http://logging.apache.org/log4net/license.html]
```

也可以将 `-ListServiceVersionInfo` 参数添加到 Get-AWSPowerShellVersion 命令来查看当前版本的工具支持的Amazon服务的列表。如果使用模块化 `AWS.Tools.*` 选项，则只显示当前已导入的模块。

```
PS > Get-AWSPowerShellVersion -ListServiceVersionInfo
```

```
...

Service                              Noun Prefix Module Name                          SDK
                                                                                      Assembly
                                                                                      Version
-------                              ----------- -----------
 -----------
Alexa For Business                   ALXB        AWS.Tools.AlexaForBusiness           3.7.0.11
Amplify Backend                      AMPB        AWS.Tools.AmplifyBackend             3.7.0.11
Amazon API Gateway                   AG          AWS.Tools.APIGateway                 3.7.0.11
Amazon API Gateway Management API    AGM         AWS.Tools.ApiGatewayManagementApi    3.7.0.11
Amazon API Gateway V2                AG2         AWS.Tools.ApiGatewayV2               3.7.0.11
Amazon Appflow                       AF          AWS.Tools.Appflow                    3.7.1.4
Amazon Route 53                      R53         AWS.Tools.Route53                    3.7.0.12
Amazon Route 53 Domains              R53D        AWS.Tools.Route53Domains             3.7.0.11
Amazon Route 53 Resolver             R53R        AWS.Tools.Route53Resolver            3.7.1.5
Amazon Simple Storage Service (S3) S3            AWS.Tools.S3                         3.7.0.13
...
```

要确定您运行的 PowerShell 版本，请输入 `$PSVersionTable` 以查看 $PSVersionTable 自动变量内容。

```
PS > $PSVersionTable

Name                           Value
----                           -----
PSVersion                      6.2.2
PSEdition                      Core
GitCommitId                    6.2.2
OS                             Darwin 18.7.0 Darwin Kernel Version 18.7.0: Tue Aug 20
 16:57:14 PDT 2019; root:xnu-4903.271.2~2/RELEASE_X86_64
Platform                       Unix
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0
```

在 Windows 上更新Amazon Tools for PowerShell

随着Amazon Tools for PowerShell的更新版本定期发布，您应更新在本地运行的版本。

更新模块化 `AWS.Tools`

要将 `AWS.Tools` 模块升级到最新版本，请运行以下命令。

```
PS > Update-AWSToolsModule -CleanUp
```

此命令会更新所有当前已安装的 `AWS.Tools` 模块，并在成功更新后删除其他已安装的版本。

> **Note**
>
> `Update-AWSToolsModule` cmdlet 从名为 `PSRepository` 的 `PSGallery` (https://www.powershellgallery.com/) 下载所有模块，并将其视为可信来源。有关此 `Get-PSRepository -Name PSGallery` 的更多信息，请使用命令：`PSRepository`。

更新 Tools for PowerShell Core

运行 `Get-AWSPowerShellVersion` cmdlet 以确定您正在运行的版本，并将此版本与 PowerShell 库网站中提供的 Tools for Windows PowerShell 版本进行比较。我们建议您每两到三个星期检查一次。只有在更新到支持新命令和 Amazon 服务的版本后，该支持才可用。

在安装 AWSPowerShell.NetCore 的更高版本之前，请卸载任何现有模块。在卸载现有程序包之前，请关闭任何打开的 PowerShell 会话。运行以下命令来卸载该程序包。

```
PS > Uninstall-Module -Name AWSPowerShell.NetCore -AllVersions
```

在卸载程序包后，通过运行以下命令来安装更新的模块。

```
PS > Install-Module -Name AWSPowerShell.NetCore
```

安装后，运行命令 `Import-Module AWSPowerShell.NetCore` 将更新的 cmdlet 加载到 PowerShell 会话中。

更新 Tools for Windows PowerShell

运行 `Get-AWSPowerShellVersion` cmdlet 以确定您正在运行的版本，并将此版本与 PowerShell 库网站中提供的 Tools for Windows PowerShell 版本进行比较。我们建议您每两到三个星期检查一次。只有在更新到支持新命令和 Amazon 服务的版本后，该支持才可用。

- 如果使用 `Install-Module` cmdlet 进行了安装，请运行以下命令。

```
PS > Uninstall-Module -Name AWSPowerShell -AllVersions
PS > Install-Module -Name AWSPowerShell
```

- 如果您使用 `.msi` 程序包安装程序或使用下载的 ZIP 文件来安装：
  1. 从 Tools for PowerShell 网站下载最新版本。将下载的文件名中的程序包版本号与运行 `Get-AWSPowerShellVersion` cmdlet 时获得的版本号进行比较。
  2. 如果下载版本的版本号高于您已安装的版本，请关闭所有 Tools for Windows PowerShell 控制台。
  3. 安装更新版本的 Tools for Windows PowerShell。

安装后，运行 `Import-Module AWSPowerShell` 将更新的 cmdlet 加载到 PowerShell 会话中。或者，从开始菜单运行自定义 Amazon Tools for PowerShell 控制台。

在 Linux 或 macOS 上安装Amazon Tools for PowerShell

本主题提供有关如何在 Linux 或 macOS 上安装Amazon Tools for PowerShell的说明。

设置概述

若要在 Linux 或 macOS 计算机上安装Amazon Tools for PowerShell，可以从两个程序包选项中选择：

- **`AWS.Tools`** (p. 12) – Amazon Tools for PowerShell 的模块化版本。每个 Amazon 服务都由其自己的小模块以及共享的支持模块 `AWS.Tools.Common` 提供支持。
- AWSPowerShell.NetCore (p. 13) – Amazon Tools for PowerShell 的单一大模块版本。所有 Amazon 服务都由这个单一大模块支持。

在运行 Linux 或 macOS 的计算机上设置这些事项涉及以下任务，本主题后面将详细介绍：

1. 在支持的系统上安装 PowerShell Core 6.0 或更高版本。
2. 安装 PowerShell Core 后，通过在系统 Shell 中运行 `pwsh` 来启动 PowerShell。

3. 安装 AWS.Tools 或 AWSPowerShell.NetCore。
4. 运行相应的 Import-Module cmdlet 以将模块导入到 PowerShell 会话中。
5. 运行 Initialize-AWSDefaultConfiguration cmdlet 以提供您的 Amazon 凭证。

Prerequisites

确保您满足设置Amazon Tools for PowerShell的先决条件 (p. 4)上列出的要求。

若要运行 Amazon Tools for PowerShell Core，您的计算机必须运行 PowerShell Core 6.0 或更高版本。

- 有关支持的 Linux 平台版本的列表，以及有关如何在基于 Linux 的计算机上安装最新版本 PowerShell 的信息，请参阅 Microsoft 网站上的在 Linux 上安装 PowerShell。尚未正式支持某些基于 Linux 的操作系统（如 Arch、Kali 和 Raspbian），但提供各种级别的社区支持。
- 有关支持的 macOS 版本的列表，以及有关如何在 macOS 上安装最新版本 PowerShell 的信息，请参阅 Microsoft 网站上的在 macOS 上安装 PowerShell。

在 Linux 或 macOS 上安装 AWS.Tools

您可以在运行 PowerShell Core 6.0 或更高版本的计算机上安装Amazon Tools for PowerShell的模块化版本。有关如何安装 PowerShell Core 的信息，请参阅 Microsoft PowerShell 网站上的安装各种版本的 PowerShell。

您可以通过以下三种方式之一安装 AWS.Tools：

- 使用 AWS.Tools.Installer 模块中的 cmdlet。AWS.Tools.Installer 模块简化了其他 AWS.Tools 模块的安装和更新。AWS.Tools.Installer 需要使用更新版本的 PowerShellGet 并会自动下载和安装它。AWS.Tools.Installer 模块还会自动保持您的模块版本同步。当您安装或更新到一个模块的较新版本时，AWS.Tools.Installer 中的 cmdlet 会自动将所有其他 AWS.Tools 模块更新为同一版本。
- 从 AWS.Tools.zip 下载模块并将它们提取到其中一个模块目录中。您可以通过输出 $Env:PSModulePath 变量的值来查找模块目录。
- 使用 Install-Module cmdlet 从 PowerShell 库安装每个服务模块，如以下过程中所述。

使用模块安装 cmdlet 在 Linux 或 macOS 上安装 **AWS.Tools** 的步骤

1. 通过运行以下命令启动 PowerShell Core 会话。

```
$ pwsh
```

> **Note**
>
> 除非所处理的任务需要，否则我们建议您不要 以具有提升权限的管理员身份运行 PowerShell。这是因为此操作具有潜在的安全风险，并且不符合最低特权原则。

2. 要使用 AWS.Tools 模块安装模块化的 AWS.Tools.Installer 程序包，请运行以下命令。

```
PS > Install-Module -Name AWS.Tools.Installer

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this
 repository, change its InstallationPolicy value by running the Set-PSRepository
 cmdlet. Are you sure
 you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "N"): y
```

如果您收到关于存储库 “不受信任” 的通知，系统会询问您是否仍要安装。输入 **y** 以允许 PowerShell 安装模块。为了在不信任存储库的情况下避免出现提示并安装模块，您可以运行以下命令。

```
PS > Install-Module -Name AWS.Tools.Installer -Force
```

3. 现在，您可以为要使用的每个服务安装模块。例如，以下命令将安装 IAM 模块。此命令还会安装指定模块工作所需的任何依赖模块。例如，当您安装第一个 `AWS.Tools` 服务模块时，它还会安装 `AWS.Tools.Common`。这是所有 Amazon 服务模块所需的共享模块。它还会删除模块的较早版本，并将其他模块更新到相同的较新版本。

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
Confirm
Are you sure you want to perform this action?
  Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
 4.0.0.0".
  [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

  Installing module AWS.Tools.Common version 4.0.0.0
  Installing module AWS.Tools.EC2 version 4.0.0.0
  Installing module AWS.Tools.Glacier version 4.0.0.0
  Installing module AWS.Tools.S3 version 4.0.0.0

  Uninstalling AWS.Tools version 3.3.618.0
  Uninstalling module AWS.Tools.Glacier
  Uninstalling module AWS.Tools.S3
  Uninstalling module AWS.Tools.SimpleNotificationService
  Uninstalling module AWS.Tools.SQS
  Uninstalling module AWS.Tools.Common
```

   Note

   `Install-AWSToolsModule` cmdlet 从名为 `PSRepository` 的 `PSGallery` (https://www.powershellgallery.com/) 下载所有请求的模块，并将存储库视为可信来源。有关此 `Get-PSRepository -Name PSGallery` 的更多信息，请使用命令 `PSRepository`。

默认情况下，这会将模块安装到 `$home\Documents\PowerShell\Modules` 文件夹中。若要为计算机的所有用户安装 `AWS.Tools` 模块，您必须在以管理员身份启动的 PowerShell 会话中运行以下命令。这会将模块安装到所有用户都可以访问的 `$env:ProgramFiles\PowerShell\Modules` 文件夹中。

```
PS > Install-AWSToolsModule -Name AWS.Tools.IdentityManagement -Scope AllUsers
```

在 Linux 或 macOS 上安装 AWSPowerShell.NetCore

要升级到 AWSPowerShell.NetCore 的更高版本，请按照 在 Linux 或 macOS 上更新 Amazon Tools for PowerShell (p. 16) 中的说明操作。首先卸载旧版本的 AWSPowerShell.NetCore。

您可以通过以下两种方式之一安装 AWSPowerShell.NetCore：

- 从 `AWSPowerShell.NetCore.zip` 下载模块并将其提取到其中一个模块目录中。您可以通过输出 `$Env:PSModulePath` 变量的值来查找模块目录。
- 使用 `Install-Module` cmdlet 从 PowerShell 库进行安装，如以下过程中所述。

使用 Install-Module cmdlet 在 Linux 或 macOS 上安装 AWSPowerShell.NetCore 的步骤

通过运行以下命令启动 PowerShell Core 会话。

```
$ pwsh
```

> **Note**
>
> 我们建议您不要 通过运行 `sudo pwsh` 来启动 PowerShell，这将以具有提升管理员权限的身份运行 PowerShell。这是因为此操作具有潜在的安全风险，并且不符合最低特权原则。

要从 PowerShell 库安装 AWSPowerShell.NetCore 单模块程序包，请运行以下命令。

```
PS > Install-Module -Name AWSPowerShell.NetCore

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository,
 change its InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure
 you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
```

如果您收到关于存储库 “不受信任” 的通知，系统会询问您是否仍要安装。输入 **y** 以允许 PowerShell 安装模块。为了在不信任存储库的情况下避免出现提示，您可以运行以下命令。

```
PS > Install-Module -Name AWSPowerShell.NetCore -Force
```

除非您要为计算机的所有用户安装Amazon Tools for PowerShell，否则不必以根用户身份运行此命令。为此，请在您通过 `sudo pwsh` 启动的 PowerShell 会话中运行以下命令。

```
PS > Install-Module -Scope AllUsers -Name AWSPowerShell.NetCore -Force
```

脚本执行

`Set-ExecutionPolicy` 命令在非 Windows 系统上不可用。您可以运行 `Get-ExecutionPolicy`，该命令显示在非 Windows 系统上运行的 PowerShell Core 中的默认执行策略设置为 `Unrestricted`。有关更多信息，请参阅 Microsoft Technet 网站上的关于执行策略。

由于 `PSModulePath` 包含 Amazon 模块的目录的位置，因此 `Get-Module -ListAvailable` cmdlet 将显示您安装的模块。

AWS.Tools

```
PS > Get-Module -ListAvailable

    Directory: /Users/username/.local/share/powershell/Modules

ModuleType Version    Name                                PSEdition ExportedCommands
---------- -------    ----                                --------- ----------------
Binary     3.3.563.1  AWS.Tools.Common                    Desk      {Clear-AWSHistory, Set-
AWSHistoryConfiguration, Initialize-AWSDefaultConfiguration, Clear-AWSDefaultConfigurat…
```

AWSPowerShell.NetCore

```
PS > Get-Module -ListAvailable

Directory: /Users/username/.local/share/powershell/Modules

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Binary     3.3.563.1  AWSPowerShell.NetCore
```

配置 PowerShell 控制台以使用 Amazon Tools for PowerShell Core（仅限 AWSPowerShell.NetCore）

只要您在模块中运行 cmdlet，PowerShell Core 通常会自动加载模块。但由于其大小较大，因此这不适用于 AWSPowerShell.NetCore。要开始运行 AWSPowerShell.NetCore cmdlet，您必须首先运行 `Import-Module AWSPowerShell.NetCore` 命令。`AWS.Tools` 模块中的 cmdlet 不需要此操作。

初始化您的 PowerShell 会话

在安装完 Amazon Tools for PowerShell 并在基于 Linux 或基于 macOS 的系统上启动 PowerShell 时，您必须运行 Initialize-AWSDefaultConfiguration 来指定要使用的 Amazon 访问密钥。有关 `Initialize-AWSDefaultConfiguration` 的更多信息，请参阅 使用 Amazon 凭证 (p. 22)。

> Note
>
> 在早期的 Amazon Tools for PowerShell 版本（3.3.96.0 之前）中，该 cmdlet 名为 `Initialize-AWSDefaults`。

Versioning

Amazon 定期发布Amazon Tools for PowerShell的新版本，以支持新的 Amazon 服务和功能。要确定您安装的Amazon Tools for PowerShell版本，请运行 Get-AWSPowerShellVersion cmdlet：

```
PS > Get-AWSPowerShellVersion

Tools for PowerShell
Version 4.0.123.0
Copyright 2012-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.3.103.22
Copyright 2009-2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md

This software includes third party software subject to the following copyrights:
- Logging from log4net, Apache License
[http://logging.apache.org/log4net/license.html]
```

要查看当前版本的工具支持的 Amazon 服务列表，请将 `-ListServiceVersionInfo` 参数添加到 Get-AWSPowerShellVersion cmdlet。

要确定您正在运行的 PowerShell 版本，请输入 `$PSVersionTable` 以查看 `$PSVersionTable` 自动变量的内容。

```
PS > $PSVersionTable
Name                           Value
----                           -----
PSVersion                      6.2.2
PSEdition                      Core
GitCommitId                    6.2.2
OS                             Darwin 18.7.0 Darwin Kernel Version 18.7.0: Tue Aug 20
 16:57:14 PDT 2019; root:xnu-4903.271.2~2/RELEASE_X86_64
Platform                       Unix
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

```
WSManStackVersion               3.0
```

在 Linux 或 macOS 上更新 Amazon Tools for PowerShell

随着Amazon Tools for PowerShell的更新版本定期发布，您应更新在本地运行的版本。

更新模块化 AWS.Tools。*

要将 AWS.Tools 模块升级到最新版本，请运行以下命令。

```
PS > Update-AWSToolsModule -CleanUp
```

此命令将更新所有当前安装的 AWS.Tools 模块，对于那些已成功更新的模块，将删除其早期版本。

> Note
>
> Update-AWSToolsModule cmdlet 从名为 PSRepository 的 PSGallery (https://www.powershellgallery.com/) 下载所有模块，并将其视为可信来源。有关此 Get-PSRepository -Name PSGallery 的更多信息，请使用命令 PSRepository。

更新 Tools for PowerShell Core

运行 Get-AWSPowerShellVersion cmdlet 以确定您正在运行的版本，并将此版本与 PowerShell 库网站中提供的 Tools for Windows PowerShell 版本进行比较。我们建议您每两到三个星期检查一次。只有在更新到支持新命令和 Amazon 服务的版本后，该支持才可用。

在安装 AWSPowerShell.NetCore 的更高版本之前，请卸载任何现有模块。在卸载现有程序包之前，请关闭任何打开的 PowerShell 会话。运行以下命令来卸载该程序包。

```
PS > Uninstall-Module -Name AWSPowerShell.NetCore -AllVersions
```

在卸载程序包后，通过运行以下命令来安装更新的模块。

```
PS > Install-Module -Name AWSPowerShell.NetCore
```

安装后，运行命令 Import-Module AWSPowerShell.NetCore 将更新的 cmdlet 加载到 PowerShell 会话中。

相关信息

- Amazon Tools for Windows PowerShell入门 (p. 22)
- 使用 Amazon Tools for PowerShell (p. 45)
- Amazon账户与访问密钥 (p. 20)

从Amazon Tools for PowerShell版本 3.3 迁移到版本 4

Amazon Tools for PowerShell版本 4 是对Amazon Tools for PowerShell版本 3.3 的向后兼容更新。它在维护现有 cmdlet 行为的同时添加了大量的改进功能。

升级到新版本后，您的现有脚本应继续正常工作，但我们建议您在升级生产环境之前实施全面测试。

本节介绍了相关更改，并说明了这些更改可能会对脚本造成的影响。

全新的完全模块化 AWS.Tools 版本

AWSPowerShell.NetCore 和 AWSPowerShell 程序包是整体式的。这意味着所有 Amazon 服务都在同一个模块中得到支持，这使得该模块非常庞大，并随着添加各个新的 Amazon 服务和功能变得越来越大。新 AWS.Tools 软件包被拆分成较小的模块，使您可以灵活地仅下载和安装所使用的 Amazon 服务需要的模块。该软件包包括所有其他模块所需的共享 AWS.Tools.Common 模块，以及可简化安装、更新和删除（如有必要）模块过程的 AWS.Tools.Installer 模块。

这还会在首次调用时启用自动导入 cmdlet，而无需先调用 Import-module。但是，要在调用 cmdlet 之前与关联的 .NET 对象进行交互，您仍需要调用 Import-Module 以让 PowerShell 了解相关的 .NET 类型。

例如，以下命令具有对 Amazon.EC2.Model.Filter 的引用。这种类型的引用无法触发自动导入，因此您必须先调用 Import-Module，否则命令将失败。

```
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
  InvalidOperation: Unable to find type [Amazon.EC2.Model.Filter].
```

```
PS > Import-Module AWS.Tools.EC2
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
PS > Get-EC2Instance -Filter $filter -Select Reservations.Instances.InstanceId
  i-0123456789abcdefg
  i-0123456789hijklmn
```

新 Get-AWSService cmdlet

为了帮助您发现 Amazon 模块集合中各个 AWS.Tools 服务的模块名称，您可以使用 Get-AWSService cmdlet。

```
PS > Get-AWSService
  Service : ACMPCA
  CmdletNounPrefix : PCA
  ModuleName : AWS.Tools.ACMPCA
  SDKAssemblyVersion : 3.3.101.56
  ServiceName : Certificate Manager Private Certificate Authority

  Service : AlexaForBusiness
  CmdletNounPrefix : ALXB
  ModuleName : AWS.Tools.AlexaForBusiness
  SDKAssemblyVersion : 3.3.106.26
  ServiceName : Alexa For Business
  ...
```

用于控制 Cmdlet 返回的对象的新 -Select 参数

版本 4 中的大多数 cmdlet 支持新 -Select 参数。每个 cmdlet 使用 Amazon 为您调用 Amazon SDK for .NET 服务 API。然后，Amazon Tools for PowerShell 客户端将响应转换为可在 PowerShell 脚本中使用的对象，并通过管道传递到其他命令。有时候，最终 PowerShell 对象在原始响应中提供的字段或属性数量超过了您的需要，而在其他情况下，您可能希望对象包含默认情况下不存在的响应字段或属性。通过 -Select 参数，您可以指定 cmdlet 返回的 .NET 对象中包含的内容。

例如，Get-S3Object cmdlet 调用 Amazon S3 开发工具包操作 ListObjects。该操作返回一个 ListObjectsResponse 对象。但是，默认情况下，Get-S3Object cmdlet 仅向 PowerShell 用户返回软件开发工具包响应的 S3Objects 元素。在下面的示例中，该对象是包含两个元素的数组。

```
PS > Get-S3Object -BucketName mybucket

ETag : "01234567890123456789012345678901111"
BucketName : mybucket
Key : file1.txt
LastModified : 9/30/2019 1:31:40 PM
Owner : Amazon.S3.Model.Owner
Size : 568
StorageClass : STANDARD

ETag : "01234567890123456789012345678902222"
BucketName : mybucket
Key : file2.txt
LastModified : 7/15/2019 9:36:54 AM
Owner : Amazon.S3.Model.Owner
Size : 392
StorageClass : STANDARD
```

在Amazon Tools for PowerShell版本 4 中，您可以指定 `-Select *` 返回由软件开发工具包 API 调用返回的完整 .NET 响应对象。

```
PS > Get-S3Object -BucketName mybucket -Select *
  IsTruncated    : False
  NextMarker     :
  S3Objects      : {file1.txt, file2.txt}
  Name           : mybucket
  Prefix         :
  MaxKeys        : 1000
  CommonPrefixes : {}
  Delimiter      :
```

您还可以指定所需特定嵌套属性的路径。以下示例仅返回 `Key` 数组中每个元素的 `S3Objects` 属性。

```
PS > Get-S3Object -BucketName mybucket -Select S3Objects.Key
file1.txt
file2.txt
```

在某些情况下，返回 cmdlet 参数可能非常有用。您可通过 `-Select ^ParameterName` 实现此目的。此功能取代 `-PassThru` 参数，该参数仍然可用，但已弃用。

```
PS > Get-S3Object -BucketName mybucket -Select S3Objects.Key |
>> Write-S3ObjectTagSet -Select ^Key -BucketName mybucket -Tagging_TagSet @{ Key='key';
 Value='value'}
  file1.txt
  file2.txt
```

每个 cmdlet 的参考主题确定它是否支持 `-Select` 参数。

更一致地限制输出中的项目数

早期版本的Amazon Tools for PowerShell允许您使用 `-MaxItems` 参数指定最终输出中返回对象的最大数量。

此行为已从 `AWS.Tools` 中删除。

此行为已在 AWSPowerShell.NetCore 和 AWSPowerShell 中弃用，并将在以后的版本中删除。

如果底层服务 API 支持 `MaxItems` 参数，则该参数仍可按照 API 指定正常使用。但它不再具有限制 cmdlet 输出中返回的项目数的添加行为。

要限制最终输出中返回的项目数，请将输出通过管道传递给 Select-Items cmdlet 并指定 -First *n* 参数，其中 *n* 是要包含在最终输出中的最大项目数。

```
PS > Get-S3Object -BucketName mybucket -Select S3Objects.Key | select -first 1*
file1.txt
```

并非所有 Amazon 服务都以相同的方式支持 -MaxItems，所以这消除了不一致性和有时出现的意外结果。此外，-MaxItems 与新的 -Select (p. 17) 参数结合，有时会导致混淆的结果。

更易于使用的流参数

Stream 或 byte[] 类型的参数现在可以接受 string、string[] 或 FileInfo 值。

例如，您可以使用以下任何示例。

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream '{
>> "some": "json"
>> }'
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream (ls .\some.json)
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream @('{', '"some": "json"',
 '}')
```

Amazon Tools for PowerShell 使用 UTF-8 编码将所有字符串转换为 byte[]。

按属性名称扩展管道

为了使用户体验更加一致，现在您可以通过为任意 参数指定属性名来传递管道输入。

在以下示例中，我们创建一个自定义对象，其属性具有与目标 cmdlet 的参数名称相匹配的名称。当 cmdlet 运行时，它会自动使用这些属性作为其参数。

```
PS > [pscustomobject] @{ BucketName='myBucket'; Key='file1.txt'; PartNumber=1 } | Get-
S3ObjectMetadata
```

> **Note**
>
> Amazon Tools for PowerShell早期版本中的部分属性支持此功能。版本 4 通过为所有 参数启用该功能，实现了更好的一致性。

静态通用参数

为了提高Amazon Tools for PowerShell版本 4.0 中的一致性，所有参数都是静态的。

在 Amazon Tools for PowerShell 的早期版本中，一些通用参数（例如 AccessKey、SecretKey、ProfileName 或 Region）均为动态，而其他参数为静态。这可能会产生问题，因为 PowerShell 在动态参数之前绑定静态参数。例如，假设您运行了以下命令。

```
PS > Get-EC2Region -Region us-west-2
```

早期版本的 PowerShell 将值 us-west-2 绑定到 -RegionName 静态参数而不是 -Region 动态参数。这可能会使用户感到困惑。

AWS.Tools 声明和强制执行强制性参数

现在，AWS.Tools.* 模块声明并强制执行强制性的 cmdlet 参数。当 Amazon 服务声明需要 API 的参数时，PowerShell 会提示您输入相应的 cmdlet 参数（如果您未指定该参数）。这仅适用于 AWS.Tools。为确保向后兼容性，这不适用于 AWSPowerShell.NetCore 或 AWSPowerShell。

所有参数均可为 null

您现在可以将 $null 分配给值类型参数（数字和日期）。此更改不应影响现有脚本。这使您能够绕过输入强制性参数的提示。强制性参数仅在 AWS.Tools 中强制执行。

如果您使用版本 4 运行以下示例，它会有效地绕过客户端验证，因为您为每个强制性参数提供一个“值”。但是，Amazon EC2 API 服务调用失败，因为Amazon服务仍需要该信息。

```
PS > Get-EC2InstanceAttribute -InstanceId $null -Attribute $null
WARNING: You are passing $null as a value for parameter Attribute which is marked as
 required.
In case you believe this parameter was incorrectly marked as required, report this by
 opening
an issue at https://github.com/aws/aws-tools-for-powershell/issues.
WARNING: You are passing $null as a value for parameter InstanceId which is marked as
 required.
In case you believe this parameter was incorrectly marked as required, report this by
 opening
an issue at https://github.com/aws/aws-tools-for-powershell/issues.

Get-EC2InstanceAttribute : The request must contain the parameter instanceId
```

删除以前弃用的功能

下列功能在Amazon Tools for PowerShell的以前版本中已弃用，并已在版本 4 中删除：

- 从 -Terminate cmdlet 删除了 Stop-EC2Instance 参数。请改用 Remove-EC2Instance。
- 从 Clear-AWSCredential cmdlet 删除了 -ProfileName 参数。请改用 Remove-AWSCredentialProfile。
- 删除了 cmdlet Import-EC2Instance 和 Import-EC2Volume。

Amazon账户与访问密钥

若要访问Amazon，您需要注册Amazon账户。

访问密钥包含访问密钥 ID 和秘密访问密钥，用于签署对Amazon发出的编程请求。如果没有访问密钥，可在https://console.aws.amazon.com/iam/中使用 IAM 控制台创建密钥。我们建议您使用 IAM 访问密钥而不是 Amazon 根账户访问密钥。IAM 让您可以安全地控制对您Amazon账户中的Amazon服务和资源的访问。

> Note
>
> 要创建访问密钥，您必须拥有执行所需 IAM 操作的权限。有关更多信息，请参阅 IAM 用户指南中的授予 IAM 用户管理密码策略和凭证的权限。

要获取访问密钥 ID 和秘密访问密钥

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航菜单上，选择用户。

3. 选择您的 IAM 用户名称 (而不是复选框)。
4. 打开安全凭证选项卡，然后选择创建访问密钥。
5. 要查看新的访问密钥，请选择显示。您的凭证与下面类似：
   - 访问密钥 ID：`AKIAIOSFODNN7EXAMPLE`
   - 秘密访问密钥：`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
6. 要下载密钥对，请选择下载 .csv 文件。将带有密钥的 .csv 文件存储在安全的位置。

   Important

   - 请对密钥保密以保护您的 Amazon 账户，切勿通过电子邮件发送密钥。请勿对企业外部共享密钥，即使有来自 Amazon 或 Amazon.com 的询问。合法代表 Amazon 的任何人永远都不会要求您提供密钥。
   - 只有 在最初创建密钥对时才能检索秘密访问密钥。像密码一样，您以后无法检索它。如果您丢失了此信息，则必须创建新的密钥对。

相关主题

- IAM 用户指南中的什么是 IAM？。
- 亚马逊云科技一般参考中的Amazon安全凭证。

Amazon Tools for Windows PowerShell入门

本节介绍使用 Tools for Windows PowerShell 的基础知识。例如，其中说明了如何在 Tools for Windows PowerShell 与 Amazon 交互时指定它应使用的凭证和Amazon区域。此部分还提供使用标准 PowerShell cmdlet 的指南，例如使用 `Get-Command` 发现 Amazon cmdlet。

主题

使用 Amazon 凭证

每条Amazon Tools for PowerShell命令均必须包含一组 Amazon 凭证，这些凭证用于以加密方式对相应的 Web 服务请求进行签名。您可以为每条命令、每个会话或所有会话指定凭证。

作为最佳实践，为了避免公开您的凭证，请不要在命令中输入凭证文本。相反，为要使用的每组凭证创建一个配置文件，并将该配置文件存储在两个凭证存储中的任一凭证存储中。在命令中按名称指定正确的配置文件，Amazon Tools for PowerShell将检索关联的凭证。有关如何安全地管理Amazon凭证的一般讨论，请参阅亚马逊云科技一般参考中的管理Amazon访问密钥的最佳实践。

> **Note**
>
> 您需要一个 Amazon 账户来获取凭证并使用Amazon Tools for PowerShell。有关如何注册账户的更多信息，请参阅 Amazon账户与访问密钥 (p. 20)。

主题

凭证存储位置

Amazon Tools for PowerShell可以使用两个凭证存储中的任意一个：

- Amazon 开发工具包存储，可加密您的凭证并将其存储在主文件夹中。在 Windows 中，此存储位于：`C:\Users\username\AppData\Local\AWSToolkit\RegisteredAccounts.json`。

  Amazon SDK for .NET 和 Toolkit for Visual Studio 也可以使用Amazon开发工具包存储。

- 共享的凭证文件也位于主文件夹中，但以纯文本形式存储凭证。

  默认情况下，凭证文件存储在以下位置：
  - 在 Windows 上: `C:\Users\username\.aws\credentials`
  - 在 Mac/Linux 上：`~/.aws/credentials`

  Amazon 开发工具包和 Amazon Command Line Interface 也可以使用凭证文件。如果您在 Amazon 用户上下文外部运行脚本，请确保将包含凭证的文件复制到可供所有用户账户（本地系统和用户）访问凭证的位置。

管理配置文件

配置文件允许您使用 Amazon Tools for PowerShell 引用不同的凭证集。您可以使用 Amazon Tools for PowerShell cmdlet 管理 Amazon SDK 存储中的配置文件。您也可以使用 Toolkit for Visual Studio 或使用 Amazon SDK for .NET 以编程方式，在Amazon开发工具包存储中管理配置文件。有关如何管理凭证文件中的配置文件的说明，请参阅管理Amazon访问密钥的最佳实践。

添加新的配置文件

要将新配置文件添加到Amazon开发工具包存储，请运行 `Set-AWSCredential` 命令。它将您的访问密钥和秘密密钥存储在您指定的配置文件名称下的默认凭证文件中。

```
PS > Set-AWSCredential `
                -AccessKey AKIA0123456787EXAMPLE `
                -SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY `
                -StoreAs MyNewProfile
```

- `-AccessKey` – 访问密钥 ID。
- `-SecretKey` - 私有密钥。
- `-StoreAs` - 配置文件名称，该名称必须是唯一的。要指定默认配置文件，请使用名称 `default`。

更新配置文件

必须手动维护Amazon开发工具包存储。如果您稍后更改服务凭证，例如使用 IAM 控制台进行更改 - 使用本地存储凭证运行命令失败，并出现以下错误消息：

```
The Access Key Id you provided does not exist in our records.
```

您可以通过对配置文件重复 `Set-AWSCredential` 命令并为它传递给新的访问密钥和秘密密钥来更新配置文件。

列出配置文件

您可以使用以下命令检查当前名称列表。在此示例中，名为 Shirley 的用户可以访问三个配置文件，这些配置文件都存储在共享凭证文件 (`~/.aws/credentials`) 中。

```
PS > Get-AWSCredential -ListProfileDetail

ProfileName  StoreTypeName         ProfileLocation
-----------  -------------         ---------------
default      SharedCredentialsFile /Users/shirley/.aws/credentials
production   SharedCredentialsFile /Users/shirley/.aws/credentials
test         SharedCredentialsFile /Users/shirley/.aws/credentials
```

删除配置文件

要删除不再需要的配置文件，请使用以下命令。

```
PS > Remove-AWSCredentialProfile -ProfileName an-old-profile-I-do-not-need
```

-ProfileName 参数指定要删除的配置文件。

弃用的命令 Clearar-AWScredential 仍可用于向后兼容，但首选使用 Remove-AWSCredentialProfile。

指定凭证

可通过多种方式指定凭证。首选方法是确定配置文件，而不是将文字凭证合并到命令行中。Amazon Tools for PowerShell使用凭证搜索顺序 (p. 25)中描述的搜索顺序查找配置文件。

在 Windows 上，存储在Amazon开发工具包存储中的Amazon凭证使用已登录的 Windows 用户身份进行加密。它们不能通过使用另一个账户进行解密，也不能在与最初创建它们的设备不同的设备上使用。要执行需要另一个用户的凭证（例如，将在其下运行计划任务的用户账户）的任务，请设置一个凭证配置文件（如上一部分中所述），您可在以该用户身份登录到计算机时使用该配置文件。以执行任务的用户身份登录以完成凭证设置步骤，并创建适用于该用户的配置文件。然后注销，并使用您自己的凭证重新登录以设置计划的任务。

> **Note**
>
> 使用 -ProfileName 通用参数指定配置文件。此参数等同于早期-StoredCredentials版本中的 Amazon Tools for PowerShell 参数。为了实现向后兼容性，仍支持 -StoredCredentials。

默认配置文件（推荐）

如果凭证存储在名为 Amazon 的配置文件中，所有 default 开发工具包和管理工具可以自动在您的本地计算机上查找您的凭证。例如，如果您在本地计算机上有一个名为 default 的配置文件，则不必运行 Initialize-AWSDefaultConfiguration cmdlet 或 Set-AWSCredential cmdlet。这些工具会自动使用该配置文件中的访问权限和秘密密钥数据。要使用默认区域之外的 Amazon 区域（Get-DefaultAWSRegion 的结果），您可以运行 Set-DefaultAWSRegion 并指定一个区域。

如果您的配置文件未命名为 default，但您要将其用作当前会话的默认配置文件，请运行 Set-AWSCredential 以将其设置为默认配置文件。

尽管运行 Initialize-AWSDefaultConfiguration 会让您为每个 PowerShell 会话指定默认配置文件，cmdlet 从您自定义命名的配置文件加载凭证，但会使用指定的配置文件覆盖 default 配置文件。

我们建议您不要运行 Initialize-AWSDefaultConfiguration，除非您在未使用实例配置文件启动的 Amazon EC2 实例上运行 PowerShell 会话，并且您希望手动设置凭证配置文件。请注意，在这种情况下凭证配置文件不包含凭证。在 EC2 实例上运行 Initialize-AWSDefaultConfiguration 而产生的凭证配置文件不会直接存储凭证，而是指向实例元数据（提供自动轮换的临时凭证）。但是，它确实存储实例的区域。如果您希望针对实例运行所在区域之外的区域运行调用，则会发生可能需要运行 Initialize-AWSDefaultConfiguration 的另一个场景。运行该命令将永久覆盖存储在实例元数据中的区域。

```
PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-west-2
```

> **Note**
>
> 默认凭证包含在 default 配置文件名下的Amazon开发工具包存储中。该命令将覆盖任何具有该名称的现有配置文件。

如果您的 EC2 实例是使用实例配置文件启动的，PowerShell 会从实例配置文件获取 Amazon 凭证和区域信息。您不需要运行 Initialize-AWSDefaultConfiguration。在使用实例配置文件启动的 EC2 实例上运行 Initialize-AWSDefaultConfiguration cmdlet 不是必需的，因为它使用 PowerShell 在默认情况下已经使用的相同实例配置文件数据。

会话配置文件

使用 `Set-AWSCredential` 指定特定会话的默认配置文件。此配置文件将在会话持续时间内覆盖任何默认配置文件。如果您想要在会话中使用自定义命名的配置文件，而不是当前 `default` 配置文件，我们建议您使用会话配置文件。

```
PS > Set-AWSCredential -ProfileName MyProfileName
```

> **Note**
>
> 在 Tools for Windows PowerShell 1.1 之前的版本中，`Set-AWSCredential` cmdlet 无法正常工作并覆盖“MyProfileName”指定的配置文件。我们建议使用更新版本的 Tools for Windows PowerShell。

命令配置文件

在单个命令上，您可以添加 `-ProfileName` 参数以指定仅适用于该一个命令的配置文件。此配置文件将覆盖任何默认配置文件或会话配置文件，如以下示例所示。

```
PS > Get-EC2Instance -ProfileName MyProfileName
```

> **Note**
>
> 当您指定默认配置文件或会话配置文件时，也可以添加 `-Region` 参数以覆盖默认区域或会话区域。有关更多信息，请参阅指定 Amazon 区域 (p. 30)。以下示例指定默认配置文件和区域。
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-
> west-2
> ```

默认情况下，假定 Amazon 共享凭证文件位于用户的主文件夹（Windows：`C:\Users\username\.aws`；或者 Linux：`~/.aws`）中。要在其他位置指定凭证文件，请包含 `-ProfileLocation` 参数并指定凭证文件路径。以下示例为特定命令指定非默认凭证文件。

```
PS > Get-EC2Instance -ProfileName MyProfileName -ProfileLocation C:\aws_service_credentials
\credentials
```

> **Note**
>
> 如果您在通常不会登录Amazon的时段内运行 PowerShell 脚本（例如，您在非正常工作时间内将 PowerShell 脚本作为计划任务运行），请在您指定要使用的配置文件时添加 `-ProfileLocation` 参数，并将值设置为用于存储凭证的文件的路径。要确保您的Amazon Tools for PowerShell脚本使用正确的账户凭证运行，那么只要您的脚本在未使用 `-ProfileLocation` 账户的上下文或过程中运行，您均应添加 Amazon 参数。您也可以将凭证文件复制到供脚本用来执行任务的本地系统或其他账户可访问的位置。

凭证搜索顺序

运行命令时，Amazon Tools for PowerShell 按照以下顺序搜索凭证。它在找到可用凭证时停止。

1. 作为参数嵌入在命令行中的文字凭证。

   我们强烈建议您使用配置文件，而不是将文字凭证输入到命令行中。
2. 指定的配置文件名称或配置文件位置。
   - 如果您仅指定配置文件名称，此命令将在Amazon开发工具包存储中查找指定的配置文件；如果该配置文件不存在，则使用默认位置的Amazon共享凭证文件中的指定配置文件。
   - 如果您仅指定配置文件位置，此命令将从该凭证文件中查找 `default` 配置文件。

- 如果同时指定名称和位置，则该命令将在该凭证文件中查找指定的配置文件。

如果未找到指定的配置文件或位置，则命令会引发异常。仅当您尚未指定配置文件或位置时，搜索才会继续执行以下步骤。

3. `-Credential` 参数指定的凭证。
4. 会话配置文件（如果存在）。
5. 按以下顺序使用默认配置文件：
   a. `default` 开发工具包存储中的 Amazon 配置文件。
   b. `default` 共享凭证文件中的 Amazon 配置文件。
   c. `AWS PS Default` 开发工具包存储中的 Amazon 配置文件。
6. 如果命令在配置为使用 IAM 角色的 Amazon EC2 实例上运行，那么将从实例配置文件访问 EC2 实例的临时凭证。

   有关针对 Amazon EC2 实例使用 IAM 角色的更多信息，请参阅 Amazon SDK for .NET。

如果此搜索未能找到指定的凭证，则该命令会引发异常。

Amazon Tools for PowerShell Core 中的证书处理

Amazon Tools for PowerShell Core 中的 cmdlet 在运行时接受 Amazon 访问密钥和秘密密钥或凭证配置文件的名称，这与 Amazon Tools for Windows PowerShell 类似。当它们在 Windows 上运行时，这两个模块都能够访问 Amazon SDK for .NET 凭证存储文件（存储在每用户 `AppData\Local\AWSToolkit\RegisteredAccounts.json` 文件中）。

该文件以加密形式存储您的密钥，并且无法在其他计算机上使用。它是Amazon Tools for PowerShell在凭证配置文件中搜索到的第一个文件，并且也是Amazon Tools for PowerShell将凭证配置文件存储到的文件。有关 Amazon SDK for .NET 凭证存储文件的更多信息，请参阅配置Amazon凭证。Tools for Windows PowerShell 模块当前不支持将凭证写入其他文件或位置。

这两个模块都可以从由其他 Amazon 开发工具包和 Amazon 使用的 Amazon CLI 共享凭证文件中读取配置文件。在 Windows 上，此文件的默认位置是 `C:\Users\<userid>\.aws\credentials`。在非 Windows 平台上，此文件存储在 `~/.aws/credentials`。`-ProfileLocation` 参数可用于指向非默认文件名或文件位置。

开发工具包凭证存储使用 Windows 加密 API 来以加密形式保存您的凭证。这些 API 在其他平台上不可用，因此 Amazon Tools for PowerShell Core 模块以独占方式使用 Amazon 共享凭证文件，并支持将新的凭证配置文件写入共享凭证文件中。

以下使用 `Set-AWSCredential` cmdlet 的示例脚本说明了用于使用 AWSPowerShell 或 AWSPowerShell.NetCore 模块处理 Windows 上的凭证配置文件的选项。

```
# Writes a new (or updates existing) profile with name "myProfileName"
# in the encrypted SDK store file

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName

# Checks the encrypted SDK credential store for the profile and then
# falls back to the shared credentials file in the default location

Set-AWSCredential -ProfileName myProfileName

# Bypasses the encrypted SDK credential store and attempts to load the
# profile from the ini-format credentials file "mycredentials" in the
# folder C:\MyCustomPath

Set-AWSCredential -ProfileName myProfileName -ProfileLocation C:\MyCustomPath\mycredentials
```

以下示例说明 AWSPowerShell.NetCore 模块在 Linux 或 Mac OS X 操作系统上的行为。

```
# Writes a new (or updates existing) profile with name "myProfileName"
# in the default shared credentials file ~/.aws/credentials

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName

# Writes a new (or updates existing) profile with name "myProfileName"
# into an ini-format credentials file "~/mycustompath/mycredentials"

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName -ProfileLocation
 ~/mycustompath/mycredentials

# Reads the default shared credential file looking for the profile "myProfileName"

Set-AWSCredential -ProfileName myProfileName

# Reads the specified credential file looking for the profile "myProfileName"

Set-AWSCredential -ProfileName myProfileName -ProfileLocation ~/mycustompath/mycredentials
```

在 Amazon Tools for PowerShell 中共享凭证

Tools for Windows PowerShell 支持使用Amazon共享凭证文件，类似于 Amazon CLI 和其他Amazon开发工具包。Tools for Windows PowerShell 现在支持在 .NET 凭证文件和Amazon共享凭证文件中读取和写入 `basic`、`session` 和 `assume role` 凭证配置文件。新的 `Amazon.Runtime.CredentialManagement` 命名空间支持此功能。

已添加到凭证相关 cmdlet Initialize-AWSDefaultConfiguration、New-AWSCredential 和 Set-AWSCredential 的以下参数支持新配置文件类型和Amazon共享凭证文件访问。在服务 cmdlet 中，您可以通过添加通用参数 `-ProfileName` 来引用配置文件。

将 IAM 角色与 Amazon Tools for PowerShell 结合使用

Amazon 共享凭证文件启用其他类型的访问。例如，您可以使用 IAM 角色而不是 IAM 用户的长期凭证来访问您的Amazon资源。为此，您必须有一个具有代入该角色的权限的标准配置文件。当您告知 Amazon Tools for PowerShell 使用指定角色的配置文件时，Amazon Tools for PowerShell将查找由 `SourceProfile` 参数标识的配置文件。这些凭证用于为由 `RoleArn` 参数指定的角色请求临时凭证。当此角色由第三方代入时，您可以选择要求使用多重身份验证 (MFA) 设备或 `ExternalId` 代码。

参数名称	描述
ExternalId	代入角色时要使用的用户定义的外部 ID (如果角色需要)。通常仅在将账户的访问权限委派给第三方时才需要。代入指定角色时，第三方必须包含 ExternalId 作为参数。有关更多信息，请参阅 IAM 用户指南中的如何在向第三方授予对Amazon资源的访问权时使用外部 ID。
MfaSerial	代入角色时要使用的 MFA 序列号 (如果角色需要)。有关更多信息，请参阅 IAM 用户指南中的在Amazon中使用多重身份验证 (MFA)。
RoleArn	要代入的角色的 ARN，用于代入角色凭证。有关创建和使用 IAM 角色的更多信息，请参阅 IAM 用户指南中的 IAM 角色。

参数名称	描述
SourceProfile	代入角色凭证要使用的源配置文件的名称。在此配置文件中找到的凭证用于代入由 RoleArn 参数指定的角色。

代入角色的配置文件设置

以下示例显示如何设置可直接代入 IAM 角色的源配置文件。

第一个命令创建由角色配置文件引用的源配置文件。第二个命令创建要代入哪个角色的角色配置文件。第三个命令显示角色配置文件的凭证。

```
PS > Set-AWSCredential -StoreAs my_source_profile -AccessKey access_key_id -
SecretKey secret_key
PS > Set-AWSCredential -StoreAs my_role_profile -SourceProfile my_source_profile -
RoleArn arn:aws:iam::123456789012:role/role-i-want-to-assume
PS > Get-AWSCredential -ProfileName my_role_profile

SourceCredentials                   RoleArn
 RoleSessionName                            Options
-----------------                   -------
 ---------------                            -------
Amazon.Runtime.BasicAWSCredentials arn:aws:iam::123456789012:role/role-i-want-to-assume
 aws-dotnet-sdk-session-636238288466144357 Amazon.Runtime.AssumeRoleAWSCredentialsOptions
```

要将此角色配置文件与 Tools for Windows PowerShell 服务 cmdlet 一起使用，请将 -ProfileName 通用参数添加到命令以引用角色配置文件。以下示例使用上一示例中定义的角色配置文件访问 Get-S3Bbucket cmdlet。Amazon Tools for PowerShell 在 my_source_profile 中查找凭证，使用这些凭证代表用户调用 AssumeRole，然后使用这些临时角色凭证调用 Get-S3Bucket。

```
PS > Get-S3Bucket -ProfileName my_role_profile

CreationDate           BucketName
------------           ----------
2/27/2017 8:57:53 AM   4ba3578c-f88f-4d8b-b95f-92a8858dac58-bucket1
2/27/2017 10:44:37 AM  2091a504-66a9-4d69-8981-aaef812a02c3-bucket2
```

使用凭证配置文件类型

要设置凭证配置文件类型，需要了解哪些参数提供配置文件类型所需的信息。

凭证类型	必须使用的参数
基本 这些是 IAM 用户的长期凭证	-AccessKey -SecretKey
会话 这些是您手动检索的 IAM 角色的短期凭证，例如通过直接调用 Use-STSRole cmdlet。	-AccessKey -SecretKey -SessionToken
角色	-SourceProfile

凭证类型	必须使用的参数
这些是 Amazon Tools for PowerShell 为您检索的 IAM 角色的短期凭证。	`-RoleArn` 可选：`-ExternalId` 可选：`-MfaSerial`

ProfilesLocation 通用参数

您可以使用 `-ProfileLocation` 写入共享凭证文件以及指示 cmdlet 从凭证文件中读取。添加 `-ProfileLocation` 参数可以控制 Tools for Windows PowerShell 是使用共享凭证文件还是 .NET 凭证文件。下表描述了参数在 Tools for Windows PowerShell 中的工作方式。

配置文件位置值	配置文件解析行为
null (未设置) 或空	首先，在 .NET 凭证文件中搜索具有指定名称的配置文件。如果找不到该配置文件，则在以下位置搜索 Amazon 共享凭证文件：*(user's home directory)*`\.aws\credentials`。
Amazon 共享凭证文件格式的文件的路径	仅在指定文件中搜索具有给定名称的配置文件。

将凭证保存到凭证文件

要编写凭证并将其保存到两个凭证文件中的一个，请运行 `Set-AWSCredential` cmdlet。下面的示例演示了具体做法。第一个命令使用 `Set-AWSCredential` 以及 `-ProfileLocation` 向由 `-ProfileName` 参数指定的配置文件添加访问密钥和秘密密钥。在第二行中，运行 Get-Content cmdlet 以显示凭证文件内容。

```
PS > Set-AWSCredential -ProfileLocation C:\Users\auser\.aws\credentials -ProfileName
 basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\auser\.aws\credentials

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

显示您的凭证配置文件

运行 Get-AWSCredential cmdlet 并添加 `-ListProfileDetail` 参数以返回凭证文件类型和位置以及配置文件名称列表。

```
PS > Get-AWSCredential -ListProfileDetail

ProfileName                    StoreTypeName          ProfileLocation
-----------                    -------------          ---------------
source_profile                 NetSDKCredentialsFile
assume_role_profile            NetSDKCredentialsFile
basic_profile                  SharedCredentialsFile C:\Users\auser\.aws\credentials
```

删除凭证配置文件

要删除凭证配置文件，请运行新的 Remove-AWSCredentialProfile cmdlet。Clear-AWSCredential 已弃用，但仍可用于向后兼容。

重要提示

只有 Initialize-AWSDefaultConfiguration、New-AWSCredential 和 Set-AWSCredential 支持角色配置文件的参数。您不能直接在命令上指定角色参数，例如 `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`。这不起作用，因为服务 cmdlet 不直接支持 `SourceProfile` 或 `RoleArn` 参数。而是您必须将这些参数存储在配置文件中，然后使用 `-ProfileName` 参数调用命令。

指定 Amazon 区域

有两种方法可以指定运行 Amazon 命令时使用的 Amazon Tools for PowerShell 区域：

- 对单个命令使用 `-Region` 通用参数。
- 使用 `Set-DefaultAWSRegion` 命令为所有命令设置默认区域。

如果 Tools for Windows PowerShell 无法确定要使用的区域，则许多 Amazon cmdlet 会失败。例外情况包括适用于 Amazon S3 (p. 47)、Amazon SES 和 IAM 与 Tools for PowerShell (p. 51) 的 cmdlet，它会自动默认为全局端点。

为单个Amazon命令指定区域

将 `-Region` 参数添加到命令中，如下所示。

```
PS > Get-EC2Image -Region us-west-2
```

为当前会话中的所有 Amazon CLI 命令设置默认区域

从 PowerShell 命令提示符键入以下命令。

```
PS > Set-DefaultAWSRegion -Region us-west-2
```

> **Note**
>
> 此设置仅为当前会话保留。要将设置应用到所有 PowerShell 会话，请将该命令添加到您的 PowerShell 配置文件，就像在 `Import-Module` 命令中一样。

查看所有 Amazon CLI 命令的当前默认区域

从 PowerShell 命令提示符键入以下命令。

```
PS > Get-DefaultAWSRegion

Region    Name             IsShellDefault
------    ----             --------------
us-west-2 US West (Oregon) True
```

清除所有 Amazon CLI 命令的当前默认区域

从 PowerShell 命令提示符键入以下命令。

```
PS > Clear-DefaultAWSRegion
```

查看所有可用Amazon区域的列表

从 PowerShell 命令提示符键入以下命令。请注意，示例输出中的第三列标识您当前会话的默认区域。

```
PS > Get-AWSRegion

Region         Name                      IsShellDefault
------         ----                      --------------
ap-east-1      Asia Pacific (Hong Kong)  False
ap-northeast-1 Asia Pacific (Tokyo)      False
...
us-east-2      US East (Ohio)            False
us-west-1      US West (N. California)   False
us-west-2      US West (Oregon)          True
...
```

Note

某些区域可能受支持，但不包含在 Get-AWSRegion cmdlet 的输出中。例如，对于尚不具有全局性的区域，有时也是如此。如果您无法通过添加 -Region 参数来指定某个区域，请尝试在一个自定义端点中指定该区域，如以下部分中所述。

指定自定义或非标准终端节点

按照如下示例的格式，在您的 Tools for Windows PowerShell 命令中添加 -EndpointUrl 通用参数，以 URL 形式指定一个自定义端点。

```
PS > Some-AWS-PowerShellCmdlet -EndpointUrl "custom endpoint URL" -Other -Parameters
```

下面是一个使用 Get-EC2Instance cmdlet 的示例。在该示例中，自定义端点位于 us-west-2 或美国西部（俄勒冈）区域中，但您可以使用任何其他支持的Amazon区域，包括 Get-AWSRegion 未列举的区域。

```
PS > Get-EC2Instance -EndpointUrl "https://service-custom-url.us-west-2.amazonaws.com" -
InstanceID "i-0555a30a2000000e1"
```

Cmdlet 发现和别名

此部分介绍如何列出Amazon Tools for PowerShell支持的服务，如何显示Amazon Tools for PowerShell为支持这些服务而提供的 cmdlet 集，以及如何查找用于访问这些服务的替代 cmdlet 名称（也称为别名）。

Cmdlet 发现

所有Amazon服务操作（或 API）都记录在每项服务的 API 参考指南中。例如，请参阅 IAM API 参考。在大多数情况下，Amazon服务 API 和 Amazon PowerShell cmdlet 之间存在一对一的对应关系。要获取与 Amazon 服务 API 名称对应的 cmdlet 名称，请使用 -ApiOperation 参数和Amazon服务 API 名称运行 Amazon Get-AWSCmdletName cmdlet。例如，要获取基于任何可用的 DescribeInstances Amazon服务 API 的所有可能的 cmdlet 名称，请运行以下命令：

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

CmdletName         ServiceOperation    ServiceName                    CmdletNounPrefix
----------         ----------------    -----------                    ----------------
Get-EC2Instance    DescribeInstances   Amazon Elastic Compute Cloud   EC2
Get-GMLInstance    DescribeInstances   Amazon GameLift Service        GML
```

-ApiOperation 参数是默认参数，因此您可以省略参数名称。以下示例等同于前一个示例：

```
PS > Get-AWSCmdletName DescribeInstances
```

如果您知道 API 和 AWS 服务的名称，则可以将 `-Service` 参数以及 cmdlet 名词前缀包含为Amazon服务名称的一部分。例如，Amazon EC2 的 cmdlet 名词前缀是 `EC2`。要在 Amazon EC2 服务中获取与 `DescribeInstances` API 对应的 cmdlet 名称，请运行以下命令之一。它们都会产生相同的输出：

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service EC2
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service Compute
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service "Compute Cloud"

CmdletName        ServiceOperation    ServiceName                    CmdletNounPrefix
----------        ----------------    -----------                    ----------------
Get-EC2Instance   DescribeInstances   Amazon Elastic Compute Cloud   EC2
```

这些命令中的参数值不区分大小写。

如果您不知道所需Amazon服务 API 或Amazon服务的名称，可以使用 `-ApiOperation` 参数以及要匹配的模式和 `-MatchWithRegex` 参数。例如，要获取包含 `SecurityGroup` 的所有可用的 cmdlet 名称，请运行以下命令：

```
PS > Get-AWSCmdletName -ApiOperation SecurityGroup -MatchWithRegex

CmdletName                                      ServiceOperation
 ServiceName                           CmdletNounPrefix
----------                                      ----------------
 -----------                           ----------------
Approve-ECCacheSecurityGroupIngress             AuthorizeCacheSecurityGroupIngress
 Amazon ElastiCache                    EC
Get-ECCacheSecurityGroup                        DescribeCacheSecurityGroups
 Amazon ElastiCache                    EC
New-ECCacheSecurityGroup                        CreateCacheSecurityGroup
 Amazon ElastiCache                    EC
Remove-ECCacheSecurityGroup                     DeleteCacheSecurityGroup
 Amazon ElastiCache                    EC
Revoke-ECCacheSecurityGroupIngress              RevokeCacheSecurityGroupIngress
 Amazon ElastiCache                    EC
Add-EC2SecurityGroupToClientVpnTargetNetwrk     ApplySecurityGroupsToClientVpnTargetNetwork
 Amazon Elastic Compute Cloud          EC2
Get-EC2SecurityGroup                            DescribeSecurityGroups
 Amazon Elastic Compute Cloud          EC2
Get-EC2SecurityGroupReference                   DescribeSecurityGroupReferences
 Amazon Elastic Compute Cloud          EC2
Get-EC2StaleSecurityGroup                       DescribeStaleSecurityGroups
 Amazon Elastic Compute Cloud          EC2
Grant-EC2SecurityGroupEgress                    AuthorizeSecurityGroupEgress
 Amazon Elastic Compute Cloud          EC2
Grant-EC2SecurityGroupIngress                   AuthorizeSecurityGroupIngress
 Amazon Elastic Compute Cloud          EC2
New-EC2SecurityGroup                            CreateSecurityGroup
 Amazon Elastic Compute Cloud          EC2
Remove-EC2SecurityGroup                         DeleteSecurityGroup
 Amazon Elastic Compute Cloud          EC2
Revoke-EC2SecurityGroupEgress                   RevokeSecurityGroupEgress
 Amazon Elastic Compute Cloud          EC2
Revoke-EC2SecurityGroupIngress                  RevokeSecurityGroupIngress
 Amazon Elastic Compute Cloud          EC2
Update-EC2SecurityGroupRuleEgressDescription    UpdateSecurityGroupRuleDescriptionsEgress
 Amazon Elastic Compute Cloud          EC2
Update-EC2SecurityGroupRuleIngressDescription UpdateSecurityGroupRuleDescriptionsIngress
 Amazon Elastic Compute Cloud          EC2
Edit-EFSMountTargetSecurityGroup                ModifyMountTargetSecurityGroups
 Amazon Elastic File System            EFS
```

```
Get-EFSMountTargetSecurityGroup              DescribeMountTargetSecurityGroups
 Amazon Elastic File System         EFS
Join-ELBSecurityGroupToLoadBalancer          ApplySecurityGroupsToLoadBalancer
 Elastic Load Balancing             ELB
Set-ELB2SecurityGroup                        SetSecurityGroups
 Elastic Load Balancing V2          ELB2
Enable-RDSDBSecurityGroupIngress             AuthorizeDBSecurityGroupIngress
 Amazon Relational Database Service RDS
Get-RDSDBSecurityGroup                       DescribeDBSecurityGroups
 Amazon Relational Database Service RDS
New-RDSDBSecurityGroup                       CreateDBSecurityGroup
 Amazon Relational Database Service RDS
Remove-RDSDBSecurityGroup                    DeleteDBSecurityGroup
 Amazon Relational Database Service RDS
Revoke-RDSDBSecurityGroupIngress             RevokeDBSecurityGroupIngress
 Amazon Relational Database Service RDS
Approve-RSClusterSecurityGroupIngress        AuthorizeClusterSecurityGroupIngress
 Amazon Redshift                    RS
Get-RSClusterSecurityGroup                   DescribeClusterSecurityGroups
 Amazon Redshift                    RS
New-RSClusterSecurityGroup                   CreateClusterSecurityGroup
 Amazon Redshift                    RS
Remove-RSClusterSecurityGroup                DeleteClusterSecurityGroup
 Amazon Redshift                    RS
Revoke-RSClusterSecurityGroupIngress         RevokeClusterSecurityGroupIngress
 Amazon Redshift                    RS
```

如果您知道Amazon服务的名称，但不知道Amazon服务 API 的名称，请同时添加 `-MatchWithRegex` 参数和 `-Service` 参数以将搜索范围限定为单个服务。例如，要仅在 Amazon EC2 服务中获取包含 `SecurityGroup` 的所有 cmdlet 名称，请运行以下命令

```
PS > Get-AWSCmdletName -ApiOperation SecurityGroup -MatchWithRegex -Service EC2

CmdletName                                   ServiceOperation
 ServiceName                  CmdletNounPrefix
----------                                   ----------------
 -----------                  ----------------
Add-EC2SecurityGroupToClientVpnTargetNetwrk  ApplySecurityGroupsToClientVpnTargetNetwork
 Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroup                         DescribeSecurityGroups
 Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroupReference                DescribeSecurityGroupReferences
 Amazon Elastic Compute Cloud EC2
Get-EC2StaleSecurityGroup                    DescribeStaleSecurityGroups
 Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupEgress                 AuthorizeSecurityGroupEgress
 Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupIngress                AuthorizeSecurityGroupIngress
 Amazon Elastic Compute Cloud EC2
New-EC2SecurityGroup                         CreateSecurityGroup
 Amazon Elastic Compute Cloud EC2
Remove-EC2SecurityGroup                      DeleteSecurityGroup
 Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupEgress                RevokeSecurityGroupEgress
 Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupIngress               RevokeSecurityGroupIngress
 Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleEgressDescription UpdateSecurityGroupRuleDescriptionsEgress
 Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleIngressDescription UpdateSecurityGroupRuleDescriptionsIngress
 Amazon Elastic Compute Cloud EC2
```

如果您知道 Amazon Command Line Interface (Amazon CLI) 命令的名称，可以使用 `-AwsCliCommand` 参数和所需的 Amazon CLI 命令名称以获取基于相同 API 的 cmdlet 名称。例如，要在 Amazon EC2 服务中获

取与 `authorize-security-group-ingress` Amazon CLI 命令调用对应的 cmdlet 名称，请运行以下命令：

```
PS > Get-AWSCmdletName -AwsCliCommand "aws ec2 authorize-security-group-ingress"

CmdletName                    ServiceOperation              ServiceName
 CmdletNounPrefix
----------                    ----------------              -----------
 ----------------
Grant-EC2SecurityGroupIngress AuthorizeSecurityGroupIngress Amazon Elastic Compute Cloud
 EC2
```

`Get-AWSCmdletName` cmdlet 仅需要具有 Amazon CLI 命令名称即可识别服务和Amazon API。

要在 Tools for PowerShell Core 中获取所有 cmdlet 的列表，请运行 PowerShell `Get-Command` cmdlet，如以下示例所示。

```
PS > Get-Command -Module AWSPowerShell.NetCore
```

您可以使用 `-Module AWSPowerShell` 运行相同的命令以查看 Amazon Tools for Windows PowerShell 中的 cmdlet。

`Get-Command` cmdlet 按字母顺序生成 cmdlet 列表。请注意，默认情况下，此列表按照 PowerShell 命令动词而不是 PowerShell 名词排序。

要改为按服务对结果排序，请运行以下命令：

```
PS > Get-Command -Module AWSPowerShell.NetCore | Sort-Object Noun,Verb
```

要筛选 `Get-Command` cmdlet 返回的 cmdlet 列表，请将输出输送到 PowerShell `Select-String` cmdlet。例如，要查看适用于Amazon区域的一组 cmdlet，请运行以下命令：

```
PS > Get-Command -Module AWSPowerShell.NetCore | Select-String region

Clear-DefaultAWSRegion
Copy-HSM2BackupToRegion
Get-AWSRegion
Get-DefaultAWSRegion
Get-EC2Region
Get-LSRegionList
Get-RDSSourceRegion
Set-DefaultAWSRegion
```

您还可以通过筛选 cmdlet 名词的服务前缀来查找特定服务的 cmdlet。要查看可用服务前缀的列表，请运行 `Get-AWSPowerShellVersion -ListServiceVersionInfo`。以下示例返回支持 Amazon CloudWatch Events 服务的 cmdlet。

```
PS > Get-Command -Module AWSPowerShell -Noun CWE*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-CWEResourceTag                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWEEventSource                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWERule                                    3.3.563.1
 AWSPowerShell.NetCore
```

```
Cmdlet          Enable-CWEEventSource                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBus                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBusList                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSource                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSourceList                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSource                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceAccountList               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceList                      3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEResourceTag                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERule                                        3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleDetail                                  3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleNamesByTarget                           3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWETargetsByRule                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEEventBus                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEPartnerEventSource                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEEventBus                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPermission                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWETarget                                   3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Test-CWEEventPattern                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEEvent                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPartnerEvent                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPermission                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWERule                                      3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWETarget                                    3.3.563.1
 AWSPowerShell.NetCore
```

cmdlet 命名和别名

Amazon Tools for PowerShell 中适用于每个服务的 cmdlet 基于该服务的 Amazon 开发工具包提供的方法。但是，由于 PowerShell 的必须遵守的命名约定，cmdlet 名称可能与它基于的 API 调用或方法名称不同。例如，`Get-EC2Instance` cmdlet 基于 Amazon EC2 `DescribeInstances` 方法。

在某些情况下，cmdlet 名称可能与方法名称类似，但实际上可能执行不同的功能。例如，Amazon S3 `GetObject` 方法检索 Amazon S3 对象。但是，`Get-S3Object` cmdlet 返回有关 Amazon S3 对象的信息，而非对象本身。

```
PS > Get-S3Object -BucketName text-content -Key aws-tech-docs

ETag         : "df000002a0fe0000f3c000004EXAMPLE"
BucketName   : aws-tech-docs
Key          : javascript/frameset.js
LastModified : 6/13/2011 1:24:18 PM
Owner        : Amazon.S3.Model.Owner
Size         : 512
StorageClass : STANDARD
```

要使用Amazon Tools for PowerShell获取 S3 对象，请运行 `Read-S3Object` cmdlet：

```
PS > Read-S3Object -BucketName text-content -Key text-object.txt -file c:\tmp\text-object-
download.text

Mode          LastWriteTime            Length Name
----          -------------            ------ ----
-a---         11/5/2012   7:29 PM      20622  text-object-download.text
```

> **Note**
>
> Amazon cmdlet 的 cmdlet 帮助提供了该 cmdlet 基于的Amazon开发工具包 API 名称。有关标准 PowerShell 动词及其含义的更多信息，请参阅批准的 PowerShell 命令动词。

使用 `Remove` 动词的所有Amazon cmdlet（以及在添加 `-Terminate` 参数时使用的 `Stop-EC2Instance` cmdlet）都会提示您在继续前进行确认。要绕过确认，请在命令中添加 `-Force` 参数。

> **Important**
>
> Amazon cmdlet 不支持 `-WhatIf` 开关。

Aliases

Amazon Tools for PowerShell 安装程序会安装一个别名文件，其中包含很多Amazon cmdlet 的别名。您可能会发现这些别名比 cmdlet 名称更直观。例如，在某些别名中，服务名称和Amazon开发工具包方法名称替换 PowerShell 动词和名词。例如 `EC2-DescribeInstances` 别名。

其他别名使用动词，尽管不遵循标准 PowerShell 约定，但它们可以更好地描述实际操作。例如，别名文件将别名 `Get-S3Content` 映射到 cmdlet `Read-S3Object`。

```
PS > Set-Alias -Name Get-S3Content -Value Read-S3Object
```

别名文件位于Amazon Tools for PowerShell安装目录中。要将别名加载到您的环境中，请对该文件使用 dot-source。下面是一个基于 Windows 的示例。

```
PS > . "C:\Program Files (x86)\AWS Tools\PowerShell\AWSPowershell\AWSAliases.ps1"
```

对于 Linux 或 macOS shell，它可能看起来像这样：

```
. ~/.local/share/powershell/Modules/AWSPowerShell.NetCore/3.3.563.1/AWSAliases.ps1
```

要显示所有Amazon Tools for PowerShell别名，请运行以下命令。该命令在 PowerShell `?` cmdlet 和 `Where-Object` 属性中使用 `Source` 别名以筛选仅来自 `AWSPowerShell.NetCore` 模块的别名。

```
PS > Get-Alias | ? Source -like "AWSPowerShell.NetCore"

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           Add-ASInstances                                    3.3.343.0  AWSPowerShell
Alias           Add-CTTag                                          3.3.343.0  AWSPowerShell
Alias           Add-DPTags                                         3.3.343.0  AWSPowerShell
Alias           Add-DSIpRoutes                                     3.3.343.0  AWSPowerShell
Alias           Add-ELBTags                                        3.3.343.0  AWSPowerShell
Alias           Add-EMRTag                                         3.3.343.0  AWSPowerShell
Alias           Add-ESTag                                          3.3.343.0  AWSPowerShell
Alias           Add-MLTag                                          3.3.343.0  AWSPowerShell
Alias           Clear-AWSCredentials                               3.3.343.0  AWSPowerShell
Alias           Clear-AWSDefaults                                  3.3.343.0  AWSPowerShell
Alias           Dismount-ASInstances                               3.3.343.0  AWSPowerShell
Alias           Edit-EC2Hosts                                      3.3.343.0  AWSPowerShell
Alias           Edit-RSClusterIamRoles                             3.3.343.0  AWSPowerShell
Alias           Enable-ORGAllFeatures                              3.3.343.0  AWSPowerShell
Alias           Find-CTEvents                                      3.3.343.0  AWSPowerShell
Alias           Get-ASACases                                       3.3.343.0  AWSPowerShell
Alias           Get-ASAccountLimits                                3.3.343.0  AWSPowerShell
Alias           Get-ASACommunications                              3.3.343.0  AWSPowerShell
Alias           Get-ASAServices                                    3.3.343.0  AWSPowerShell
Alias           Get-ASASeverityLevels                              3.3.343.0  AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckRefreshStatuses          3.3.343.0  AWSPowerShell
Alias           Get-ASATrustedAdvisorChecks                        3.3.343.0  AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckSummaries                3.3.343.0  AWSPowerShell
Alias           Get-ASLifecycleHooks                               3.3.343.0  AWSPowerShell
Alias           Get-ASLifecycleHookTypes                           3.3.343.0  AWSPowerShell
Alias           Get-AWSCredentials                                 3.3.343.0  AWSPowerShell
Alias           Get-CDApplications                                 3.3.343.0  AWSPowerShell
Alias           Get-CDDeployments                                  3.3.343.0  AWSPowerShell
Alias           Get-CFCloudFrontOriginAccessIdentities             3.3.343.0  AWSPowerShell
Alias           Get-CFDistributions                                3.3.343.0  AWSPowerShell
Alias           Get-CFGConfigRules                                 3.3.343.0  AWSPowerShell
Alias           Get-CFGConfigurationRecorders                      3.3.343.0  AWSPowerShell
Alias           Get-CFGDeliveryChannels                            3.3.343.0  AWSPowerShell
Alias           Get-CFInvalidations                                3.3.343.0  AWSPowerShell
Alias           Get-CFNAccountLimits                               3.3.343.0  AWSPowerShell
Alias           Get-CFNStackEvents                                 3.3.343.0  AWSPowerShell

...
```

要在该文件中添加您自己的别名，您可能需要将 PowerShell 的 `$MaximumAliasCount` 首选项变量值提高到大于 5500 的值。默认值为 4096；您可以将其提高到最多 32768。为此，请运行以下命令。

```
PS > $MaximumAliasCount = 32768
```

要验证您的更改是否成功，请输入变量名称以显示其当前值。

```
PS > $MaximumAliasCount
32768
```

管道传输和 $AWSHistory

对于返回集合的 Amazon 服务调用，集合中的对象列举到管道。对于生成的对象，当其中包含集合之外附加字段以及不分页控制字段时，将这些字段作为调用的 Note 属性添加。这些 Note 属性记录在新的 `$AWSHistory` 会话变量中，您需要访问此数据。`$AWSHistory` 变量在下一节中介绍。

Note

在 Tools for Windows PowerShell 1.1 之前的版本中，集合对象本身将进行发送，这需要使用 foreach {$_.getenumerator()} 来继续管道传输。

示例

以下示例返回Amazon区域列表和每个区域中的 Amazon EC2 系统镜像 (AMI)。

```
PS > Get-AWSRegion | % { Echo $_.Name; Get-EC2Image -Owner self -Region $_ }
```

以下示例停止当前默认区域中的所有 Amazon EC2 实例。

```
PS > Get-EC2Instance | Stop-EC2Instance
```

由于集合列举到管道，来自指定 cmdlet 的输出可能是 `$null`、单个对象或集合。如果是集合，则可以使用 `.Count` 属性来确定集合的大小。但是，在只发出单个对象时，`.Count` 属性不存在。如果您的脚本需要以一致的方式来确定发出了多少个对象，则可以检查 `EmittedObjectsCount` 中最后一个命令值的 `$AWSHistory` 属性。

$AWSHistory

为了更好地支持管道传输，来自 Amazon cmdlet 的输出不会进行改造以包括服务响应和结果实例，与已发出集合对象上的 Note 属性不同。相反，对于发出单个集合作为输出的调用，现在将调用列举到 PowerShell 管道。这意味着Amazon开发工具包响应和结果数据不能存在于管道中，因为其中不包含可以将其附加到的集合对象。

虽然大部分用户可能不需要此数据，不过它对于诊断用途可能非常有用，因为您可以准确地查看 cmdlet 在基础Amazon服务调用中发送和接收的内容。

从版本 1.1 开始，现在此数据以及更多数据在名为 `$AWSHistory` 的新 shell 变量中提供。此变量维护 Amazon cmdlet 调用的记录以及为每次调用接收的服务响应。（可选）可以配置此历史记录以便同时记录每次 cmdlet 发出的服务请求。其他有用数据，例如 cmdlet 的整体执行时间也可以从各个条目中获取。

`$AWSHistory.Commands` 列表中的每个条目的类型为 `AWSCmdletHistory`。此类型具有以下有用成员：

CmdletName

cmdlet 的名称。

CmdletStart

运行 cmdlet 的日期时间。

CmdletEnd

cmdlet 完成所有处理的日期时间。

请求

如果启用了请求记录，则为上个服务请求的列表。

响应

收到的上个服务响应的列表。

LastServiceResponse

返回最近服务响应的帮助程序。

LastServiceRequest

返回最近服务请求（如果可用）的帮助程序。

请注意，在使用发出服务调用的 Amazon cmdlet 之前，不创建 `$AWSHistory` 变量。在该时间之前，它的计算结果为 $null。

Note

Tools for Windows PowerShell 的早期版本发出与服务响应相关的数据，以作为返回的对象的 `Note` 属性。这些内容现在可在为列表中每次调用记录的响应条目中找到。

Set-AWSHistoryConfiguration

一个 cmdlet 调用可以容纳零个或多个服务请求和响应条目。为了减少对内存的影响，默认情况下 `$AWSHistory` 列表仅保留最后五次 cmdlet 执行的记录，对于每次执行，保留最后五个服务响应（如果启用，包括最后五个服务请求）。您可以通过运行 `Set-AWSHistoryConfiguration` cmdlet 来更改这些默认限制。它允许您控制列表的大小以及是否同时记录服务请求：

```
PS > Set-AWSHistoryConfiguration -MaxCmdletHistory <value> -MaxServiceCallHistory <value> -
RecordServiceRequests
```

`-MaxCmdletHistory` 参数设置可以随时跟踪的 cmdlet 最大数量。值为 0 将禁止记录 Amazon cmdlet 活动。`-MaxServiceCallHistory` 参数设置为每个 cmdlet 跟踪的服务响应（和/或请求）的最大数量。如果指定 `-RecordServiceRequests` 参数，则为每个 cmdlet 启用服务请求跟踪。所有参数都是可选的。

如果未带参数运行，则单纯运行 `Set-AWSHistoryConfiguration` 将停用所有之前的请求记录，并保持当前列表大小不变。

要清除当前历史记录列表中的所有条目，请运行 `Clear-AWSHistory` cmdlet。

$AWSHistory 示例

将保存在列表中的 Amazon cmdlet 的详细信息列举到管道。

```
PS > $AWSHistory.Commands
```

访问上次运行的 Amazon cmdlet 的详细信息：

```
PS > $AWSHistory.LastCommand
```

访问上次运行的 Amazon cmdlet 最后接收的服务响应的详细信息。如果 Amazon cmdlet 分页输出，则可以进行多次服务调用来获取所有数据或最大数量的数据（由 cmdlet 上的参数确定）。

```
PS > $AWSHistory.LastServiceResponse
```

访问最近发出的请求的详细信息 (再次强调，如果代表用户进行分页，则 cmdlet 可能发出多个请求)。除非启用了服务请求跟踪，否则将生成 $null。

```
PS > $AWSHistory.LastServiceRequest
```

对于返回多页的操作，自动从分页到完成

如果服务 API 对指定调用施加默认最大对象返回计数或支持可分页结果集，则所有 cmdlet 默认情况下“分页到完成”。根据需要，每个 cmdlet 代表您尽可能多地发出调用，以将完整的数据集返回到管道。

以下示例使用 `Get-S3Object`，`$c` 变量包含 `S3Object` 存储桶中的每个 键的 `test` 实例，这可能会生成非常大的数据集。

```
PS > $c = Get-S3Object -BucketName test
```

如果您希望保持对返回数据量的控制，可以在单独的 cmdlet 上使用参数（例如，MaxKey 上的 Get-S3Object），或者自己明确地处理分页，将 cmdlet 上的分页参数与放在 $AWSHistory 变量中的数据结合使用来获取服务的下一个令牌数据。以下示例使用 MaxKeys 参数将返回的 S3Object 实例数量限制为不超过在存储桶中找到的前 500 个。

```
PS > $c = Get-S3Object -BucketName test -MaxKey 500
```

要了解是否有更多数据可用但未返回，请使用记录 cmdlet 所发出的服务调用的 $AWSHistory 会话变量条目。

如果以下表达式计算结果为 $true，则可以使用 next 查找下一组结果的 $AWSHistory.LastServiceResponse.NextMarker 标记：

```
$AWSHistory.LastServiceResponse -ne $null && $AWSHistory.LastServiceResponse.IsTruncated
```

要使用 Get-S3Object 手动控制分页，请将 cmdlet 的 MaxKey 和 Marker 参数与最后记录的响应的 IsTruncated/NextMarker 备注结合使用。在以下示例中，变量 $c 包含在存储桶中的指定键前缀标记之后，找到的接下来 500 个对象的最多 500 个 S3Object 实例。

```
PS > $c = Get-S3Object -BucketName test -MaxKey 500 -Marker
 $AWSHistory.LastServiceResponse.NextMarker
```

使用Amazon Tools for PowerShell配置联合身份

要让组织中的用户访问 Amazon 资源，您必须配置一种可重复的标准身份验证方法来提高安全性、可审核性、合规性，并能够支持角色和账户分离。尽管常常允许用户访问 Amazon API，但是如果没有联合 API 访问，您还必须创建 Amazon Identity and Access Management (IAM) 用户，这样会无法达到使用联合验证的目的。本主题介绍Amazon Tools for PowerShell中的 SAML（安全断言标记语言）支持，这可帮助您简化联合访问解决方案。

Amazon Tools for PowerShell中的 SAML 支持可让您为用户提供对 Amazon 服务的联合访问权限。SAML 是一种基于 XML 的开放标准格式，用于在服务之间传输用户身份验证和授权数据，特别是在身份提供商（如 Active Directory 联合服务）和服务提供商（如Amazon）之间传输数据。有关 SAML 及其工作原理的更多信息，请参阅 Wikipedia 上的 SAML 或结构信息标准化促进组织 (OASIS) 网站上的 SAML 技术规范。Amazon Tools for PowerShell中的 SAML 支持与 SAML 2.0 兼容。

Prerequisites

首次尝试使用 SAML 支持前，您必须做好以下准备。

- 与您的Amazon账户正确集成的联合身份解决方案，用于仅使用组织凭证进行控制台访问。有关如何专门为 Active Directory 联合服务执行该操作的更多信息，请参阅 IAM 用户指南中的关于 SAML 2.0 联合身份验证，以及博文使用 Windows Active Directory、AD FS 和 SAML 2.0 启用Amazon联合身份验证。尽管该博文涵盖 AD FS 2.0，但如果您运行的是 AD FS 3.0，其步骤与该文章中的步骤也是类似的。
- 本地工作站上安装的版本 3.1.31.0 或更高版本的Amazon Tools for PowerShell。

联合身份用户如何获取对Amazon服务 API 的联合访问权限

以下过程从较高层面介绍 Active Directory (AD) 用户如何由 AD FS 联合起来获取Amazon资源的访问权限。

1. 联合用户计算机上的客户端将针对 AD FS 进行身份验证。
2. 如果身份验证成功，AD FS 会向用户发送 SAML 断言。
3. 用户的客户端将 SAML 断言作为 SAML 联合请求的一部分发送到 Amazon Security Token Service (STS)。
4. STS 返回 SAML 响应，其中包含用户可代入的角色的Amazon临时凭证。
5. 用户通过在 Amazon Tools for PowerShell 发出的请求中包括这些临时凭证来访问Amazon服务 API。

SAML 支持在Amazon Tools for PowerShell中的工作原理

此部分介绍Amazon Tools for PowerShell cmdlet 如何支持为用户配置基于 SAML 的联合身份。

1. 当用户尝试运行需要凭证才能调用Amazon的 cmdlet 时，Amazon Tools for PowerShell 使用 Windows 用户的当前凭证或者以交互方式针对 AD FS 进行身份验证。
2. AD FS 会对该用户进行身份验证。
3. AD FS 生成包含断言的 SAML 2.0 身份验证响应；断言目的是为了标识和提供用户相关信息。Amazon Tools for PowerShell从 SAML 断言中提取用户授权角色的列表。
4. Amazon Tools for PowerShell 通过执行 `AssumeRoleWithSAMLRequest` API 调用将 SAML 请求（包括请求的角色 Amazon 资源名称 (ARN)）转发到 STS。
5. 如果 SAML 请求有效，STS 会返回包含 Amazon `AccessKeyId`、`SecretAccessKey` 和 `SessionToken` 的响应。这些凭证的有效期为 3,600 秒（1 小时）。
6. 用户现在具有有效的凭证，可使用该用户的角色有权访问的任何Amazon服务 API。Amazon Tools for PowerShell 会在任何后续Amazon API 调用中自动应用这些凭证，并在它们过期时自动续订。

   Note

   如果凭证过期且需要新凭证，Amazon Tools for PowerShell将自动向 AD FS 重新进行身份验证，并在随后一小时内获得新凭证。对于加入域的账户用户，将以无提示方式执行此过程。对于未加入域的账户，Amazon Tools for PowerShell将在重新进行身份验证之前提示用户输入其凭证。

如何使用 PowerShell SAML 配置 Cmdlet

Amazon Tools for PowerShell包括两个提供 SAML 支持的新 cmdlet。

- `Set-AWSSamlEndpoint` 配置 AD FS 终端节点，为终端节点分配易记名称，并选择性地描述终端节点的身份验证类型。
- `Set-AWSSamlRoleProfile` 创建或编辑要与 AD FS 终端节点关联的用户账户配置文件，该终端节点通过指定您向 `Set-AWSSamlEndpoint` cmdlet 提供的易记名称加以标识。每个角色配置文件都映射到用户有权执行的一个角色。

  与Amazon凭证配置文件相同，您还将为角色配置文件分配一个易记名称。您可以使用与 `Set-AWSCredential` cmdlet 相同的易记名称，或与调用Amazon服务 API 的任何 cmdlet 的 `-ProfileName` 参数值相同的易记名称。

打开新的Amazon Tools for PowerShell会话。如果您运行的是 PowerShell 3.0 或更新版本，Amazon Tools for PowerShell模块将在您运行其任何 cmdlet 时自动导入。如果您正在运行 PowerShell 2.0，则必须通过运行 “导入模块”cmdlet 手动导入模块，如以下示例所示。

```
PS > Import-Module "C:\Program Files (x86)\AWS Tools\PowerShell\AWSPowerShell
\AWSPowerShell.psd1"
```

如何运行 Set-AWSSamlEndpoint 和 Set-AWSSamlRoleProfile Cmdlet

1. 首先，为 AD FS 系统配置终端节点设置。该操作最简单的方法是将终端节点存储在变量中，如本步骤所示。请确保将占位符账户 ID 和 AD FS 主机名替换为您自己的账户 ID 和 AD FS 主机名。在 Endpoint 参数中指定 AD FS 主机名。

```
PS > $endpoint = "https://adfs.example.com/adfs/ls/IdpInitiatedSignOn.aspx?
loginToRp=urn:amazon:webservices"
```

2. 要创建终端节点设置，请运行 Set-AWSSamlEndpoint cmdlet，为 AuthenticationType 参数指定正确的值。有效值包括 Basic、Digest、Kerberos、Negotiate 和 NTLM。如果未指定该参数，则默认值为 Kerberos。

```
PS > $epName = Set-AWSSamlEndpoint -Endpoint $endpoint -StoreAs ADFS-Demo -
AuthenticationType NTLM
```

   cmdlet 返回您使用 -StoreAs 参数分配的易记名称，因此，在下一行中运行 Set-AWSSamlRoleProfile 时，您可以使用该名称。

3. 现在，运行 Set-AWSSamlRoleProfile cmdlet 以便使用 AD FS 身份提供商进行身份验证，并获取用户有权执行的角色集（在 SAML 断言中）。

   Set-AWSSamlRoleProfile cmdlet 使用返回的角色集提示用户选择要与所指定配置文件关联的角色，或者验证参数中提供的角色数据是否存在（如果不存在，则提示用户进行选择）。如果仅向用户授予了一个角色，cmdlet 会自动将该角色与配置文件关联，而不会提示用户。无需提供凭证即可设置配置文件，以便在加入域时使用。

```
PS > Set-AWSSamlRoleProfile -StoreAs SAMLDemoProfile -EndpointName $epName
```

   或者，对于未加入域的账户，可以提供 Active Directory 凭证，然后选择用户可以访问的Amazon角色，如下一行所示。如果您具有不同 Active Directory 用户账户来区分组织中的角色（例如，管理功能），此功能会很有用。

```
PS > $credential = Get-Credential -Message "Enter the domain credentials for the
 endpoint"
PS > Set-AWSSamlRoleProfile -EndpointName $epName -NetworkCredential $credential -
StoreAs SAMLDemoProfile
```

4. 在任一情况下，Set-AWSSamlRoleProfile cmdlet 都会提示您选择应存储在配置文件中的角色。以下示例显示了两个可用角色：ADFS-Dev 和 ADFS-Production。IAM 角色与 AD FS 管理员的 AD 登录凭证相关联。

```
Select Role
Select the role to be assumed when this profile is active
[1] 1 - ADFS-Dev  [2] 2 - ADFS-Production  [?] Help (default is "1"):
```

   或者，您可以通过输入 RoleARN、PrincipalARN 和可选 NetworkCredential 参数来指定没有提示的角色。如果身份验证返回的断言中未列出指定的角色，则提示用户从可用角色中进行选择。

```
PS > $params = @{ "NetworkCredential"=$credential,
 "PrincipalARN"="{arn:aws:iam::012345678912:saml-provider/ADFS}",
 "RoleARN"="{arn:aws:iam::012345678912:role/ADFS-Dev}"
}
PS > $epName | Set-AWSSamlRoleProfile @params -StoreAs SAMLDemoProfile1 -Verbose
```

5. 您可以通过添加 StoreAllRoles 参数在一个命令中为所有角色创建配置文件，如以下代码所示。请注意，角色名称用作配置文件名称。

```
PS > Set-AWSSamlRoleProfile -EndpointName $epName -StoreAllRoles
ADFS-Dev
ADFS-Production
```

如何使用角色配置文件运行需要Amazon凭证的 Cmdlet

要运行需要 Amazon 凭证的 cmdlet，可以使用在 Amazon 共享凭证文件中定义的角色配置文件。为 Set-AWSCredential 提供角色配置文件的名称（或作为 Amazon Tools for PowerShell 中的任何 ProfileName 参数的值），以自动为配置文件中描述的角色获取临时Amazon凭证。

尽管一次只能使用一个角色配置文件，但是在一个 shell 会话中，您可以在多个配置文件之间切换。Set-AWSCredential cmdlet 本身不会在运行时执行身份验证并获取凭证；该 cmdlet 记录您想要使用指定的角色配置文件。在您运行需要 Amazon 凭证的 cmdlet 之前，不会执行身份验证或者请求提供凭证。

现在，您可以使用通过 SAMLDemoProfile 配置文件获取的临时Amazon凭证来使用Amazon服务 API。以下各部分介绍如何使用角色配置文件的示例。

示例 1：使用 Set-AWSCredential 设置默认角色

该示例使用 Amazon Tools for PowerShell 为Set-AWSCredential会话设置默认角色。然后，您可以运行需要凭证且由指定角色授权的 cmdlet。此示例列出美国西部（俄勒冈）区域中与您使用 Set-AWSCredential cmdlet 指定的配置文件关联的所有 Amazon Elastic Compute Cloud 实例。

```
PS > Set-AWSCredential -ProfileName SAMLDemoProfile
PS > Get-EC2Instance -Region us-west-2 | Format-Table -Property Instances,GroupNames

Instances                                                        GroupNames
---------                                                        ----------
{TestInstance1}                                                  {default}
{TestInstance2}                                                  {}
{TestInstance3}                                                  {launch-wizard-6}
{TestInstance4}                                                  {default}
{TestInstance5}                                                  {}
{TestInstance6}                                                  {AWS-OpsWorks-Default-Server}
```

示例 2：在 PowerShell 会话期间更改角色配置文件

该示例列出具有与 SAMLDemoProfile 配置文件关联的角色的Amazon账户中的所有可用的 Amazon S3 存储桶。该示例说明，尽管您以前可能在Amazon Tools for PowerShell会话中使用其他配置文件，您仍然可以在支持它的 cmdlet 中为 -ProfileName 参数指定不同的值以更改配置文件。对于从 PowerShell 命令行管理 Amazon S3 的管理员而言，这是一项常见任务。

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

CreationDate                                                     BucketName
------------                                                     ----------
7/25/2013 3:16:56 AM                                             mybucket1
4/15/2015 12:46:50 AM                                            mybucket2
4/15/2015 6:15:53 AM                                             mybucket3
1/12/2015 11:20:16 PM                                            mybucket4
```

请注意，Get-S3Bucket cmdlet 指定通过运行 Set-AWSSamlRoleProfile cmdlet 创建的配置文件的名称。如果以前您已在会话中设置角色配置文件（例如，通过运行 Set-AWSCredential cmdlet），并且

您想要在 `Get-S3Bucket` cmdlet 中使用不同的角色配置文件，该命令可能会很有用。配置文件管理器向 `Get-S3Bucket` cmdlet 提供临时凭证。

虽然凭证会在 1 小时后过期（STS 强制实施的限制），但是Amazon Tools for PowerShell在检测到当前凭证已过期时，会通过请求新的 SAML 断言来自动更新凭证。

对于加入域的用户，由于在身份验证期间使用了当前用户的 Windows 身份，因此执行该过程时不会中断。对于未加入域的用户账户，Amazon Tools for PowerShell会显示请求提供用户密码的 PowerShell 凭证提示。用户应提供凭证，用于重新验证用户以及获取新断言。

示例 3：获取区域中的实例

以下示例列出了亚太地区（悉尼）区域中与账户所用的 `ADFS-Production` 配置文件关联的所有 Amazon EC2 实例。该命令对返回某个区域中的所有Amazon EC2 实例非常有用。

```
PS > (Get-Ec2Instance -ProfileName ADFS-Production -Region ap-southeast-2).Instances |
 Select InstanceType, @{Name="Servername";Expression={$_.tags | where key -eq "Name" |
 Select Value -Expand Value}}

 InstanceType                                                    Servername
 ------------                                                    ----------
 t2.small                                                        DC2
 t1.micro                                                        NAT1
 t1.micro                                                        RDGW1
 t1.micro                                                        RDGW2
 t1.micro                                                        NAT2
 t2.small                                                        DC1
 t2.micro                                                        BUILD
```

补充阅读

有关如何实施联合 API 访问的常规信息，请参阅如何使用 SAML 2.0 实施联合 API/CLI 访问常规解决方案。

如果您有任何支持问题或意见，请访问Amazon开发人员论坛中的编写 PowerShell 脚本或 .NET 开发。

使用 Amazon Tools for PowerShell

主题

本节提供使用 Amazon Tools for PowerShell 访问Amazon服务的示例。这些示例帮助说明如何使用 cmdlet 执行实际的Amazon任务。

PowerShell 文件联接编码

Amazon Tools for PowerShell 中的一些 cmdlet 编辑您目前在 Amazon 中拥有的文件或记录。例如 `Edit-R53ResourceRecordSet`，它对 Amazon Route 53 调用 ChangeResourceRecordSets API。

在 PowerShell 5.1 或更早版本中编辑或联接文件时，PowerShell 会以 UTF-16（而不是 UTF-8）格式对输出进行编码。这可能会添加不需要的字符并创建无效的结果。十六进制编辑器可以显示不需要的字符。

要避免将文件输出转换为 UTF-16，您可以将命令传递到 PowerShell 的 `Out-File` cmdlet 中并指定 UTF-8 编码，如以下示例所示：

```
PS > *some file concatenation command* | Out-File filename.txt -Encoding utf8
```

如果您在 PowerShell 控制台中运行 Amazon CLI 命令，则相同的行为将适用。您可以将 Amazon CLI 命令的输出传递到 PowerShell 控制台中的 `Out-File`。其他 cmdlet（例如 `Export-Csv` 或 `Export-Clixml`）也具有 `Encoding` 参数。有关具有 `Encoding` 参数并允许您纠正联接文件输出的编码的 cmdlet 的完整列表，请运行以下命令：

```
PS > Get-Command -ParameterName "Encoding"
```

Note

PowerShell 6.0 及更高版本（包括 PowerShell Core）会自动为联接的文件输出保留 UTF-8 编码。

PowerShell 工具的返回对象

为了使 Amazon Tools for PowerShell 在本机 PowerShell 环境中发挥更大作用，Amazon Tools for PowerShell cmdlet 返回的对象是一个 .NET 对象，而不是通常从 Amazon 开发工具包中的相应 API 返回的 JSON 文本对象。例如，`Get-S3Bucket` 发出 `Buckets` 集合，而不是 Amazon S3 JSON 响应对象。`Buckets` 集合可以放置在 PowerShell 管道中，并以适当的方式交互。同样，`Get-EC2Instance` 发出一个 `Reservation` .NET 对象集合，而不是 `DescribeEC2Instances` JSON 结果对象。此行为是设计使然，以使 Amazon Tools for PowerShell 体验与惯用的 PowerShell 更加一致。

如果您需要，您可以使用实际的服务响应。它们作为 `note` 属性存储在返回的对象上。对于使用 `NextToken` 字段支持分页的 API 操作，这些还可作为 `note` 属性附加。

Amazon EC2 (p. 54)

本节演示了启动 Amazon EC2 实例所需的步骤，包括如何：

- 检索 Amazon Machine Image (AMI) 列表。
- 为 SSH 身份验证创建密钥对。
- 创建和配置 Amazon EC2 安全组。
- 启动实例并检索关于它的信息。

Amazon S3 (p. 47)

本节演示创建托管在 Amazon S3 中的静态网站所需的步骤。它将介绍如何：

- 创建和删除 Amazon S3 存储桶。
- 将文件作为对象上传到 Amazon S3 存储桶中。
- 从 Amazon S3 存储桶中删除对象。
- 指定 Amazon S3 存储桶作为网站。

IAM 与 Amazon Tools for PowerShell (p. 51)

本节说明 Amazon Identity and Access Management (IAM) 中的基本操作，包括如何：

- 创建 IAM 组。
- 创建 IAM 用户。
- 将 IAM 用户添加到 IAM 组。
- 为 IAM 用户指定策略。
- 为 IAM 用户设置密码和凭证。

Amazon Lambda 和 Amazon Tools for PowerShell (p. 63)

本节简要说明了 Amazon Lambda Tools for PowerShell 模块以及设置该模块所需的步骤。

Amazon SNS 和 Amazon SQS (p. 64)

本节介绍为 Amazon SQS 队列订阅 Amazon SNS 主题所需的步骤。它将介绍如何：

- 创建 Amazon SNS 主题。
- 创建 Amazon SQS 队列。
- 为队列订阅主题。
- 发送消息到主题。
- 从队列接收消息。

CloudWatch (p. 67)

本节提供如何将自定义数据发布到 CloudWatch 的示例。

- 将自定义指标发布到您的 CloudWatch 控制面板。

另请参阅

- Amazon Tools for Windows PowerShell入门 (p. 22)

Amazon S3 与 Tools for Windows PowerShell

主题

在本节中，我们通过使用 Amazon S3 和 CloudFront 的 Amazon Tools for Windows PowerShell 来创建一个静态网站。在此过程中，我们通过这些服务演示了大量常见任务。本演练仿效托管静态网站的入门指南，其中说明了使用 Amazon管理控制台的类似过程。

此处显示的命令假设您已为 PowerShell 会话设置了默认凭证和默认区域。因此，凭证和区域未包含在 cmdlet 的调用中。

> Note
>
> 目前，没有用于重命名存储桶或对象的 Amazon S3 API，因此，没有用于执行此任务的单个 Tools for Windows PowerShell cmdlet。要在 S3 中重命名对象，我们建议您运行 Copy-S3Object cmdlet 以将此对象复制到使用新名称的对象，然后运行 Remove-S3Object cmdlet 以删除原始对象。

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 在 Amazon S3 上托管静态网站
- Amazon S3 控制台

创建 Amazon S3 存储桶，验证它的区域，然后删除它（可选）

使用 `New-S3Bucket` cmdlet 创建新的 Amazon S3 存储桶。以下示例创建一个名为 `website-example` 的存储桶。该存储桶的名称在所有区域中必须是唯一的。该示例在 `us-west-1` 区域中创建存储桶。

```
PS > New-S3Bucket -BucketName website-example -Region us-west-2

CreationDate          BucketName
------------          ----------
8/16/19 8:45:38 PM    website-example
```

您可以使用 `Get-S3BucketLocation` cmdlet 验证存储桶所在的区域。

```
PS > Get-S3BucketLocation -BucketName website-example

Value
-----
us-west-2
```

完成本教程后，您可以使用以下行删除此存储桶。我们建议您保留此存储桶，因为我们在后续示例中会使用它。

```
PS > Remove-S3Bucket -BucketName website-example
```

请注意，存储桶删除过程需要花一些时间才能完成。如果您尝试立即重新创建同名存储桶，则 `New-S3Bucket` cmdlet 可能会失败，直到旧存储桶完全消失。

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 放置存储桶（Amazon S3 服务参考）
- Amazon 适用于 Amazon S3 的 PowerShell 区域

将 Amazon S3 存储桶配置为网站并启用日志记录

使用 `Write-S3BucketWebsite` cmdlet 将 Amazon S3 存储桶配置为静态网站。以下示例为默认内容网页指定名称 `index.html`，并为默认错误网页指定名称 `error.html`。请注意，该 cmdlet 不创建这些页面。需要将它们上传为 Amazon S3 对象 (p. 49)。

```
PS > Write-S3BucketWebsite -BucketName website-example -
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument
 error.html
RequestId      : A1813E27995FFDDD
AmazonId2      : T7hlDOeLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xAlI
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}
Metadata       : {}
ResponseXml    :
```

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)

- 放置存储桶网站（Amazon S3 API 参考）
- 放置存储桶 ACL（Amazon S3 API 参考）

将对象上传到 Amazon S3 存储桶

使用 `Write-S3Object` cmdlet 将文件作为对象从本地文件系统上传到 Amazon S3 存储桶中。以下示例创建两个简单的 HTML 文件并将其上传到 Amazon S3 存储桶，然后验证上传的对象是否存在。`-File` 参数和 `Write-S3Object` 指定本地文件系统中文件的名称。`-Key` 参数指定 Amazon S3 中对应的对象将具有的名称。

Amazon 从文件扩展名来推断对象的内容类型，在本例中为“.html”。

```
PS > # Create the two files using here-strings and the Set-Content cmdlet
PS > $index_html = @"
>> <html>
>>   <body>
>>     <p>
>>       Hello, World!
>>     </p>
>>   </body>
>> </html>
>> "@
>>
PS > $index_html | Set-Content index.html
PS > $error_html = @"
>> <html>
>>   <body>
>>     <p>
>>       This is an error page.
>>     </p>
>>   </body>
>> </html>
>> "@
>>
>>$error_html | Set-Content error.html
>># Upload the files to Amazon S3 using a foreach loop
>>foreach ($f in "index.html", "error.html") {
>> Write-S3Object -BucketName website-example -File $f -Key $f -CannedACLName public-read
>> }
>>
PS > # Verify that the files were uploaded
PS > Get-S3BucketWebsite -BucketName website-example

IndexDocumentSuffix                                          ErrorDocument
-------------------                                          -------------
index.html                                                   error.html
```

标准 ACL 选项

利用 Tools for Windows PowerShell 指定标准 ACL 时所用的值，与 Amazon SDK for .NET 所用的值相同。但要注意，这些值不同于 Amazon S3 `Put Object` 操作使用的值。Tools for Windows PowerShell 支持以下标准 ACL：

- NoACL
- 私有
- public-read
- public-read-write
- aws-exec-read
- authenticated-read

- bucket-owner-read
- bucket-owner-full-control
- log-delivery-write

有关这些标准 ACL 设置的更多信息，请参阅访问控制列表概述。

关于分段上传的备注

如果您使用 Amazon S3 API 上传大于 5 GB 大小的文件，则需要使用分段上传。但是，Tools for Windows PowerShell 提供的 `Write-S3Object` cmdlet 可以透明地处理大于 5 GB 的上传文件。

测试网站

此时，您可以通过使用浏览器导航到网站来测试网站。托管在 Amazon S3 中的静态网站的 URL 遵循标准格式。

```
http://<bucket-name>.s3-website-<region>.amazonaws.com
```

例如：

```
http://website-example.s3-website-us-west-1.amazonaws.com
```

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 放置对象（Amazon S3 API 参考）
- 标准 ACL（Amazon S3 API 参考）

删除 Amazon S3 对象和存储桶

此部分说明如何删除在前面的部分中创建的网站。可以删除 HTML 文件的对象，然后删除站点的 Amazon S3 存储桶。

首先，运行 `Remove-S3Object` cmdlet 以从 Amazon S3 存储桶中删除 HTML 文件的对象。

```
PS > foreach ( $obj in "index.html", "error.html" ) {
>> Remove-S3Object -BucketName website-example -Key $obj
>> }
>>
IsDeleteMarker
--------------
False
```

`False` 响应是 Amazon S3 请求处理方式的预期构件。在此上下文中，它不表示出现问题。

现在，您可以运行 `Remove-S3Bucket` cmdlet 以删除站点的现在为空的 Amazon S3 存储桶。

```
PS > Remove-S3Bucket -BucketName website-example

RequestId      : E480ED92A2EC703D
AmazonId2      : k6tqaqC1nMkoeYwbuJXUx1/UDa49BJd6dfLN0Ls1mWYNPHjbc8/Nyvm6AGbWcc2P
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Date, Server}
Metadata       : {}
```

```
ResponseXml    :
```

在 1.1 和更高版本的Amazon Tools for PowerShell中，您可以在 `-DeleteBucketContent` 中添加 `Remove-S3Bucket` 参数，这会先删除指定存储桶中的所有对象和对象版本，然后再尝试删除存储桶本身。根据存储桶中的对象或对象版本的数目，该操作可能需要花费较长时间。在 Tools for Windows PowerShell 1.1 之前的版本中，存储桶必须为空，`Remove-S3Bucket` 才能将其删除。

> **Note**
>
> 除非您添加 `-Force` 参数，否则在 cmdlet 运行之前 Amazon Tools for PowerShell 会提示您进行确认。

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 删除对象（Amazon S3 API 参考）
- DeleteBucket（Amazon S3 API 参考）

将内联文本内容上传到 Amazon S3

`Write-S3Object` 支持将内联文本内容上传到 Amazon S3 的功能。通过使用 `-Content` 参数（别名 `-Text`），您可以指定应上传到 Amazon S3 的基于文本的内容，而不必首先将其放置到文件中。该参数接受简单的一行字符串，以及此处包含多行的字符串。

```
PS > # Specifying content in-line, single line text:
PS > write-s3object mybucket -key myobject.txt -content "file content"

PS > # Specifying content in-line, multi-line text: (note final newline needed to end in-
line here-string)
PS > write-s3object mybucket -key myobject.txt -content @"
>> line 1
>> line 2
>> line 3
>> "@
>>
PS > # Specifying content from a variable: (note final newline needed to end in-line here-
string)
PS > $x = @"
>> line 1
>> line 2
>> line 3
>> "@
>>
PS > write-s3object mybucket -key myobject.txt -content $x
```

IAM 与 Tools for PowerShell

本节介绍与 Amazon Identity and Access Management (IAM) 相关的一些常见任务以及如何使用 Amazon Tools for PowerShell 执行这些任务。

此处显示的命令假设您已为 PowerShell 会话设置了默认凭证和默认区域。因此，凭证和区域未包含在 cmdlet 的调用中。

主题

- 创建新的 IAM 用户和组 (p. 52)

- 为 IAM 用户设置 IAM 策略 (p. 53)
- 为 IAM 用户设置初始密码 (p. 53)

创建新的 IAM 用户和组

此部分介绍如何创建新的 IAM 组和新的 IAM 用户，然后将该用户添加到该组中。

首先，使用 `New-IAMGroup` cmdlet 创建组。我们在此处包括了 `-Path` 参数，不过该参数是可选的。

```
PS > New-IAMGroup -Path "/ps-created-groups/" -GroupName "powerUsers"

Path       : /ps-created-groups/
GroupName  : powerUsers
GroupId    : AGPAJPHUEYD5XPCGIUH3E
Arn        : arn:aws:iam::455364113843:group/ps-created-groups/powerUsers
CreateDate : 11/20/2012 3:32:50 PM
```

然后，使用 `New-IAMUser` cmdlet 创建用户。与前例中相似，`-Path` 参数可选。

```
PS > New-IAMUser -Path "/ps-created-users/" -UserName "myNewUser"

Path       : /ps-created-users/
UserName   : myNewUser
UserId     : AIDAJOJSPSPXADHBT7IN6
Arn        : arn:aws:iam::455364113843:user/ps-created-users/myNewUser
CreateDate : 11/20/2012 3:26:31 PM
```

最后，使用 `Add-IAMUserToGroup` cmdlet 将用户添加到组中。

```
PS > Add-IAMUserToGroup -UserName myNewUser -GroupName powerUsers

ServiceResponse
---------------
Amazon.IdentityManagement.Model.AddUserToGroupResponse
```

要验证 `powerUsers` 组包含 `myNewUser`，请使用 `Get-IAMGroup` cmdlet。

```
PS > Get-IAMGroup -GroupName powerUsers

Group                          Users                          IsTruncated
 Marker
-----                          -----                          -----------
 ------
Amazon.IdentityManagement.... {myNewUser}                     False
```

您还可以使用 Amazon Web Services Management Console 查看 IAM 用户和组

- 用户视图
- 组视图

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 将新用户添加到您的Amazon账户（IAM 用户指南）
- CreateGroup（IAM 服务参考）

为 IAM 用户设置 IAM 策略

以下命令说明如何将 IAM 策略分配给 IAM 用户。下面指定的策略为用户提供“高级用户访问”。此策略与 IAM 控制台中提供的高级用户访问 策略模板相同。下面显示的策略名称遵循用于 IAM 策略模板（例如，高级用户访问 的模板）的命名约定。命名约定为

```
<template name>+<user name>+<date stamp>
```

为了指定策略文档，我们使用了 PowerShell here-string。我们将 here-string 的内容分配给一个变量，然后将此变量用作 `Write-IAMUserPolicy` 中的参数值。

```
PS > $policyDoc = @"
>> {
>>   "Version": "2012-10-17",
>>   "Statement": [
>>     {
>>       "Effect": "Allow",
>>       "NotAction": "iam:*",
>>       "Resource": "*"
>>     }
>>   ]
>> }
>> "@
>>

PS > Write-IAMUserPolicy -UserName myNewUser -PolicyName "PowerUserAccess-
myNewUser-201211201605" -PolicyDocument $policyDoc

ServiceResponse
---------------
Amazon.IdentityManagement.Model.PutUserPolicyResponse
```

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- 使用 Windows PowerShell“Here-Strings”
- PutUserPolicy

为 IAM 用户设置初始密码

下面的示例介绍如何使用 `New-IAMLoginProfile` cmdlet 为 IAM 用户设置初始密码。有关密码的字符限制和建议的更多详细信息，请参阅 IAM 用户指南中的密码策略选项。

```
PS > New-IAMLoginProfile -UserName myNewUser -Password "&!123!&"

UserName                                                 CreateDate
--------                                                 ----------
myNewUser                                                11/20/2012 4:23:05 PM
```

使用 `Update-IAMLoginProfile` cmdlet 更改 IAM 用户的密码。

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)

- 管理密码
- CreateLoginProfile

Amazon EC2 与 Tools for Windows PowerShell

您可以使用 Amazon Tools for PowerShell 执行与 Amazon EC2 相关的常见任务。

此处显示的示例命令假设您已为 PowerShell 会话设置了默认凭证和默认区域。因此，我们在调用 cmdlet 时不包括凭证或区域。有关更多信息，请参阅Amazon Tools for Windows PowerShell入门 (p. 22)。

主题

创建密钥对

以下 `New-EC2KeyPair` 示例创建密钥对并将其存储在 PowerShell 变量 `$myPSKeyPair` 中。

```
PS > $myPSKeyPair = New-EC2KeyPair -KeyName myPSKeyPair
```

将密钥对对象通过管道传输到 `Get-Member` cmdlet 中以查看对象的结构。

```
PS > $myPSKeyPair | Get-Member

     TypeName: Amazon.EC2.Model.KeyPair

  Name                MemberType   Definition
  ----                ----------   ----------
  Equals              Method       bool Equals(System.Object obj)
  GetHashCode         Method       int GetHashCode()
  GetType             Method       type GetType()
  ToString            Method       string ToString()
  KeyFingerprint      Property     System.String KeyFingerprint {get;set;}
  KeyMaterial         Property     System.String KeyMaterial {get;set;}
  KeyName             Property     System.String KeyName {get;set;}
```

将密钥对对象通过管道传输到 `Format-List` cmdlet 以查看 `KeyName`、`KeyFingerprint` 和 `KeyMaterial` 成员的值。（为了便于阅读，输入内容已截断。）

```
PS > $myPSKeyPair | Format-List KeyName, KeyFingerprint, KeyMaterial

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
  KeyMaterial    : ----BEGIN RSA PRIVATE KEY----
                   MIIEogIBAAKCAQEAkK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                   Mz6btoxPcE7EMeH1wySUp8nouAS9xbl9l7+VkD74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5il/MpkrRogHqOB0rigeTeV5Yc3lvO0RFFPu0Kz4kcm...
                   w3Jg8dKsWn0plOpX7V3sRC02KgJIbejQUvBFGi5OQK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvgWkcFQkLmRHRoDpPb+OdFsZtjHZDpMVFmA9tT8EdbkEF...
                   3SrNeqZPsxJJIxOodb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
```

```
                    GGlLfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZWa7KFNRZuCuX7jssC...
                    xO98abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwfLtwhmJEy...
                    1BX9X8WFX/A8VLHrT1elrKmLkNECgYEAwltkV1pOJAFhz9p7ZFEv...
                    vvVsPaF0Ev9bk9pqhx269PB5Ox2KokwCagDMMaYvasWobuLmNu/1...
                    lmwRx7KTeQ7W1J3OLgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                    63g6N6rk2FkHZX1E62BgbewUd3eZOS05Ip4VUdvtGcuc8/qa+e5C...
                    KXgyt9nl64pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                    gYALEI7m1jJKpHWAEs0hiemw5VmKyIZpzGstSJsFStERlAjiETDH...
                    YAtnI4J8dRyP9I7BOVOn3wNfIjk85gi1/0Oc+j8S65giLAfndWGR...
                    9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvrUkepIbUDTD...
                    AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hkOSmWhlZGWikJ5VizBf...
                    drkBr/vTKVRMTi3lVFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                    TTld5N6opgOXAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                    x3O2duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzcOajwieNrblr7c...
                    -----END RSA PRIVATE KEY-----
```

KeyMaterial 成员存储密钥对的私有密钥。公有密钥存储在Amazon中。您无法从Amazon检索公有密钥，但是，您可以通过将私有密钥的 KeyFingerprint 与Amazon返回的公有密钥的该值进行对比来验证公有密钥。

查看密钥对的指纹

您可以使用 Get-EC2KeyPair cmdlet 查看密钥对的指纹。

```
PS > Get-EC2KeyPair -KeyName myPSKeyPair | format-list KeyName, KeyFingerprint

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
```

存储私有密钥

要将私有密钥存储到文件中，请将 KeyFingerMaterial 成员通过管道传输到 Out-File cmdlet。

```
PS > $myPSKeyPair.KeyMaterial | Out-File -Encoding ascii myPSKeyPair.pem
```

在将私有密钥写入到文件中时，必须指定 -Encoding ascii。否则，openssl 等工具可能无法正确读取此文件。您可以使用类似于以下的命令来验证生成文件的格式是否正确：

```
PS > openssl rsa -check < myPSKeyPair.pem
```

（ openssl 工具未包含在 Amazon Tools for PowerShell 或 Amazon SDK for .NET 中。）

删除密钥对

您需要密钥对来启动和连接到实例。在用完密钥对后，您可以将其删除。要从Amazon中删除公有密钥，请使用 Remove-EC2KeyPair cmdlet。在出现提示时，按 Enter 可删除密钥对。

```
PS > Remove-EC2KeyPair -KeyName myPSKeyPair

Confirm
Performing the operation "Remove-EC2KeyPair (DeleteKeyPair)" on target "myPSKeyPair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

变量 $myPSKeyPair 仍存在于当前 PowerShell 会话中，并仍包含密钥对信息。myPSKeyPair.pem 文件也存在。但是，公有密钥不再有效，因为密钥对的私有密钥不再存储在Amazon中。

使用 Windows PowerShell 创建安全组

您可以使用Amazon Tools for PowerShell来创建和配置安全组。创建安全组时，您可以指定将其用于 EC2-Classic 还是 EC2-VPC。该操作以安全组 ID 作为响应。

如果您需要连接到您的实例，就必须配置安全组以允许 SSH 流量 (Linux) 或 RDP 流量 (Windows)。

主题

Prerequisites

您需要您的计算机的公有 IP 地址，该地址使用 CIDR 表示法。您可以通过一项服务来获取本地计算机的公有 IP 地址。例如，Amazon 提供以下服务：http://checkip.amazonaws.com/ 或 https://checkip.amazonaws.com/。要查找另一项可提供您的 IP 地址的服务，请使用搜索短语“what is my IP address”。如果您正通过 ISP 或从防火墙后面连接，没有静态 IP 地址，您需要找出客户端计算机可以使用的 IP 地址范围。

> Warning
>
> 如果您指定 `0.0.0.0/0`，则会启用来自世界上任何 IP 地址的流量。对于 SSH 和 RDP 协议，您可能考虑这在测试环境下短时间内是可以接受的，但它对于生产环境并不安全。对于生产环境，请确保仅授权从适当的单个 IP 地址或地址范围进行访问。

为 EC2-Classic 创建安全组

以下示例使用 `New-EC2SecurityGroup` cmdlet 为 `EC2-Classic` 创建安全组。

```
PS > New-EC2SecurityGroup -GroupName myPSSecurityGroup -GroupDescription "EC2-Classic from
 PowerShell"

sg-0a346530123456789
```

要查看安全组的初始配置，请使用 `Get-EC2SecurityGroup` cmdlet。

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

Description         : EC2-Classic from PowerShell
GroupId             : sg-0a346530123456789
GroupName           : myPSSecurityGroup
IpPermissions       : {}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-9668ddef
```

要配置安全组以允许 TCP 端口 22 (SSH) 和 TCP 端口 3389 上的传入流量，请使用 `Grant-EC2SecurityGroupIngress` cmdlet。例如，以下示例脚本显示如何可以从单个 IP 地址 `203.0.113.25/32` 启用 SSH 流量。

```
$cidrBlocks = New-Object 'collections.generic.list[string]'
$cidrBlocks.add("203.0.113.25/32")
$ipPermissions = New-Object Amazon.EC2.Model.IpPermission
$ipPermissions.IpProtocol = "tcp"
```

```
$ipPermissions.FromPort = 22
$ipPermissions.ToPort = 22
ipPermissions.IpRanges = $cidrBlocks
Grant-EC2SecurityGroupIngress -GroupName myPSSecurityGroup -IpPermissions $ipPermissions
```

要验证已更新安全组，请再次运行 Get-EC2SecurityGroup cmdlet。请注意，您不能为 EC2-Classic 指定出站规则。

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-0a346530123456789
Description         : EC2-Classic from PowerShell
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
VpcId               :
Tags                : {}
```

要查看安全组规则，请使用 IpPermissions 属性。

```
PS > (Get-EC2SecurityGroup -GroupNames myPSSecurityGroup).IpPermissions

IpProtocol       : tcp
FromPort         : 22
ToPort           : 22
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}
```

为 EC2-VPC 创建安全组

以下 New-EC2SecurityGroup 示例将添加 -VpcId 参数为指定的 VPC 创建安全组。

```
PS > $groupid = New-EC2SecurityGroup `
    -VpcId "vpc-da0013b3" `
    -GroupName "myPSSecurityGroup" `
    -GroupDescription "EC2-VPC from PowerShell"
```

要查看安全组的初始配置，请使用 Get-EC2SecurityGroup cmdlet。默认情况下，VPC 的安全组中包含允许所有出站流量的规则。请注意，您不能通过名称引用 EC2-VPC 的安全组。

```
PS > Get-EC2SecurityGroup -GroupId sg-5d293231

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-5d293231
Description         : EC2-VPC from PowerShell
IpPermissions       : {}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
VpcId               : vpc-da0013b3
Tags                : {}
```

要为 TCP 端口 22 (SSH) 和 TCP 端口 3389 上的入站流量定义权限，请使用 New-Object cmdlet。以下示例脚本为从来自单个 IP 地址 203.0.113.25/32 的 TCP 端口 22 和 3389 定义权限。

```
$ip1 = new-object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
```

```
$ip1.IpRanges.Add("203.0.113.25/32")
$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")
Grant-EC2SecurityGroupIngress -GroupId $groupid -IpPermissions @( $ip1, $ip2 )
```

要验证已更新安全组，请再次使用 `Get-EC2SecurityGroup` cmdlet。

```
PS > Get-EC2SecurityGroup -GroupIds sg-5d293231

OwnerId             : 123456789012
GroupName           : myPSSecurityGroup
GroupId             : sg-5d293231
Description         : EC2-VPC from PowerShell
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
VpcId               : vpc-da0013b3
Tags                : {}
```

要查看入站规则，您可以从上一个命令返回的集合对象中检索 `IpPermissions` 属性。

```
PS > (Get-EC2SecurityGroup -GroupIds sg-5d293231).IpPermissions

IpProtocol       : tcp
FromPort         : 22
ToPort           : 22
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}

IpProtocol       : tcp
FromPort         : 3389
ToPort           : 3389
UserIdGroupPairs : {}
IpRanges         : {203.0.113.25/32}
```

使用 Windows PowerShell 查找 Amazon Machine Image

在启动 Amazon EC2 实例时，您需要指定一个 Amazon Machine Image (AMI) 以用作该实例的模板。不过，Amazon Windows AMI 的 ID 会频繁发生更改，因为Amazon会提供新 AMI 以及最新的更新和安全增强。可以使用 Get-EC2Image 和 Get-EC2ImageByName cmdlet 来查找当前 Windows AMI 并获取其 ID。

主题

Get-EC2Image

`Get-EC2Image` cmdlet 会检索您可使用的 AMI 的列表。

将 `-Owner` 参数与数组值 `amazon, self` 结合使用，以便 `Get-EC2Image` 仅检索属于 Amazon 或您的 AMI。在此上下文中，您指的是您要使用其凭证来调用 cmdlet 的用户。

```
PS > Get-EC2Image -Owner amazon, self
```

可使用 `-Filter` 参数限定结果的范围。要指定筛选条件，可创建类型为 `Amazon.EC2.Model.Filter` 的对象。例如，使用以下筛选条件可以只显示 Windows AMI。

```
$platform_values = New-Object 'collections.generic.list[string]'
$platform_values.add("windows")
$filter_platform = New-Object Amazon.EC2.Model.Filter -Property @{Name = "platform"; Values
 = $platform_values}
Get-EC2Image -Owner amazon, self -Filter $filter_platform
```

以下示例是 cmdlet 返回的 AMI 之一；上一条命令的实际输出提供了多个 AMI 的信息。

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc…}
CreationDate        : 2019-06-12T10:41:31.000Z
Description         : Microsoft Windows Server 2019 Full Locale English with SQL Web 2017
 AMI provided by Amazon
EnaSupport          : True
Hypervisor          : xen
ImageId             : ami-000226b77608d973b
ImageLocation       : amazon/Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2019-English-Full-SQL_2017_Web-2019.06.12
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

Get-EC2ImageByName

`Get-EC2ImageByName` cmdlet 使您能够基于所需的服务器配置类型筛选 Amazon Windows AMI 的列表。

在没有参数的情况下运行时（如下所示），cmdlet 会发出整组当前筛选条件名称：

```
PS > Get-EC2ImageByName

WINDOWS_2016_BASE
WINDOWS_2016_NANO
WINDOWS_2016_CORE
WINDOWS_2016_CONTAINER
WINDOWS_2016_SQL_SERVER_ENTERPRISE_2016
WINDOWS_2016_SQL_SERVER_STANDARD_2016
WINDOWS_2016_SQL_SERVER_WEB_2016
WINDOWS_2016_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_BASE
WINDOWS_2012R2_CORE
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2016
WINDOWS_2012R2_SQL_SERVER_STANDARD_2016
WINDOWS_2012R2_SQL_SERVER_WEB_2016
WINDOWS_2012R2_SQL_SERVER_EXPRESS_2014
WINDOWS_2012R2_SQL_SERVER_STANDARD_2014
WINDOWS_2012R2_SQL_SERVER_WEB_2014
WINDOWS_2012_BASE
```

```
WINDOWS_2012_SQL_SERVER_EXPRESS_2014
WINDOWS_2012_SQL_SERVER_STANDARD_2014
WINDOWS_2012_SQL_SERVER_WEB_2014
WINDOWS_2012_SQL_SERVER_EXPRESS_2012
WINDOWS_2012_SQL_SERVER_STANDARD_2012
WINDOWS_2012_SQL_SERVER_WEB_2012
WINDOWS_2012_SQL_SERVER_EXPRESS_2008
WINDOWS_2012_SQL_SERVER_STANDARD_2008
WINDOWS_2012_SQL_SERVER_WEB_2008
WINDOWS_2008R2_BASE
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2012
WINDOWS_2008R2_SQL_SERVER_STANDARD_2012
WINDOWS_2008R2_SQL_SERVER_WEB_2012
WINDOWS_2008R2_SQL_SERVER_EXPRESS_2008
WINDOWS_2008R2_SQL_SERVER_STANDARD_2008
WINDOWS_2008R2_SQL_SERVER_WEB_2008
WINDOWS_2008RTM_BASE
WINDOWS_2008RTM_SQL_SERVER_EXPRESS_2008
WINDOWS_2008RTM_SQL_SERVER_STANDARD_2008
WINDOWS_2008_BEANSTALK_IIS75
WINDOWS_2012_BEANSTALK_IIS8
VPC_NAT
```

要缩小返回的映像集，请使用 Names 参数指定一个或多个筛选条件名称。

```
PS > Get-EC2ImageByName -Names WINDOWS_2016_CORE

Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc…}
CreationDate        : 2019-08-16T09:36:09.000Z
Description         : Microsoft Windows Server 2016 Core Locale English AMI provided by
 Amazon
EnaSupport          : True
Hypervisor          : xen
ImageId             : ami-06f2a2afca06f15fc
ImageLocation       : amazon/Windows_Server-2016-English-Core-Base-2019.08.16
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2016-English-Core-Base-2019.08.16
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

使用 Windows PowerShell 启动 Amazon EC2 实例

要启动 Amazon EC2 实例，您需要您在之前的章节中创建的密钥对和安全组。您还需要 Amazon Machine Image (AMI) 的 ID。有关更多信息，请参阅以下文档：

- 创建密钥对 (p. 54)
- 使用 Windows PowerShell 创建安全组 (p. 56)
- 使用 Windows PowerShell 查找 Amazon Machine Image (p. 58)

Important

如果您启动不在免费套餐范围内的实例，那么在启动实例后您将需要付费，费用按实例的运行时间计算，即使实例处于闲置状态也需付费。

主题

在 EC2-Classic 中启动实例

以下命令创建并启动一个 t1.micro 实例。

```
PS > New-EC2Instance -ImageId ami-c49c0dac `
    -MinCount 1 `
    -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroups myPSSecurityGroup `
    -InstanceType t1.micro

ReservationId   : r-b70a0ef1
OwnerId         : 123456789012
RequesterId     :
Groups          : {myPSSecurityGroup}
GroupName       : {myPSSecurityGroup}
Instances       : {}
```

最初，您的实例处于 pending 状态，但在几分钟后将进入 running 状态。要查看有关您的实例的信息，请使用 Get-EC2Instance cmdlet。如果您有多个实例，则可以使用 Filter 参数按照预留 ID 筛选结果。首先，创建类型为 Amazon.EC2.Model.Filter 的对象。接下来，调用使用该过滤器的 Get-EC2Instance，然后显示 Instances 属性。

```
PS > $reservation = New-Object 'collections.generic.list[string]'
PS > $reservation.add("r-5caa4371")
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =
 "reservation-id"; Values = $reservation}
PS > (Get-EC2Instance -Filter $filter_reservation).Instances

AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           :
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    :
ImageId               : ami-c49c0dac
InstanceId            : i-5203422c
InstanceLifecycle     :
InstanceType          : t1.micro
KernelId              :
KeyName               : myPSKeyPair
LaunchTime            : 12/2/2018 3:38:52 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        :
PrivateIpAddress      : 10.25.1.11
ProductCodes          : {}
```

```
PublicDnsName         :
PublicIpAddress       : 198.51.100.245
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {myPSSecurityGroup}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              :
Tags                  : {}
VirtualizationType    : hvm
VpcId                 :
```

在 VPC 中启动实例

以下命令在指定的私有子网中创建一个 m1.small 实例。安全组必须对指定的子网有效。

```
PS > New-EC2Instance `
    -ImageId ami-c49c0dac `
    -MinCount 1 -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroupId sg-5d293231 `
    -InstanceType m1.small `
    -SubnetId subnet-d60013bf

ReservationId   : r-b70a0ef1
OwnerId         : 123456789012
RequesterId     :
Groups          : {}
GroupName       : {}
Instances       : {}
```

最初，您的实例处于 pending 状态，但在几分钟后将进入 running 状态。要查看有关您的实例的信息，请使用 Get-EC2Instance cmdlet。如果您有多个实例，则可以使用 Filter 参数按照预留 ID 筛选结果。首先，创建类型为 Amazon.EC2.Model.Filter 的对象。接下来，调用使用该过滤器的 Get-EC2Instance，然后显示 Instances 属性。

```
PS > $reservation = New-Object 'collections.generic.list[string]'
PS > $reservation.add("r-b70a0ef1")
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =
 "reservation-id"; Values = $reservation}
PS > (Get-EC2Instance -Filter $filter_reservation).Instances

AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           :
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    :
ImageId               : ami-c49c0dac
InstanceId            : i-5203422c
InstanceLifecycle     :
InstanceType          : m1.small
KernelId              :
KeyName               : myPSKeyPair
LaunchTime            : 12/2/2018 3:38:52 PM
Monitoring            : Amazon.EC2.Model.Monitoring
```

```
NetworkInterfaces     : {}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        :
PrivateIpAddress      : 10.25.1.11
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       : 198.51.100.245
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {myPSSecurityGroup}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-d60013bf
Tags                  : {}
VirtualizationType    : hvm
VpcId                 : vpc-a01106c2
```

在 VPC 中启动 Spot 实例

以下示例脚本在指定的子网中请求一个 Spot 实例。该安全组必须是您为包含指定子网的 VPC 创建的安全组。

```
$interface1 = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$interface1.DeviceIndex = 0
$interface1.SubnetId = "subnet-b61f49f0"
$interface1.PrivateIpAddress = "10.0.1.5"
$interface1.Groups.Add("sg-5d293231")
Request-EC2SpotInstance `
    -SpotPrice 0.007 `
    -InstanceCount 1 `
    -Type one-time `
    -LaunchSpecification_ImageId ami-7527031c `
    -LaunchSpecification_InstanceType m1.small `
    -Region us-west-2 `
    -LaunchSpecification_NetworkInterfaces $interface1
```

Amazon Lambda、 和 Amazon Tools for PowerShell

通过使用 AWSLambdaPSCore 模块，您可以使用 .NET Core 2.1 运行时在 PowerShell Core 6.0 中开发 Amazon Lambda 函数。PowerShell 开发人员可以使用 Lambda 在 PowerShell 环境中管理Amazon资源和编写自动化脚本。借助于 Lambda 中的 PowerShell 支持，您可以运行 PowerShell 脚本或函数以响应任何 Lambda 事件，如 Amazon S3 事件或 Amazon CloudWatch 计划事件。AWSLambdaPSCore 模块是一个适用于 PowerShell 的单独Amazon模块；它不是 Amazon Tools for PowerShell 的一部分，在安装 AWSLambdaPSCore 模块时，也不会安装 Amazon Tools for PowerShell。

安装 AWSLambdaPSCore 模块后，您可以使用任何可用的 PowerShell 命令（或开发自己的命令）来编写无服务器函数。Amazon Lambda Tools for PowerShell 模块包含基于 PowerShell 的无服务器应用程序的项目模板，以及用于将项目发布到Amazon的工具。

支持 Lambda 的所有区域中现已提供 AWSLambdaPSCore 模块支持。有关支持的区域的更多信息，请参阅Amazon区域列表。

Prerequisites

需要先执行以下步骤，然后才能安装和使用 AWSLambdaPSCore 模块。有关这些步骤的更多详细信息，请参阅 Amazon Lambda 开发人员指南中的设置 PowerShell 开发环境。

- 安装正确的 PowerShell 版本 – Lambda 的 PowerShell 支持基于跨平台的 PowerShell Core 6.0 版本。您可以在 Windows、Linux 或 Mac 上开发 PowerShell Lambda 函数。如果未安装该版本的 PowerShell，请参阅 Microsoft PowerShell 文档网站上提供的说明。
- 安装 .NET Core 2.1 开发工具包 – PowerShell Core 基于 .NET Core，因此，对 PowerShell 的 Lambda 支持使用相同的 .NET Core 2.1 Lambda 运行时开发 .NET Core 和 PowerShell Lambda 函数。Lambda PowerShell 发布 cmdlet 使用 .NET Core 2.1 开发工具包创建 Lambda 部署程序包。在 Microsoft 下载中心提供了 .NET Core 2.1 开发工具包。请务必安装开发工具包，而不是运行时。

安装 AWSLambdaPSCore 模块

在满足先决条件后，您便可以安装 AWSLambdaPSCore 模块。在 PowerShell Core 会话中运行以下命令。

```
PS> Install-Module AWSLambdaPSCore -Scope CurrentUser
```

您可以直接在 PowerShell 中开始开发 Lambda 函数。有关如何开始使用的更多信息，请参阅 Amazon Lambda 开发人员指南中的编写 PowerShell 中的 Lambda 函数的编程模型。

另请参阅

- 在Amazon开发人员博客上发布对 PowerShell Core 的 Lambda 支持
- PowerShell 库网站上的 AWSLambdaPSCore 模块
- 设置 PowerShell 开发环境
- Amazon GitHub 上的 Lambda Tools for Powershell
- Amazon Lambda 控制台

Amazon SQS、Amazon SNS 与 Tools for Windows PowerShell

本节提供演示如何执行以下操作的示例：

- 创建 Amazon SQS 队列并获取队列 ARN（Amazon 资源名称）。
- 创建 Amazon SNS 主题。
- 向 SNS 主题授予权限，以便该主题向队列发送消息。
- 为队列订阅 SNS 主题
- 向 IAM 用户或Amazon账户授予权限，以便上述用户或账户能够发布 SNS 主题并读取来自 SQS 队列的消息。
- 通过向主题发布消息并读取来自队列的消息来验证结果。

创建 Amazon SQS 队列并获取队列 ARN

以下命令在您的默认区域中创建 SQS 队列。输出会显示新队列的 URL。

```
PS > New-SQSQueue -QueueName myQueue
https://sqs.us-west-2.amazonaws.com/123456789012/myQueue
```

以下命令检索队列的 ARN。

```
PS > Get-SQSQueueAttribute -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/
myQueue -AttributeName QueueArn
...
QueueARN               : arn:aws:sqs:us-west-2:123456789012:myQueue
...
```

创建 Amazon SNS 主题

以下命令在您的默认区域中创建 SNS 主题，并返回新主题的 ARN。

```
PS > New-SNSTopic -Name myTopic
arn:aws:sns:us-west-2:123456789012:myTopic
```

向 SNS 主题授予权限

下面的示例脚本创建一个 SQS 队列和一个 SNS 主题，并授予对 SNS 主题的权限，以便它可以将消息发送到 SQS 队列：

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeNames "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": "*",
        "Action": "SQS:SendMessage",
        "Resource": "$qarn",
        "Condition": { "ArnEquals": { "aws:SourceArn": "$topicarn" } }
    }
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

为队列订阅 SNS 主题

以下命令将队列 `myQueue` 订阅到 SNS 主题 `myTopic`，并返回订阅 ID：

```
PS > Connect-SNSNotification `
    -TopicARN arn:aws:sns:us-west-2:123456789012:myTopic `
    -Protocol SQS `
```

```
    -Endpoint arn:aws:sqs:us-west-2:123456789012:myQueue
arn:aws:sns:us-west-2:123456789012:myTopic:f8ff77c6-e719-4d70-8e5c-a54d41feb754
```

授予权限

以下命令授予对主题 sns:Publish 执行 myTopic 操作的权限。

```
PS > Add-SNSPermission `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Label ps-cmdlet-topic `
    -AWSAccountIds 123456789012 `
    -ActionNames publish
```

以下命令授予对队列 sqs:ReceiveMessage 执行 sqs:DeleteMessage 和 myQueue 操作的权限。

```
PS > Add-SQSPermission `
    -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/myQueue `
    -AWSAccountId "123456789012" `
    -Label queue-permission `
    -ActionName SendMessage, ReceiveMessage
```

验证结果

以下命令通过向 SNS 主题 myTopic 发布消息来测试新队列和主题，并返回 MessageId。

```
PS > Publish-SNSMessage `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
    -Message "Have A Nice Day!"
728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

以下命令从 SQS 队列 myQueue 检索消息并显示此消息。

```
PS > Receive-SQSMessage -QueueUrl https://sqs.us-west-2.amazonaws.com/123456789012/myQueue

Attributes             : {}
Body                   : {
                           "Type" : "Notification",
                           "MessageId" : "491c687d-b78d-5c48-b7a0-3d8d769ee91b",
                           "TopicArn" : "arn:aws:sns:us-west-2:123456789012:myTopic",
                           "Message" : "Have A Nice Day!",
                           "Timestamp" : "2019-09-09T21:06:27.201Z",
                           "SignatureVersion" : "1",
                           "Signature" : "llE17A2+XOuJZnw3TlgcXz4C4KPLXZxbxoEMIirelhl3u/
oxkWmz5+9tJKFMns1ZOqQvKxk+ExfEZcD5yWt6biVuBb8pyRmZ1bO3hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
                         vyuGkJReV7lOJWPHnCN
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yoRs1IkbLV
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
                         kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvr3WbaSvg==",
                           "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-6aad65c2f9911b05cd53efda11f913f9.pem",
                           "UnsubscribeURL" :
                         "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:myTopic:22b77de7-
a216-4000-9a23-bf465744ca84"
                         }
MD5OfBody              : 5b5ee4f073e9c618eda3718b594fa257
MD5OfMessageAttributes :
MessageAttributes      : {}
```

```
MessageId              : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
ReceiptHandle          :
 AQEB2vvk1e5cOKFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJbl7F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHnl8SlbQa0R/kD+Saqa4OIvfj8x3M4Oh1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
                        +HmXdkax2Wd+9AxrHlQZV5ur1MoByKWWbDbsqoYJTJquCclOgWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
                        sHN12776axknhg3j9K/Xwj54DixdsegnrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2MseyOeM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==
```

Amazon Tools for Windows PowerShell 中的 CloudWatch

本节显示的示例说明了如何使用 Tools for Windows PowerShell 将自定义指标数据发布到 CloudWatch。

该示例假设您已为 PowerShell 会话设置默认凭证和默认区域。

将自定义指标发布到您的 CloudWatch 控制面板

以下 PowerShell 代码可初始化 CloudWatch `MetricDatum` 对象并将其发布到服务。您可以导航到 CloudWatch 控制台以查看该操作的结果。

```
$dat = New-Object Amazon.CloudWatch.Model.MetricDatum
$dat.Timestamp = (Get-Date).ToUniversalTime()
$dat.MetricName = "New Posts"
$dat.Unit = "Count"
$dat.Value = ".50"
Write-CWMetricData -Namespace "Usage Metrics" -MetricData $dat
```

请注意以下几点：

- 用于初始化 `$dat.Timestamp` 的日期时间信息必须用世界时 (UTC) 表示。
- 用于初始化 `$dat.Value` 的值可以是括在引号中的字符串值或数字值（无引号）。该示例显示了一个字符串值。

另请参阅

- 使用 Amazon Tools for PowerShell (p. 45)
- AmazonCloudWatchClient.PutMetricData（.NET 开发工具包参考）
- MetricDatum（服务 API 参考）
- Amazon CloudWatch 控制台

Amazon Tools for PowerShell 中的安全性

Amazon的云安全性的优先级最高。作为 Amazon 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。 责任共担模式 将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为Amazon 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 Amazon Tools for PowerShell 的合规性计划，请参阅合规性计划范围内的Amazon服务。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，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。

此文档将帮助您了解如何在使用 Amazon Tools for PowerShell 时应用责任共担模式。以下主题说明如何配置 Amazon Tools for PowerShell 以实现您的安全性和合规性目标。您还将了解如何使用 Amazon Tools for PowerShell 来帮助您监控和保护 Amazon 资源。

主题

Amazon Tools for PowerShell 中的数据保护

Amazon责任共担模式适用于 Amazon Tools for PowerShell 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的Amazon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 Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 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 中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 终端节点。有关可用的 FIPS 终端节点的更多信息，请参阅美国联邦信息处理标准 (FIPS) 第 140-2 版。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用控制台、API、Amazon Tools for PowerShell 或 Amazon 开发工具包处理 Amazon CLI 或其他 Amazon 服务时。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果向外部服务器提供 URL，强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

数据加密

所有安全服务均具有一项重要功能，即信息在未处于活动使用状态时都会加密。

静态加密

除了代表用户与Amazon服务交互所需的凭证以外，Amazon Tools for PowerShell 本身不存储任何客户数据。

如果您使用 Amazon Tools for PowerShell 来调用 Amazon 服务，此服务将客户数据传输到本地计算机进行存储，则请参阅该服务的《用户指南》中的“安全与合规”章节，了解如何存储、保护和加密数据的相关信息。

传输中加密

默认情况下，从运行 Amazon Tools for PowerShell 和 Amazon 服务终端节点的客户端计算机上传输的所有数据，均通过使用 HTTPS/TLS 连接发送所有内容来加密。

您无需执行任何操作即可使用 HTTPS/TLS。它始终处于启用状态。

适用于 Amazon Tools for PowerShell 的 Identity and Access Management

Amazon Tools for PowerShell 使用您在通过Amazon Web Services Management Console访问Amazon资源时使用的相同 IAM 用户和角色。授予权限的策略也相同，因为Amazon Tools for PowerShell调用服务控制台使用的相同 API 操作。有关更多信息，请参阅所要使用 Amazon 服务的“安全性”章节中的“Identity and Access Management”部分。

唯一的主要区别在于使用标准 IAM 用户和长期凭证时如何进行身份验证。尽管 IAM 用户需要密码才能访问Amazon服务的控制台，但相同的 IAM 用户需要访问密钥（而不是密码）才能使用 Amazon Tools for PowerShell 执行相同的操作。所有其他短期凭证的使用方式与在控制台中使用时相同。

Amazon Tools for PowerShell使用的凭证通常存储在纯文本文件中，并且不 加密。但是，您可以选择当在 Windows 上运行时使用加密的 .NET 软件开发工具包凭证存储。

- `$HOME/.aws/credentials` 文件存储访问 Amazon 资源所需的长期凭证。这包括访问密钥 ID 和秘密访问密钥。

风险防范

- 我们强烈建议您在 `$HOME/.aws` 文件夹及其子文件夹和文件上配置文件系统权限，仅限授权用户访问。
- 尽可能使用具有临时凭证的角色，以减少凭证泄露时造成损坏的机会。仅使用长期凭证来请求和刷新短期角色凭证。

Amazon Tools for PowerShell 的合规性验证

作为多个 Amazon 合规性计划的一部分，第三方审计员将评估 Amazon 服务的安全性和合规性。使用Amazon Tools for PowerShell访问服务不会改变该服务的合规性。

有关特定合规性计划范围内的 Amazon 服务列表，请参阅合规性计划范围内的 Amazon 服务。有关常规信息，请参阅Amazon 合规性计划。

您可以使用下载第三方审计报告Amazon Artifact 有关更多信息，请参阅下载 Amazon Artifact 中的报告。

您使用 Amazon Tools for PowerShell 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。Amazon 提供以下资源来帮助满足合规性：

- 安全性与合规性 Quick Start 指南安全性与合规性 Quick Start 指南 - 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署基于安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用Amazon创建符合 HIPAA 标准的应用程序。
- Amazon合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config 开发人员指南中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub – 此Amazon服务提供了Amazon中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

文档历史记录

本主题介绍了对Amazon Tools for PowerShell文档的重大更改。

我们还会根据客户反馈定期更新文档。要发送有关某个主题的反馈，请使用“此页面对您有帮助吗？”旁边的反馈按钮，其位于每个页面的底部。

有关对Amazon Tools for PowerShell的更改和更新的更多信息，请参阅发行说明。

update-history-change	更新-历史记录-描述	更新-历史记录-日期
Amazon Tools for PowerShell 版本 4 (p. 71)	添加了有关版本 4 的信息，包括适用于 Windows 和 Linux/macOS 的安装说明，以及描述与版本 3 的差异的迁移主题和对新功能的介绍。	2019 年 11 月 21 日
Amazon Tools for PowerShell 3.3.563 (p. 71)	添加了有关如何安装和使用 `AWS.Tools.Common` 模块预览版本的信息。这个新模块将较旧的整体程序包分为一个共享模块和每个Amazon服务中的一个模块。	2019 年 10 月 18 日
Amazon Tools for PowerShell 3.3.343.0 (p. 71)	在使用 Amazon Tools for PowerShell 部分中添加了信息，介绍可供 PowerShell Core 开发人员构建 Amazon Lambda 函数的 Amazon Lambda Tools for PowerShell。	2018 年 9 月 11 日
Amazon Tools for Windows PowerShell 3.1.31.0 (p. 71)	在入门部分中添加了有关新 cmdlet 的信息，它们使用安全断言标记语言 (SAML) 支持为用户配置联合身份。	2015 年 12 月 1 日
Amazon Tools for Windows PowerShell 2.3.19 (p. 71)	在 cmdlet 查找和别名部分中添加了有关新 `Get-AWSCmdletName` cmdlet 的信息，它们可以帮助用户更轻松地找到所需的 Amazon cmdlet。	2015 年 2 月 5 日
Amazon Tools for Windows PowerShell 1.1.1.0 (p. 71)	来自 cmdlet 的集合输出始终列举到 PowerShell 管道。自动支持可分页的服务调用。新 $AWSHistory shell 变量收集服务响应并选择性地为请求提供服务。AWSRegion 实例使用“Region”字段（而非 SystemName）来帮助进行管道传输。Remove-S3Bucket 支持 -DeleteObjects 开关选项。已解决 Set-AWSCredentials 的可用性问题。Initialize-AWSDefaults 从其获取凭证和区域数据的位置进行报告。Stop-EC2Instance 接受	2013 年 5 月 15 日

	Amazon.EC2.Model.Reservation 实例作为输入。通用的 List<T> 参数类型已替换为数组类型 (T[])。删除或终止资源的 cmdlet 会在删除前提示确认。Write-S3Object 支持将内联文本内容上传到 Amazon S3。	
Amazon Tools for Windows PowerShell 1.0.1.0 (p. 71)	Tools for Windows PowerShell 模块的安装位置已更改，以便使用 Windows PowerShell 版本 3 的环境可以利用自动加载功能。此模块和支持文件现已安装到 `AWS ToolsPowerShell` 下的 `AWSPowerShell` 子文件夹中。安装程序会自动删除位于 `AWS ToolsPowerShell` 文件夹中的早期版本文件。在该版本中更新了 `PSModulePath` for Windows PowerShell（所有版本）以包含该模块的父文件夹 (`AWS ToolsPowerShell`)。对于采用 Windows PowerShell 版本 2 的系统，已更新开始菜单快捷方式，以便从新位置导入模块，然后运行 `Initialize-AWSDefaults`。对于采用 Windows PowerShell 版本 3 的系统，已更新开始菜单快捷方式，以便删除 `Import-Module` 命令，而仅保留 `Initialize-AWSDefaults`。如果您已编辑您的 PowerShell 配置文件以便对 `Import-Module` 文件执行 `AWSPowerShell.psd1`，那么您需要更新配置文件以指向该文件的新位置 (或者，如果使用 PowerShell 版本 3，由于不再需要 `Import-Module` 语句，因此可将其删除)。鉴于这些更改，在执行 `Get-Module -ListAvailable` 时，Tools for Windows PowerShell 模块现在将作为可用的模块列出。此外，对于 Windows PowerShell 版本 3 用户，执行模块导出的任何 cmdlet 会在当前 PowerShell shell 中自动加载该模块，而不必先使用 `Import-Module`。这样当系统采用禁止执行脚本的执行策略时，可以交互使用 cmdlet。	2012 年 12 月 21 日
Amazon Tools for Windows PowerShell 1.0.0.0 (p. 71)	首次发布	2012 年 12 月 6 日