



用户指南

AWS Tools for PowerShell (第 4 版)



AWS Tools for PowerShell (第 4 版) : 用户指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

.....	x
什么是 AWS Tools for PowerShell ?	1
SDK 主要版本的维护和支持	1
AWS.Tools	2
AWSPowerShell。 NetCore	2
AWSPowerShell	3
如何使用本指南	3
此部分中的其他主题	4
修订历史记录	4
新增内容	4
安装	6
在 Windows 上安装	6
先决条件	7
安装 AWS.Tools	7
安装 AWSPowerShell.NetCore	9
安装 AWSPowerShell	10
启用脚本执行	11
版本控制	13
更新 AWS Tools for PowerShell	15
在 Linux 或 macOS 上安装	16
设置概述	16
先决条件	7
安装 AWS.Tools	17
安装 AWSPowerShell。 NetCore	20
脚本执行	11
配置控制 PowerShell 台	22
初始化您的 PowerShell 会话	22
版本控制	13
AWS Tools for PowerShell 在 Linux 或 macOS 上更新	23
相关信息	24
从 AWS Tools for PowerShell 版本 3.3 迁移到版本 4	24
全新的完全模块化 AWS.Tools 版本	25
新 Get-AWSService cmdlet	25
用于控制 Cmdlet 返回的对象的新 -Select 参数	26

更一致地限制输出中的项目数	27
更易于使用的流参数	28
按属性名称扩展管道	28
静态通用参数	29
AWS.Tools 声明和强制执行强制性参数	29
所有参数均可为 null	29
删除以前弃用的功能	30
开始使用	31
配置工具身份验证	31
启用和配置 IAM Identity Center	32
将适用于 PowerShell 的工具配置为使用 IAM Identity Center。	32
开始 AWS 访问门户会话	34
示例	34
其他信息	35
使用 AWS CLI	35
指定 AWS 区域	38
指定自定义或非标准终端节点	40
附加信息	40
配置联合身份	41
先决条件	41
联合身份用户如何获得对服务的联合访问权限 AWS APIs	41
SAML Support 在 AWS Tools for PowerShell	42
如何使用 PowerShell SAML 配置 Cmdlet	44
补充阅读	48
Cmdlet 发现和别名	48
Cmdlet 发现	48
cmdlet 命名和别名	54
管道、输出和迭代	58
管道	58
Cmdlet 输出	58
通过分页数据进行迭代	60
凭证和配置文件解析	61
凭证搜索顺序	61
用户和角色	62
用户和权限集	63
服务角色	63

使用旧版凭证	64
重要警告和指南	64
AWS 凭证	65
共享凭证	73
特征	78
可观测性	78
使用 AWS 服务	82
PowerShell 文件串联编码	82
PowerShell 工具返回的对象	82
Amazon EC2	83
Amazon S3	83
AWS Lambda 和 AWS Tools for PowerShell	83
Amazon SNS 和 Amazon SQS	83
CloudWatch	84
另请参阅	84
主题	84
Amazon S3 与 Tools for Windows PowerShell	84
创建 Amazon S3 存储桶，验证它的区域，然后删除它（可选）	85
将 Amazon S3 存储桶配置为网站并启用日志记录	86
将对象上传到 Amazon S3 存储桶	87
删除 Amazon S3 对象和存储桶	89
将内联文本内容上传到 Amazon S3	90
Amazon EC2 与 Tools for Windows PowerShell	91
创建密钥对	91
创建安全组	93
查找 AMI	96
启动实例	99
AWS Lambda 和 AWS Tools for PowerShell	102
先决条件	7
安装 AWSLambdaPSCore 模块	102
另请参阅	84
亚马逊 SQS、亚马逊 SNS 和适用于 Windows 的工具 PowerShell	103
创建 Amazon SQS 队列并获取队列 ARN	103
创建 Amazon SNS 主题	104
向 SNS 主题授予权限	104
为队列订阅 SNS 主题	104

授予权限	105
验证结果	105
CloudWatch 来自 AWS Tools for Windows PowerShell	106
将自定义指标发布到您的 CloudWatch 控制面板	106
另请参阅	84
使用 ClientConfig	107
使用 ClientConfig 参数	107
使用未定义的属性	108
指定 AWS 区域	109
代码示例	110
ACM	112
操作	112
Application Auto Scaling	116
操作	112
WorkSpaces 应用程序	123
操作	112
Aurora	150
操作	112
Auto Scaling	151
操作	112
AWS Budgets	185
操作	112
AWS Cloud9	186
操作	112
CloudFormation	193
操作	112
CloudFront	204
操作	112
CloudTrail	211
操作	112
CloudWatch	216
操作	112
CodeCommit	221
操作	112
CodeDeploy	226
操作	112

CodePipeline	245
操作	112
Amazon Cognito Identity	263
操作	112
AWS Config	267
操作	112
Device Farm	285
操作	112
Directory Service	286
操作	112
AWS DMS	311
操作	112
DynamoDB	312
操作	112
Amazon EC2	326
操作	112
Amazon ECR	455
操作	112
Amazon ECS	456
操作	112
Amazon EFS	462
操作	112
Amazon EKS	468
操作	112
Elastic Load Balancing——版本 1	481
操作	112
Elastic Load Balancing——版本 2	501
操作	112
Amazon FSx	524
操作	112
Amazon Glacier	532
操作	112
AWS Glue	536
操作	112
AWS Health	538
操作	112

IAM	539
操作	112
Kinesis	609
操作	112
Lambda	613
操作	112
Amazon ML	626
操作	112
Macie	631
操作	112
AWS 价目表	632
操作	112
资源组	635
操作	112
资源组标记 API	643
操作	112
Route 53	647
操作	112
Amazon S3	661
操作	112
Security Hub CSPM	696
操作	112
Amazon SES	697
操作	112
Amazon SES API v2	699
操作	112
Amazon SNS	700
操作	112
Amazon SQS	701
操作	112
AWS STS	713
操作	112
支持	717
操作	112
Systems Manager	723
操作	112

Amazon Translate	794
操作	112
AWS WAFV2	795
操作	112
WorkSpaces	796
操作	112
安全性	812
数据保护	812
数据加密	813
身份和访问管理	814
受众	814
使用身份进行身份验证	814
使用策略管理访问	816
如何 AWS 服务 使用 IAM	817
对 AWS 身份和访问进行故障排除	817
合规性验证	819
强制实施最低 TLS 版本	819
其它安全注意事项	819
敏感信息的记录	819
Cmdlet 参考	821
文档历史记录	822
.....	dccccxxviii

AWS Tools for PowerShell V4 的支持已到期。

我们建议您迁移到 [AWS Tools for PowerShell V5](#)。有关如何迁移的更多详细信息和信息，请参阅我们的[终止支持公告](#)。

什么是 AWS Tools for PowerShell ?

AWS Tools for PowerShell 是一组基于公开的功能构建的 PowerShell 模块 适用于 .NET 的 AWS SDK。AWS Tools for PowerShell 使您能够通过 PowerShell 命令行编写对 AWS 资源的操作脚本。

cmdlet 为指定参数和处理结果提供了一种惯用的 PowerShell 体验，即使它们是使用各种 AWS 服务 HTTP 查询实现的。APIs 例如，用于 AWS Tools for PowerShell 支持 PowerShell 管道的 cmdlet，也就是说，您可以通过管道将 PowerShell 对象传入和传出 cmdlet。

它们允许您灵活处理证书，包括对 AWS Identity and Access Management (IAM) 基础设施的支持。AWS Tools for PowerShell 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。

它们 AWS Tools for PowerShell 支持与 SDK 支持的相同服务和 AWS 区域集。你可以将它们安装 AWS Tools for PowerShell 在运行 Windows、Linux 或 macOS 操作系统的电脑上。

Note

AWS Tools for PowerShell 版本 4 (V4) 是 3.3 版的向后兼容更新。AWS Tools for PowerShell 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 V4 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 V4 中更改的更多信息，请参阅 [从 AWS Tools for PowerShell 版本 3.3 迁移到版本 4](#)。

AWS Tools for PowerShell 它们有以下三个不同的软件包可供选择：

- [AWS.Tools](#)
- [AWSPowerShell。NetCore](#)
- [AWSPowerShell](#)

SDK 主要版本的维护和支持

有关 SDK 主要版本及其底层依赖项的维护和支持的信息，请参阅《[AWS SDKs 和工具参考指南](#)》中的以下内容：

- [AWS SDKs 和工具维护政策](#)
- [AWS SDKs 和工具版本支持矩阵](#)

AWS.Tools-的模块化版本 AWS Tools for PowerShell

PowerShell Gallery **AWS.Tools.Installer**

PowerShell Gallery **AWS.Tools.Common**

ZIP Archive **AWS.Tools**

此版本 AWS Tools for PowerShell 是生产环境 PowerShell 中运行的任何计算机的推荐版本。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，并且大多数情况下，可以启用自动导入 `AWS.Tools cmdlet`，而无需先手动调用 `Import-Module`。

这是最新版本，可在所有支持的操作系统上运行，包括 Windows、Linux 和 macOS。AWS Tools for PowerShell 此软件包为每项 AWS 服务提供一个安装模块 `AWS.Tools.Common`、一个通用模块和一个模块，例如 `AWS.Tools.EC2`、`AWS.Tools.IdentityManagement`、`AWS.Tools.S3`、、、等。`AWS.Tools.Installer`

该 `AWS.Tools.Installer` 模块提供了 `cmdlet`，使您能够安装、更新和删除每项服务的模块。AWS 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。要将 `cmdlet` 用于 AWS 服务，只需运行命令即可。PowerShell 自动导入要运行其 `cmdlet` 的 AWS 服务的 `AWS.Tools.Common` 模块和模块。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。

可以在正在运行的计算机 AWS Tools for PowerShell 上安装此版本的：

- PowerShell Windows、Linux 或 macOS 上的酷睿 6.0 或更高版本。
- Windows PowerShell 5.1 或更高版本在 Windows 上使用 .NET 框架 4.7.2 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 -单模块版本的 AWS Tools for PowerShell

PowerShell Gallery **AWSPowerShell.NetCore**

ZIP Archive **AWSPowerShell.NetCore**

此版本由一个大型模块组成，该模块包含对所有 AWS 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。

可以在正在运行的计算机 AWS Tools for PowerShell 上安装此版本的：

- PowerShell Windows、Linux 或 macOS 上的酷睿 6.0 或更高版本。
- Windows PowerShell 3.0 或更高版本适用于安装 .NET Framework 4.7.2 或更高版本的 Windows

在本指南中，当我们只需要指定此版本时，我们用其模块名称来指代它：AWSPowerShell.
NetCore。

AWSPowerShell-适用于 Windows 的单模块版本 PowerShell

PowerShell Gallery **AWSPowerShell**

ZIP Archive **AWSPowerShell**

此版本 AWS Tools for PowerShell 仅与运行 Windows PowerShell 版本 2.0 到 5.1 的 Windows 计算机兼容，并且只能安装在这些计算机上。它与 PowerShell 酷睿 6.0 或更高版本或任何其他操作系统（Linux 或 macOS）不兼容。此版本由一个大型模块组成，该模块包含对所有 AWS 服务的支持。

在本指南中，当我们只需要指定此版本时，我们用其模块名称来指代它：AWSPowerShell。

如何使用本指南

本指南分为以下几个主要部分。

[安装 AWS Tools for PowerShell](#)

本节介绍如何安装 AWS Tools for PowerShell。它包括 AWS 如果您还没有账户，如何注册，以及如何创建可用于运行 cmdlet 的 IAM 用户。

[开始使用 AWS Tools for Windows PowerShell](#)

本节介绍使用的基础知识 AWS Tools for PowerShell，例如指定凭据和 AWS 区域、为特定服务查找 cmdlet 以及为 cmdlet 使用别名。

[使用中的 AWS 服务 AWS Tools for PowerShell](#)

本节包含有关使用 AWS Tools for PowerShell 执行一些最常见 AWS 任务的信息。

此部分中的其他主题

- [修订历史记录](#)
- [AWS Tools for PowerShell 中的新增内容](#)

修订历史记录

要了解不同版本中发生了哪些变化，请参阅以下内容：

- [更改日志](#)
- [AWS Tools for PowerShell 中的新增内容](#)
- [文档历史记录](#)

AWS Tools for PowerShell 中的新增内容

有关与 AWS Tools for PowerShell 相关的新开发的总体信息，请参阅产品页面 (<https://aws.amazon.com/powershell/>) 和 [变更日志](#)。

以下是 Tools for PowerShell 的新增功能。

2025 年 9 月 17 日：即将终止对 AWS Tools for PowerShell 版本 4 的支持

已宣布终止对 AWS Tools for PowerShell 此版本 (V4) 的支持。请参阅 [《V5 迁移指南》](#)，迁移您的 V4 脚本并避免中断。有关更多信息，请参阅博客文章 [Announcing the end-of-support for AWS Tools for PowerShell v4](#)。

2025 年 6 月 23 日：AWS Tools for PowerShell 版本 5

AWS Tools for PowerShell 的版本 5 (V5) 已正式发布！有关更多信息，请参阅 [《AWS Tools for PowerShell 用户指南 \(V5\)》](#)，尤其是 [迁移到 V5](#) 主题。

2025 年 2 月 10 日：可观测性的 GA 版本

可观测性是指可以根据系统发出的数据推断出其当前状态的程度。Tools for PowerShell 已添加可观测性，包括遥测提供程序的实现。有关更多信息，请参阅本指南中的 [可观测性](#) 和博客文章 [Announcing the general availability of AWS .NET OpenTelemetry libraries](#)。

2025 年 1 月 15 日：完整性保护的新默认行为

从 AWS Tools for PowerShell 4.1.737 版本开始，该工具通过自动计算上传文件的 CRC32 校验和来提供默认的完整性保护。有关更多信息，请参阅 GitHub 上的公告，网址为 <https://github.com/aws/aws-tools-for-powershell/issues/370>。该工具还提供了可在外部设置的数据完整性保护全局设置，您可以参阅 [《AWS SDK 和工具参考指南》](#) 中的 [数据完整性保护](#)。

2024 年 11 月 18 日：版本 5 (预览版 1) 发布

AWS Tools for PowerShell 版本 5 第 1 个预览版已于 2024 年 11 月 18 日发布。有关此预览版的更多信息，请参阅博客文章 [Preview 1 of AWS Tools for PowerShell V5](#)。

安装 AWS Tools for PowerShell

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

主题

- [在 Windows 上安装 AWS Tools for PowerShell](#)
- [AWS Tools for PowerShell 在 Linux 或 macOS 上安装](#)
- [从 AWS Tools for PowerShell 版本 3.3 迁移到版本 4](#)

在 Windows 上安装 AWS Tools for PowerShell

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

- [AWS.Tools](#) - 的模块化版本。。AWS Tools for PowerShell 每个 AWS 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](#) – AWS Tools for PowerShell 的单一大模块版本。所有 AWS 服务都由这个单一大模块支持。

Note

请注意，单个模块可能太大，无法与 [AWS Lambda](#) 函数一起使用。请改用上面显示的模块化版本。

- [AWSPowerShell](#) - 特定于传统 Windows 的单一大模块版本 AWS Tools for PowerShell。所有 AWS 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

AWS Tools for PowerShell 默认安装在所有基于 Windows 的 Amazon 系统映像 (AMI) 上。安装的选项取决于 AMI。许多 AMI 包含 `AWSPowerShell` 模块，但有些可能具有不同的选项。例如，用于 Windows Server 2025 的 Amazon EC2 AMI 使用模块化 `AWS.Tools` 选项。

设置适用于 AWS 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AWS 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AWS Tools for PowerShell模块导入您的 PowerShell 会话。

先决条件

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AWS 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.Installer** 模块中的 cmdlet。该模块简化了其他 **AWS.Tools** 模块的安装和更新。**AWS.Tools.Installer** 需要 `PowerShellGet`，并自动下载和安装其更新版本。**AWS.Tools.Installer** 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。

该方法在以下过程中描述。

- 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。您可以通过显示 `PSModulePath` 变量的值来查找模块文件夹。

Warning

下载 ZIP 文件后并在提取内容之前，您可能需要将其解除锁定。通常，通过打开文件的属性，查看常规选项卡，并选择解除锁定复选框（如果存在）即可完成这一操作。

如果 ZIP 文件需要解除锁定但您没有这样做，您可能会收到类似这样的错误：“Import-Module：无法加载文件或程序集”。

- 使用 `Install-Module` cmdlet 从 PowerShell 库安装每个服务模块。

使用 **AWS.Tools.Installer** 模块在 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 **AWS 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 Amazon EC2 和 Amazon S3 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 AWS 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 PSGallery 的 PSRepository (<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，前面的命令会将模块安装到 `%USERPROFILE%\Documents\WindowsPowerShell\Modules` 文件夹中。若要为计算机的所有用户安装 AWS 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。例如，以下命令将 IAM 模块安装到所有用户均可访问的 `%ProgramFiles%\WindowsPowerShell\Modules` 文件夹中。

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

要安装其他模块，请使用相应的模块名称运行类似的命令，可参阅 [PowerShell 库](#)。

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。您可以通过显示 `PSModulePath` 变量的值来查找模块目录。

⚠ Warning

下载 ZIP 文件后并在提取内容之前，您可能需要将其解除锁定。通常，通过打开文件的属性，查看常规选项卡，并选择解除锁定复选框（如果存在）即可完成这一操作。

如果 ZIP 文件需要解除锁定但您没有这样做，您可能会收到类似这样的错误：“Import-Module：无法加载文件或程序集”。

- 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`

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 AWSPowerShell1.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 AWS 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 AWS 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 AWSPowerShell1.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` 模块显式加载到 PowerShell 会话中。

```
PS > Import-Module AWSPowerShell1.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

您可以通过以下两种方式之一安装 AWS Tools for Windows PowerShell：

- 从 [AWSPowerShell.zip](#) 下载模块并将其提取到其中一个模块目录中。您可以通过显示 `PSModulePath` 变量的值来查找模块目录。

Warning

下载 ZIP 文件后并在提取内容之前，您可能需要将其解除锁定。通常，通过打开文件的属性，查看常规选项卡，并选择解除锁定复选框（如果存在）即可完成这一操作。如果 ZIP 文件需要解除锁定但您没有这样做，您可能会收到类似这样的错误：“Import-Module : 无法加载文件或程序集”。

- 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`

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 AWS 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 AWS 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...}

版本控制

AWS 定期发布 AWS Tools for PowerShell 的新版本，以支持新的 AWS 服务和特征。要确定您安装的工具版本，请运行 [Get-AWSPowerShellVersion](#) cmdlet。

例如：

```
PS > Get-AWSPowerShellVersion
```

```
AWS Tools for PowerShell
Version 4.1.849
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET
Core Runtime Version 3.7.402.75
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/
CHANGELOG.ALL.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 AWS 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。

例如：

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

Service                                Noun Prefix Module Name                                SDK
-----
Assembly
Version
-----
-----
AWS IAM Access Analyzer                IAMAA      AWS.Tools.AccessAnalyzer
3.7.400.33
AWS Account                            ACCT       AWS.Tools.Account
3.7.400.33
AWS Certificate Manager Private... PCA        AWS.Tools.ACMPCA
3.7.400.34
AWS Amplify                            AMP        AWS.Tools.Amplify
3.7.401.28
Amplify Backend                        AMPB       AWS.Tools.AmplifyBackend
3.7.400.33
...
```

要确定您运行的 PowerShell 版本，请输入 `$PSVersionTable` 以查看 `$PSVersionTable` [自动变量](#) 内容。

例如：

```
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 AWS Tools for PowerShell

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

更新模块化的 **AWS.Tools** 模块

要将 AWS.Tools 模块更新到最新版本，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 PSGallery 的 PSRepository (<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 AWS 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 AWS 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
```

- 如果您已使用下载的 ZIP 文件进行安装：
 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 AWS Tools for PowerShell 控制台。

AWS Tools for PowerShell 在 Linux 或 macOS 上安装

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

设置概述

要 AWS Tools for PowerShell 在 Linux 或 macOS 电脑上安装，你可以从两个软件包选项中进行选择：

- [AWS.Tools](#)— 的模块化版本。AWS Tools for PowerShell 每项 AWS 服务都由其自己的独立小型模块提供支持，并带有共享的支持模块 `AWS.Tools.Common`。
- [AWSPowerShell-NetCore](#)— 的单一大模块版本。AWS Tools for PowerShell 这个单一的大型模块支持所有 AWS 服务。

Note

请注意，单个模块可能太大，无法与 [AWS Lambda](#) 函数一起使用。请改用上面显示的模块化版本。

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 酷睿 6.0 或更高版本。
2. 安装 PowerShell Core 后，PowerShell 首先 pwsh 在系统外壳中运行。
3. 安装其中一个 AWS.Tools 或 AWSPower 命令行管理程序。NetCore。
4. 运行相应的 Import-Module cmdlet 将模块导入到您的 PowerShell 会话中。
5. 运行 “[初始化-AWSDefault 配置](#)” cmdlet 以提供您的 AWS 凭据。

先决条件

要运行 AWS 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 平台版本列表以及有关如何在基于 Linux 的计算机 PowerShell 上安装最新版本的信息，请参阅 Microsoft 网站 [PowerShell 上的在 Linux 上安装](#)。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 版本以及如何在 macOS 上安装最新版本的信息，请参阅 PowerShell 微软网站 [上的在 macOS 上安装](#)。

在 Linux 或 macOS 上安装 AWS.Tools

可以在运行 PowerShell Core 6.0 或更高版本的计算机上安装 AWS Tools for PowerShell 模块化版本。有关如何安装 PowerShell Core 的信息，请参阅 PowerShell 在 Microsoft PowerShell 网站上 [安装各种版本](#) 的。

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。该模块简化了其他 AWS.Tools 模块的安装和更新。AWS.Tools.Installer 需要 PowerShellGet，并自动下载和安装其最新版本。AWS.Tools.Installer 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。

该方法在以下过程中描述。

- 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 库中安装每个服务模块。

使用 **AWS.Tools.Installer** 模块在 Linux 或 macOS 上安装 **AWS.Tools** 的步骤

1. 通过运行以下命令启动 PowerShell 核心会话。

```
$ pwsh
```

Note

我们建议您不要以具有更高权限的管理员 PowerShell 身份运行，除非手头的任务需要这样做。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.Installer` 模块安装模块化的 `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** PowerShell 以允许安装模块。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 Amazon EC2 和 Amazon S3 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 AWS 服务模块都需要的共享模块。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 PSRepository 命名 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 PSRepository 的更多信息，请使用命令 `Get-PSRepository -Name PSGallery`。

前面的命令会将模块安装到系统上的默认目录中。实际目录取决于您的操作系统的发行版和版本以及所 PowerShell 安装的版本。例如，如果您在类似 RHEL 的系统上安装了 PowerShell 7，则默认模块很可能位于 `/opt/microsoft/powershell/7/Modules` (或 `$PSHOME/Modules`) 中，而用户模块很可能位于 `~/.local/share/powershell/Modules`。有关更多信息，请参阅微软 PowerShell 网站 [PowerShell 上的在 Linux 上安装](#)。要查看模块安装位置，请运行以下命令：

```
PS > Get-Module -ListAvailable
```

要安装其他模块，请使用相应的模块名称运行类似的命令，如图 [PowerShell 库中所示](#)。

安装 AWSPowerShell。NetCore 在 Linux 或 macOS 上

升级到较新版本的 AWSPowerShell。NetCore，请按照中的说明进行操作[AWS Tools for PowerShell 在 Linux 或 macOS 上更新](#)。卸载早期版本的 AWSPowerShell 命令行管理程序。NetCore 第一。

您可以安装 AWSPowerShell 命令行管理程序。NetCore 用以下两种方式之一：

- 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。
- 按照以下步骤所述，使用 `Install-Module` cmdlet 从 PowerShell 库中安装。

安装 AWSPowerShell 命令行管理程序。NetCore 在 Linux 或 macOS 上使用安装模块 cmdlet

通过运行以下命令启动 PowerShell 核心会话。

```
$ pwsh
```

Note

我们建议您不要 `sudo pwsh` 以提升的管理员权限开始 PowerShell 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。

安装 AWSPowerShell 命令行管理程序。NetCore PowerShell 图库中的单模块包，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` PowerShell 以允许安装模块。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
```

除非您想为计算机的所有用户安装，否则不必以 root 用户身份运行此命令。AWS Tools for PowerShell 为此，请在已启动的 PowerShell 会话中运行以下命令 `sudo pwsh`。

```
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。你可以运行 `Get-ExecutionPolicy`，这表明在非 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 AWS 模块目录的位置，因此 `Get-Module -ListAvailable` cmdlet 会显示您安装的模块。

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	

将 PowerShell 控制台配置为使用 AWS Tools for PowerShell Core (AWSPowerShell.NetCore 只有)

PowerShell 每当您在模块中运行 cmdlet 时，Core 通常都会自动加载模块。但这对 AWSPowerShell 不起作用。NetCore 因为它的体积很大。开始运行 AWSPowerShell 命令行管理程序。NetCore cmdlet，你必须先运行该命令。Import-Module AWSPowerShell.NetCoreAWS.Tools 模块中的 cmdlet 不需要此操作。

初始化您的 PowerShell 会话

安装完之后，在基于 Linux 或 macOS 的系统 PowerShell 上启动时 AWS Tools for PowerShell，必须运行“[初始化-AWSDefault 配置](#)”来指定要使用的 AWS 访问密钥。有关 Initialize-AWSDefaultConfiguration 的更多信息，请参阅[使用 AWS 凭证](#)。

Note

在的早期 (3.3.96.0 之前) 版本中，此 cmdlet 被 AWS Tools for PowerShell 命名为。Initialize-AWSDefaults

版本控制

AWS AWS Tools for PowerShell 定期发布新版本以支持新的 AWS 服务和功能。要确定已安装的版本 AWS Tools for PowerShell，请运行 [Get-AWSPowerShellVersion](#) cmdlet。

例如：

```
PS > Get-AWSPowerShellVersion
```

```
AWS Tools for PowerShell
```

```
Version 4.1.849
```

```
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET
```

```
Core Runtime Version 3.7.402.75
```

```
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/  
CHANGELOG.ALL.md
```


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

要查看当前版本工具中支持的 AWS 服务列表，请将 `-ListServiceVersionInfo` 参数添加到 [Get-AWSPowerShellVersion](#) cmdlet 中。

要确定您正在运行 PowerShell 的版本，请输入 `$PSVersionTable` 以查看 `$PSVersionTable` [自动变量](#) 的内容。

例如：

```
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
```

AWS Tools for PowerShell 在 Linux 或 macOS 上更新

随着更新版本的 AWS Tools for PowerShell 发布，您应该定期更新在本地运行的版本。

更新模块化的 **AWS.Tools** 模块

要将 AWS.Tools 模块更新到最新版本，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-AWSToolsModulecmdlet 从 PSRepository 命名的 PSGallery (<https://www.powershellgallery.com/>) 下载所有模块，并将其视为可信来源。有关此 PSRepository 的更多信息，请使用命令 `Get-PSRepository -Name PSGallery`。

更新 PowerShell 核心工具

运行 `Get-AWSPowerShellVersion` cmdlet 以确定你正在运行的版本，并将其与 [PowerShell 图库](#) 网站上提供的适用于 Windows PowerShell 的工具版本进行比较。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 AWS 服务的版本后，才会提供对新命令和服务的支持。

在安装更新版本的 AWSPowerShell 之前，NetCore，卸载现有模块。在卸载现有软件包之前，请关闭所有打开的 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
```

安装完成后，运行命令将更新后的 cmdlet 加载 `Import-Module AWSPowerShell.NetCore` 到您的 PowerShell 会话中。

相关信息

- [开始使用 AWS Tools for Windows PowerShell](#)
- [使用中的 AWS 服务 AWS Tools for PowerShell](#)

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

AWS Tools for PowerShell 版本 4 是 3.3 版本的向后兼容更新。AWS Tools for PowerShell 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 而且 AWSPowerShell 软件包是“整体的”。这意味着所有 AWS 服务都支持在同一个模块中，这使得它变得非常大，并且随着每项新 AWS 服务和功能的添加而变得越来越大。新 AWS.Tools 软件包被分成较小的模块，使您可以灵活地仅下载和安装所用 AWS 服务所需的模块。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，您仍必须致电 Import-Module 以 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

为了帮助您发现模块 AWS.Tools 集合中每项 AWS 服务的模块名称，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 都使用 API 为您调用 AWS 服务。适用于 .NET 的 AWS SDK 然后，AWS Tools for PowerShell 客户端将响应转换为可以在 PowerShell 脚本中使用的对象，并通过管道传输到其他命令。有时，最终 PowerShell 对象在原始响应中包含的字段或属性比您需要的要多，而其他时候，您可能希望该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 软件开发工具包操作。[ListObjects](#) 该操作返回一个 [ListObjectsResponse](#) 对象。但是，默认情况下，Get-S3Object cmdlet 仅向用户返回 SDK 响应 S3Objects 中的元素。PowerShell 在下面的示例中，该对象是包含两个元素的数组。

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket
```

```
ETag : "01234567890123456789012345678901111"
BucketName : amzn-s3-demo-bucket
Key : file1.txt
LastModified : 9/30/2019 1:31:40 PM
Owner : Amazon.S3.Model.Owner
Size : 568
StorageClass : STANDARD
```

```
ETag : "01234567890123456789012345678902222"
BucketName : amzn-s3-demo-bucket
Key : file2.txt
LastModified : 7/15/2019 9:36:54 AM
Owner : Amazon.S3.Model.Owner
Size : 392
StorageClass : STANDARD
```

在 AWS Tools for PowerShell 版本 4 中，您可以指定 -Select * 返回由 SDK API 调用返回的完整 .NET 响应对象。

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select *
IsTruncated      : False
NextMarker       :
S3Objects        : {file1.txt, file2.txt}
Name             : amzn-s3-demo-bucket
```

```
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 S3objects 数组中每个元素的 Key 属性。

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -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 amzn-s3-demo-bucket -Select S3objects.Key |  
>> Write-S3ObjectTagSet -Select ^Key -BucketName amzn-s3-demo-bucket -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

的早期版本 AWS Tools for PowerShell 允许您使用 -MaxItems 参数来指定最终输出中返回的最大对象数。

此行为已从 AWS.Tools 中删除。

AWSPower 命令行管理程序中已不推荐使用此行为。NetCore 和 AWSPower Shell，并将在未来的版本中从这些版本中删除。

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输到 Select-Object 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-S3ObjectV2 -BucketName amzn-s3-demo-bucket -Select S3objects.Key | select -  
first 2
```

```
file1.txt  
file2.txt
```

并非所有 AWS 服务都以相同 `-MaxItems` 的方式支持，因此这消除了这种不一致性以及有时会出现的意外结果。此外，`-MaxItems` 与新的 [-Select](#) 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"', '}' )
```

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

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='amzn-s3-demo-bucket'; Key='file1.txt';  
    PartNumber=1 } | Get-S3ObjectMetadata
```

Note

某些属性在早期版本中支持此功能 AWS Tools for PowerShell。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

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

在的早期版本中 AWS Tools for PowerShell，一些常用参数 (例如 AccessKey、SecretKeyProfileNameRegion、或) 是 [动态](#) 的，而所有其他参数都是静态的。这可能会造成问题，因为在动态参数之前 PowerShell 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
```

的早期版本将值 PowerShell 绑定 us-west-2 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 AWS 服务声明需要某个 API 的参数时，如果您没有指定相应的 cmdlet 参数，则 PowerShell 会提示您输入相应的 cmdlet 参数。这仅适用于 AWS.Tools。为确保向后兼容，这不适用于 AWSPower Shell。NetCore 或者 AWSPower 壳牌。

所有参数均可 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，因为该 AWS 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

以下功能在以前的版本中已被弃用，AWS Tools for PowerShell 在版本 4 中已删除：

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

开始使用 AWS Tools for Windows PowerShell

本节中的一些主题介绍了在[安装 Tools for Windows PowerShell](#) 后使用此类工具的基础知识。例如，这些主题说明了如何在 Tools for Windows PowerShell 与 AWS 交互时指定它应使用的[凭证](#)和[AWS 区域](#)。

本节中的其它主题提供了有关配置这些工具的高级方法、您的环境和项目的信息。

主题

- [使用 AWS 配置工具身份验证](#)
- [指定 AWS 区域](#)
- [使用配置联合身份 AWS Tools for PowerShell](#)
- [Cmdlet 发现和别名](#)
- [AWS Tools for PowerShell 中的管道、输出和迭代](#)
- [凭证和配置文件解析](#)
- [有关用户和角色的其它信息](#)
- [使用旧版凭证](#)

使用 AWS 配置工具身份验证

使用 AWS 服务进行开发时，您必须确定您的代码是如何使用 AWS 进行身份验证的。您可以通过不同的方式对 AWS 资源的编程访问权限进行配置，具体取决于环境和您可用的 AWS 访问权限。

要查看适用于 PowerShell 的工具的各种身份验证方法，请参阅《AWS 开发工具包和工具参考指南》中的[身份验证和访问](#)。

本主题假设新用户正在本地进行开发工作，雇主未向其提供身份验证方法，新用户将使用 AWS IAM Identity Center 来获取临时凭证。如果您的环境与这些假设不符，则本主题中的某些信息可能不适用于您，或者某些信息可能已经提供给您。

配置此环境需要几个步骤，总结如下：

1. [启用和配置 IAM Identity Center](#)
2. [将适用于 PowerShell 的工具配置为使用 IAM Identity Center。](#)
3. [开始 AWS 访问门户会话](#)

启用和配置 IAM Identity Center

要使用 AWS IAM Identity Center，必须先启用和配置它。要详细了解如何为 PowerShell 执行此操作，请查看《AWS 开发工具包和工具参考指南》中 [IAM Identity Center 身份验证](#) 主题内的步骤 1。具体而言，请按照我没有通过 IAM Identity Center 确立访问权限下的所有必要说明进行操作。

将适用于 PowerShell 的工具配置为使用 IAM Identity Center。

Note

从 Tools for PowerShell 的 4.1.538 版本开始，配置 SSO 凭证并启动 AWS 访问门户会话的推荐方法是使用 [Initialize-AWSSSOConfiguration](#) 和 [Invoke-AWSSSOLogin](#) cmdlet，如本主题所述。如果您无法访问该版本（或更高版本）的 Tools for PowerShell 或无法使用这些 cmdlet，您仍然可以通过使用 AWS CLI 来执行这些任务。要了解操作方法，请参阅 [使用 AWS CLI 进行门户登录](#)。

以下过程使用 SSO 信息更新共享的 AWS config 文件，Tools for PowerShell 使用该信息获取临时凭证。此过程的结果是，同时会启动一个 AWS 访问门户会话。如果共享的 config 文件已包含 SSO 信息，并且您只想了解如何使用 Tools for PowerShell 启动访问门户会话，请参阅本主题的下一部分 [开始 AWS 访问门户会话](#)。

1. 如果尚未完成，请打开 PowerShell 并根据您的操作系统和环境安装 AWS Tools for PowerShell，包括通用 cmdlet。有关如何执行此操作的信息，请参阅 [安装 AWS Tools for PowerShell](#)。

例如，如果在 Windows 上安装模块化版本的 Tools for PowerShell，您很可能会运行类似于以下内容的命令：

```
Install-Module -Name AWS.Tools.Installer
Install-AWSToolsModule AWS.Tools.Common
```

2. 运行以下命令。将示例属性值替换为来自您的 IAM Identity Center 配置的值。有关这些属性及其查找方法的信息，请参阅《AWS SDK 和工具参考指南》中的 [IAM Identity Center 凭证提供程序设置](#)。

```
$params = @{
    ProfileName = 'my-sso-profile'
    AccountId = '111122223333'
    RoleName = 'SamplePermissionSet'
```

```
SessionName = 'my-sso-session'  
StartUrl = 'https://provided-domain.awsapps.com/start'  
SSORegion = 'us-west-2'  
RegistrationScopes = 'sso:account:access'  
};  
Initialize-AWSSSOConfiguration @params
```

或者，您可以单独使用 cmdlet `Initialize-AWSSSOConfiguration`，Tools for PowerShell 将提示您输入属性值。

某些属性值的注意事项：

- 如果您仅按照[启用并配置 IAM Identity Center](#) 的说明操作，则 `-RoleName` 的值可能为 `PowerUserAccess`。但如果您专门为 PowerShell 工作创建了 IAM Identity Center 权限集，请改用该权限集。
 - 请确保使用您已配置 IAM Identity Center 的 AWS 区域。
3. 此时，共享的 AWS config 文件包含一个名为 `my-sso-profile` 的配置文件，其中包含一组可以从 Tools for PowerShell 引用的配置值。要查找此文件的位置，请参阅《AWS 开发工具包和工具参考指南》中的[共享文件的位置](#)。

适用于 PowerShell 的工具在向 AWS 发送请求之前，使用配置文件的 SSO 令牌提供程序获取凭证。`sso_role_name` 值是与 IAM Identity Center 权限集关联的 IAM 角色，应允许访问您的应用程序中使用的 AWS 服务。

以下示例显示了使用上述命令创建的配置文件。您实际配置文件中的某些属性值及其顺序可能有所不同。配置文件的 `sso-session` 属性引用名为 `my-sso-session` 的部分，该部分包含用于启动 AWS 访问门户会话的设置。

```
[profile my-sso-profile]  
sso_account_id=111122223333  
sso_role_name=SamplePermissionSet  
sso_session=my-sso-session  
  
[sso-session my-sso-session]  
sso_region=us-west-2  
sso_registration_scopes=sso:account:access  
sso_start_url=https://provided-domain.awsapps.com/start/
```

4. 如果您已有活动的 AWS 访问门户会话，Tools for PowerShell 会通知您已登录。

如果情况并非如此，Tools for PowerShell 会尝试在您的默认 Web 浏览器中自动打开 SSO 授权页面。按照浏览器中的提示操作，其中可能包括 SSO 授权码、用户名和密码以及访问 AWS IAM Identity Center 账户和权限集的权限。

Tools for PowerShell 会通知您 SSO 登录成功。

开始 AWS 访问门户会话

在运行访问 AWS 服务的命令之前，您需要有一个活动的 AWS 访问门户会话，以便 Tools for PowerShell 可以使用 IAM Identity Center 身份验证来解析凭证。要登录到 AWS 访问门户，请在 PowerShell 中运行以下命令，其中 `-ProfileName my-sso-profile` 是您按照本主题前一节中的过程操作时在共享 config 文件中创建的配置文件名称。

```
Invoke-AWSSSOLogin -ProfileName my-sso-profile
```

如果您已有活动的 AWS 访问门户会话，Tools for PowerShell 会通知您已登录。

如果情况并非如此，Tools for PowerShell 会尝试在您的默认 Web 浏览器中自动打开 SSO 授权页面。按照浏览器中的提示操作，其中可能包括 SSO 授权码、用户名和密码以及访问 AWS IAM Identity Center 账户和权限集的权限。

Tools for PowerShell 会通知您 SSO 登录成功。

要测试您是否已拥有活动会话，请在安装或导入 `AWS.Tools.SecurityToken` 模块后运行以下命令。

```
Get-STSCallerIdentity -ProfileName my-sso-profile
```

对 `Get-STSCallerIdentity` cmdlet 的响应报告了在共享 config 文件中配置的 IAM Identity Center 账户和权限集。

示例

以下示例说明如何将 IAM Identity Center 与适用于 PowerShell 的工具一起使用。该示例假定以下内容：

- 您已启用 IAM Identity Center，并按照本主题前面所述对其进行了配置。SSO 属性位于 `my-sso-profile` 配置文件中，该文件在本主题前面部分已配置。

- 当您通过 `Initialize-AWSSSOConfiguration` 或 `Invoke-AWSSSOLogin` cmdlet 登录时，用户至少拥有对 Amazon S3 的只读权限。
- 某些 S3 存储桶可供该用户查看。

根据需要安装或导入 `AWS.Tools.S3` 模块，然后使用以下 PowerShell 命令显示 S3 存储桶列表。

```
Get-S3Bucket -ProfileName my-sso-profile
```

其他信息

- 有关适用于 PowerShell 的工具身份验证的更多选项，例如配置文件和环境变量的使用，请参阅《AWS 开发工具包和工具参考指南》中的[配置](#)章节。
- 某些命令需要指定 AWS 区域。有多种方法可以实现，包括 `-Region` cmdlet 选项、`[default]` 配置文件和 `AWS_REGION` 环境变量。有关更多信息，请参阅本指南中的[指定 AWS 区域](#)以及《AWS SDK 和工具参考指南》中的[AWS 区域](#)。
- 有关最佳实践的更多信息，请参阅《IAM 用户指南》中的[IAM 中的安全最佳实践](#)。
- 要创建短期 AWS 凭证，请参阅《IAM 用户指南》中的[临时安全凭证](#)。
- 要了解其他凭证提供程序，请参阅《AWS 开发工具包和工具参考指南》中的[标准化凭证提供程序](#)。

主题

- [使用 AWS CLI 进行门户登录](#)

使用 AWS CLI 进行门户登录

从 Tools for PowerShell 的 4.1.538 版本开始，配置 SSO 凭证并启动 AWS 访问门户会话的推荐方法是使用 [Initialize-AWSSSOConfiguration](#) 和 [Invoke-AWSSSOLogin](#) cmdlet，如 [使用 AWS 配置工具身份验证](#) 中所述。如果您无法访问该版本（或更高版本）的 Tools for PowerShell 或无法使用这些 cmdlet，您仍然可以通过使用 AWS CLI 来执行这些任务。

通过 AWS CLI 配置 Tools for PowerShell 以使用 IAM Identity Center。

如果尚未完成，请确保在继续操作之前[启用并配置 IAM Identity Center](#)。

关于如何通过 AWS CLI 配置 Tools for PowerShell 以使用 IAM Identity Center 的信息位于《AWS SDK 和工具参考指南》中 [IAM Identity Center 身份验证](#) 主题的步骤 2 中。完成此配置后，您的系统应包含以下元素：

- AWS CLI，您可用于在运行应用程序之前启动 AWS 访问门户会话。
- 该共享 AWS config 文件包含一个 [\[default\] 配置文件](#)，在该配置文件中包含一组配置值，可以从适用于 PowerShell 的工具中引用这些配置值。要查找此文件的位置，请参阅《AWS 开发工具包和工具参考指南》中的[共享文件的位置](#)。适用于 PowerShell 的工具在向 AWS 发送请求之前，使用配置文件的 SSO 令牌提供程序获取凭证。sso_role_name 值是与 IAM Identity Center 权限集关联的 IAM 角色，应允许访问您的应用程序中使用的 AWS 服务。

以下示例 config 文件显示了使用 SSO 令牌提供程序设置的 [default] 配置文件。配置文件的 sso_session 设置是指所指定的 sso-session 节。sso-session 节包含启动 AWS 访问门户会话的设置。

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

Important

您的 PowerShell 会话必须安装并导入以下模块，这样 SSO 解析才能生效：

- AWS.Tools.SSO
- AWS.Tools.SSOOIDC

如果您使用的是旧版本的 Tools for PowerShell 且没有这些模块，您将收到类似于以下的错误：“找不到程序集 AWSSDK.SSOOIDC...”。

开始 AWS 访问门户会话

在运行访问 AWS 服务的命令之前，您需要有一个活动的 AWS 访问门户会话，以便 Windows PowerShell 工具可以使用 IAM Identity Center 身份验证来解析凭证。根据您的配置的会话时长，您的访

问权限最终将过期，并且适用于 Windows PowerShell 的工具将遇到身份验证错误。要登录 AWS 访问门户，请在 AWS CLI 中运行以下命令。

```
aws sso login
```

由于您正在使用 [default] 配置文件，因此无需使用 --profile 选项调用命令。如果您的 SSO 令牌提供程序配置使用命名配置文件，则该命令为 `aws sso login --profile named-profile`。有关命名配置文件的更多信息，请参阅《AWS SDK 和工具参考指南》中的[配置文件](#)部分。

要测试您是否已拥有活动会话，请运行以下 AWS CLI 命令（命名配置文件的注意事项相同）：

```
aws sts get-caller-identity
```

对此命令的响应应该报告共享 config 文件中配置的 IAM Identity Center 账户和权限集。

Note

如果您已经有一个有效的 AWS 访问门户会话并且运行了 `aws sso login`，则无需提供凭证。

登录过程可能会提示您允许 AWS CLI 访问您的数据。由于 AWS CLI 基于适用于 Python 的 SDK 而构建，因此权限消息可能包含 `botocore` 名称的变体。

示例

以下示例说明如何将 IAM Identity Center 与适用于 PowerShell 的工具一起使用。该示例假定以下内容：

- 您已启用 IAM Identity Center，并按照本主题前面所述对其进行了配置。SSO 属性位于 [default] 配置文件中。
- 当您使用 `aws sso login` 通过 AWS CLI 登录时，该用户至少具有 Amazon S3 的只读权限。
- 某些 S3 存储桶可供该用户查看。

使用以下 PowerShell 命令显示 S3 存储桶的列表：

```
Install-Module AWS.Tools.Installer  
Install-AWSToolsModule S3
```



```
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SS0, SS00IDC

# In older versions of the AWS Tools for PowerShell, we're not invoking a cmdlet from
these modules directly,
# so we must import them explicitly:
Import-Module AWS.Tools.SS0
Import-Module AWS.Tools.SS00IDC

# Older versions of the AWS Tools for PowerShell don't support the SS0 login flow, so
login with the CLI
aws sso login

# Now we can invoke cmdlets using the SS0 profile
Get-S3Bucket
```

如上所述，由于您正在使用 [default] 配置文件，因此无需使用 -ProfileName 选项调用 Get-S3Bucket cmdlet。如果您的 SSO 令牌提供程序配置在使用指定的配置文件，则命令为 Get-S3Bucket -ProfileName *named-profile*。有关命名配置文件的更多信息，请参阅《AWS SDK 和工具参考指南》中的[配置文件](#)部分。

其他信息

- 有关适用于 PowerShell 的工具身份验证的更多选项，例如配置文件和环境变量的使用，请参阅《AWS 开发工具包和工具参考指南》中的[配置](#)章节。
- 某些命令需要指定 AWS 区域。有多种方法可以实现，包括 -Region cmdlet 选项、[default] 配置文件和 AWS_REGION 环境变量。有关更多信息，请参阅本指南中的[指定 AWS 区域](#)以及《AWS SDK 和工具参考指南》中的[AWS 区域](#)。
- 有关最佳实践的更多信息，请参阅《IAM 用户指南》中的[IAM 中的安全最佳实践](#)。
- 要创建短期 AWS 凭证，请参阅《IAM 用户指南》中的[临时安全凭证](#)。
- 要了解其他凭证提供程序，请参阅《AWS 开发工具包和工具参考指南》中的[标准化凭证提供程序](#)。

指定 AWS 区域

有两种方法可以指定运行 AWS Tools for PowerShell 命令时要使用的 AWS 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 Windows 工具 PowerShell 无法确定要使用哪个区域，许多 AWS cmdlet 就会失败。例外情况包括适用于 [Amazon S3](#)、[Amazon S](#) ES 和的 cmdlet AWS Identity and Access Management，它们会自动默认为全局终端节点。

为单个 AWS 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 AWS CLI 命令设置默认区域

在 PowerShell 命令提示符下，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。要将该设置应用于您的所有 PowerShell 会话，请将此命令添加到您的 PowerShell 配置文件中，就像在 `Import-Module` 命令中所做的那样。

查看所有 AWS CLI 命令的当前默认区域

在 PowerShell 命令提示符下，键入以下命令。

```
PS > Get-DefaultAWSRegion
```

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

清除所有 AWS CLI 命令的当前默认区域

在 PowerShell 命令提示符下，键入以下命令。

```
PS > Clear-DefaultAWSRegion
```

查看所有可用 AWS 区域的列表

在 PowerShell 命令提示符下，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

按照以下示例格式，将 -EndpointUrl 通用参数添加到 Windows 工具 PowerShell 命令中，将自定义终端节点指定为 URL。

```
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 AWS 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"
```

附加信息

有关 AWS 区域的更多信息，请参阅 AWS SDKs 和工具参考指南中的 [AWS 区域](#)。

使用配置联合身份 AWS Tools for PowerShell

要允许组织中的用户访问 AWS 资源，您必须配置标准且可重复的身份验证方法，以实现安全性、可审计性、合规性以及支持角色和帐户分离的能力。尽管通常为用户提供访问权限 AWS APIs，但如果没有联合 API 访问权限，您还必须创建 AWS Identity and Access Management (IAM) 用户，这违背了使用联合身份验证的目的。本主题介绍中对简化联合访问解决方案的 SAML (安全断言标记语言) 支持。

AWS Tools for PowerShell

中的 SAML 支持 AWS Tools for PowerShell 允许您为用户提供对 AWS 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，特别是在身份提供商 (例如 [Active Directory 联合身份验证服务](#)) 和服务提供商 (例如) 之间传输用户身份验证和授权数据。AWS 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](#)。中的 SAML 支持与 S AWS Tools for PowerShell AML 2.0 兼容。

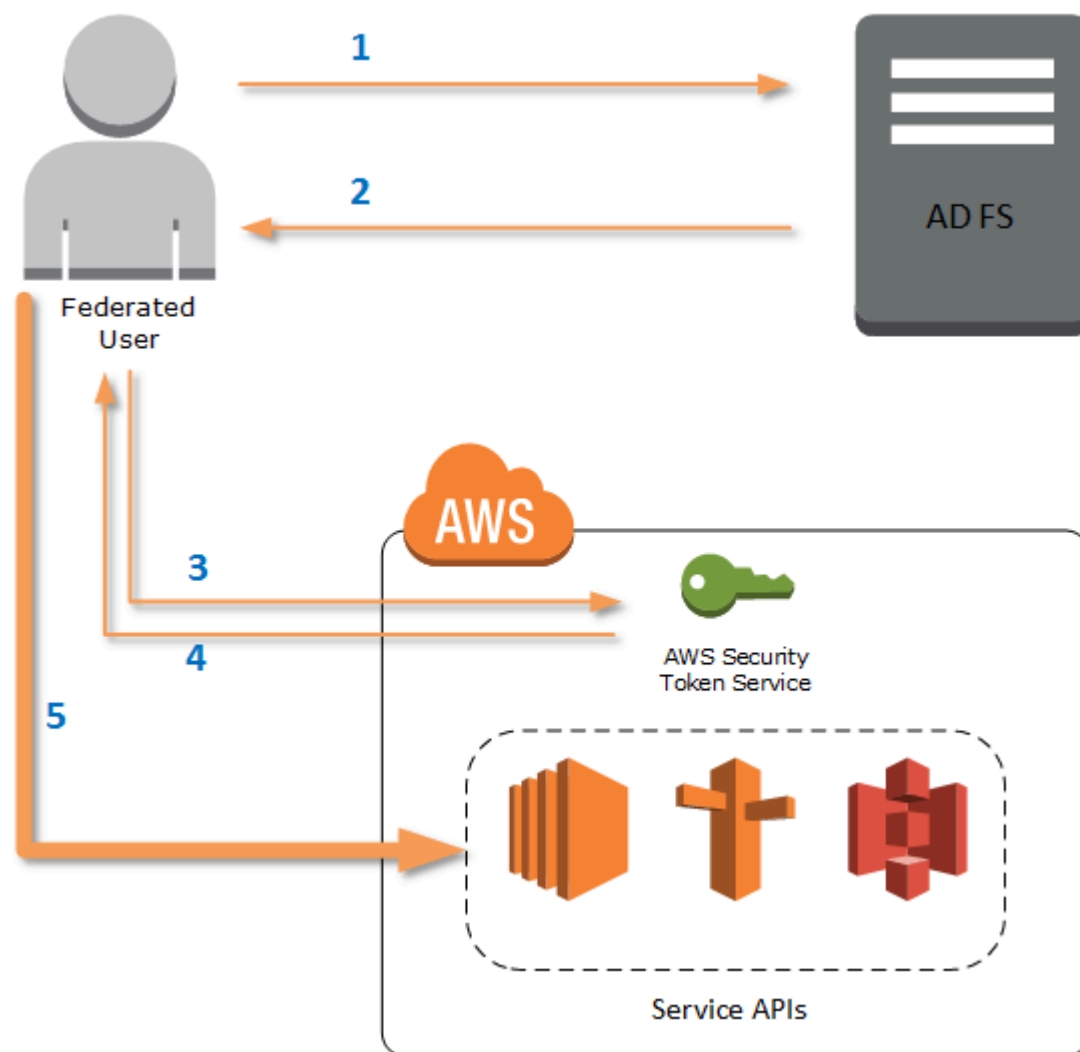
先决条件

首次尝试使用 SAML 支持前，您必须做好以下准备。

- 与您的 AWS 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。有关如何专门针对 Active Directory 联合身份验证服务执行此操作的更多信息，请参阅 IAM 用户指南中的[关于 SAML 2.0 联合](#)，以及博客文章《[为 AWS 使用 Windows Active Directory、AD FS 和 SAML 2.0 启用联合](#)》。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。
- AWS Tools for PowerShell 安装在本地工作站上的版本 3.1.31.0 或更高版本。

联合身份用户如何获得对服务的联合访问权限 AWS APIs

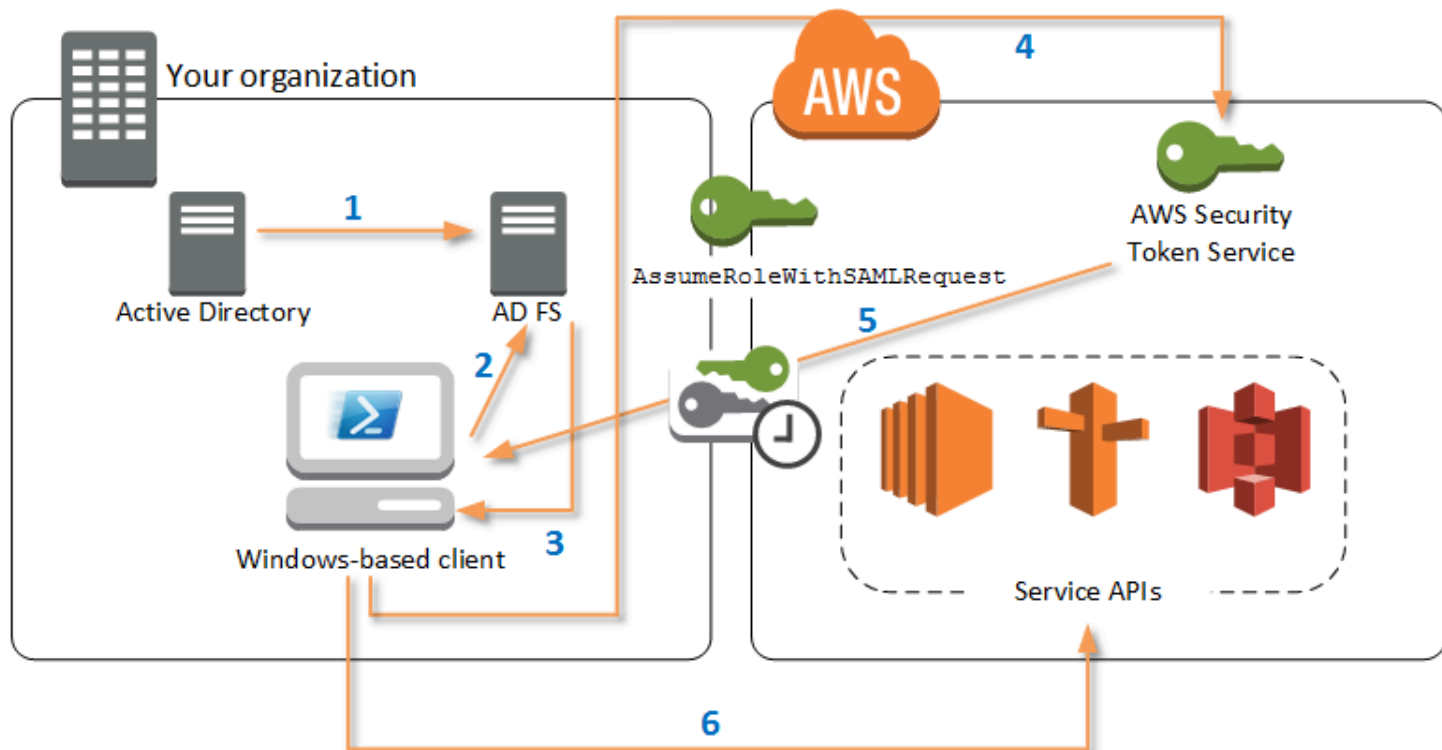
以下过程简要介绍了 AD FS 如何联合活动目录 (AD) 用户以获得对 AWS 资源的访问权限。



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给 AWS Security Token Service (STS)。
4. STS 返回一个 SAML 响应，其中包含用户可以扮演的角色的 AWS 临时证书。
5. 用户通过在发 APIs 出的请求中包含这些临时证书来 AWS Tools for PowerShell 访问 AWS 服务。

SAML Support 在 AWS Tools for PowerShell

本节介绍 AWS Tools for PowerShell cmdlet 如何为用户启用基于 SAML 的身份联合配置。



1. AWS Tools for PowerShell 使用 Windows 用户的当前凭据对照 AD FS 进行身份验证，或者在用户尝试运行需要凭据才能调用的 cmdlet 时以交互方式进行身份验证。AWS
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；断言的目的是识别和提供有关用户的信息。AWS Tools for PowerShell 从 SAML 断言中提取用户的授权角色列表。
4. AWS Tools for PowerShell 通过调用 API 将 SAML 请求（包括所请求角色的亚马逊资源名称 (ARN)）转发给 STS。AssumeRoleWithSAMLRequest
5. 如果 SAML 请求有效，STS 会返回包含 AWS AccessKeyId、SecretAccessKey 和 SessionToken 的响应。这些凭证的有效期为 3,600 秒（1 小时）。
6. 现在，用户拥有有效的凭证，可以使用 APIs 该用户角色有权访问的任何 AWS 服务。AWS Tools for PowerShell 会自动将这些证书应用于任何后续 AWS 的 API 调用，并在它们过期时自动续订。

Note

如果凭证过期且需要新凭证，AWS 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，AWS Tools for PowerShell 会提示用户输入凭证，然后才能重新进行身份验证。

如何使用 PowerShell SAML 配置 Cmdlet

AWS 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 AWS 凭据配置文件一样，您可以为角色配置文件指定一个友好的名称。您可以在 **Set-AWSCredential** cmdlet 中使用相同的友好名称，也可以将其用作调用服务的任何 cmdlet 的 **-ProfileName** 参数值。AWS APIs

打开一个新 AWS Tools for PowerShell 会话。如果您运行的是 PowerShell 3.0 或更高版本，则在运行其任何 cmdlet 时都会自动导入该 AWS Tools for PowerShell 模块。如果您运行的是 PowerShell 2.0，则必须通过运行 `Import-Module` cmdlet 来手动导入模块，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。请务必将占位符帐户 IDs 和 AD FS 主机名替换为您自己的帐户 IDs 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。

- 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 non-domain-joined 帐户，您可以提供 Active Directory 凭据，然后选择用户有权访问的 AWS 角色，如下行所示。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
```

- 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凭证的 Cmdlet AWS

要运行需要 AWS 凭据的 cmdlet，您可以使用 AWS 共享凭据文件中定义的角色配置文件。将角色配置文件的名称提供给 `Set-AWSCredential`（或作为中任何 `ProfileName` 参数的值 AWS Tools for PowerShell），以自动获取配置文件中描述的角色临时 AWS 证书。

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 AWS 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` 配置文件获得的临时 AWS 证书来使用 AWS 服务 APIs。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

此示例使用为 AWS Tools for PowerShell 会话设置默认角色 `Set-AWSCredential`。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

此示例列出了与SAMLDemoProfile个人资料关联的角色 AWS 账户中所有可用的 Amazon S3 存储桶。该示例显示，尽管您可能在 AWS Tools for PowerShell 会话的早期使用过其他配置文件，但您可以通过使用支持该配置文件的 cmdlet 为 -ProfileName 参数指定不同的值来更改配置文件。对于通过 PowerShell 命令行管理 Amazon S3 的管理员来说，这是一项常见的任务。

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile
```

CreationDate	BucketName
-----	-----
7/25/2013 3:16:56 AM	<i>amzn-s3-demo-bucket</i>
4/15/2015 12:46:50 AM	<i>amzn-s3-demo-bucket1</i>
4/15/2015 6:15:53 AM	<i>amzn-s3-demo-bucket2</i>
1/12/2015 11:20:16 PM	<i>amzn-s3-demo-bucket3</i>

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 AWS 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 non-domain-joined 用户帐户，AWS Tools for PowerShell 会显示要求输入用户密码的 PowerShell 凭据提示。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

有关如何实现联合 API 访问的一般信息，请参阅[如何使用 SAML 2.0 实现联合 API/CLI 访问的通用解决方案](#)。

如有支持问题或意见，请访问[PowerShell 脚本](#) AWS 开发者论坛或 [.NET 开发论坛](#)。

Cmdlet 发现和别名

此部分介绍如何列出 AWS Tools for PowerShell 支持的服务，如何显示 AWS 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 AWS 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，AWS 服务 API 和 AWS PowerShell cmdlet 之间存在一对一的对应关系。要获取与 AWS 服务 API 名称对应的 cmdlet 名称，请使用 `-ApiOperation` 参数和 AWS 服务 API 名称运行 `Get-AWSCmdletName` cmdlet。例如，要获取基于任何可用的 `DescribeInstances` AWS 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 和服务的名称，则可以将 `-Service` 参数以及 cmdlet 名词前缀包含为 AWS 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AWS服务 API 或AWS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-EC2SecurityGroupToClientVpnTargetNetwork			
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AWS服务的名称，但不知道AWS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	ServiceName	ServiceOperation	CmdletNounPrefix
-----	-----	-----	-----
Add-EC2SecurityGroupToClientVpnTargetNetwork	Amazon Elastic Compute Cloud EC2	ApplySecurityGroupsToClientVpnTargetNetwork	
Get-EC2SecurityGroup	Amazon Elastic Compute Cloud EC2	DescribeSecurityGroups	
Get-EC2SecurityGroupReference	Amazon Elastic Compute Cloud EC2	DescribeSecurityGroupReferences	
Get-EC2StaleSecurityGroup	Amazon Elastic Compute Cloud EC2	DescribeStaleSecurityGroups	
Grant-EC2SecurityGroupEgress	Amazon Elastic Compute Cloud EC2	AuthorizeSecurityGroupEgress	
Grant-EC2SecurityGroupIngress	Amazon Elastic Compute Cloud EC2	AuthorizeSecurityGroupIngress	
New-EC2SecurityGroup	Amazon Elastic Compute Cloud EC2	CreateSecurityGroup	
Remove-EC2SecurityGroup	Amazon Elastic Compute Cloud EC2	DeleteSecurityGroup	
Revoke-EC2SecurityGroupEgress	Amazon Elastic Compute Cloud EC2	RevokeSecurityGroupEgress	
Revoke-EC2SecurityGroupIngress	Amazon Elastic Compute Cloud EC2	RevokeSecurityGroupIngress	
Update-EC2SecurityGroupRuleEgressDescription	Amazon Elastic Compute Cloud EC2	UpdateSecurityGroupRuleDescriptionsEgress	
Update-EC2SecurityGroupRuleIngressDescription	Amazon Elastic Compute Cloud EC2	UpdateSecurityGroupRuleDescriptionsIngress	

如果您知道 AWS Command Line Interface (AWS CLI) 命令的名称，可以使用 `-AwsCliCommand` 参数和所需的 AWS 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` AWS 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 AWS CLI 命令名称即可识别服务和 AWS 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 AWS 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 AWS 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

AWS Tools for PowerShell 中适用于每个服务的 cmdlet 基于该服务的 AWS 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AWS 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.txt
```

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

Note

AWS cmdlet 的 cmdlet 帮助提供了该 cmdlet 基于的AWS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AWS 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

AWS cmdlet 不支持 -WhatIf 开关。

别名

AWS Tools for PowerShell 安装程序会安装一个别名文件，其中包含很多AWS 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AWS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AWS 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 AWS Tools for PowerShell 别名，请运行以下命令。该命令在 PowerShell 的 `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 AWSPowerShell	Find-CTEvents	3.3.343.0
Alias AWSPowerShell	Get-ASACases	3.3.343.0
Alias AWSPowerShell	Get-ASAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-ASACommunications	3.3.343.0
Alias AWSPowerShell	Get-ASAServices	3.3.343.0
Alias AWSPowerShell	Get-ASASeverityLevels	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckRefreshStatuses	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorChecks	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckSummaries	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHooks	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHookTypes	3.3.343.0
Alias AWSPowerShell	Get-AWSCredentials	3.3.343.0
Alias AWSPowerShell	Get-CDApplications	3.3.343.0
Alias AWSPowerShell	Get-CDDeployments	3.3.343.0
Alias AWSPowerShell	Get-CFCloudFrontOriginAccessIdentities	3.3.343.0
Alias AWSPowerShell	Get-CFDistributions	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigRules	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigurationRecorders	3.3.343.0
Alias AWSPowerShell	Get-CFGDeliveryChannels	3.3.343.0
Alias AWSPowerShell	Get-CFInvalidations	3.3.343.0
Alias AWSPowerShell	Get-CFNAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-CFNStackEvents	3.3.343.0

...

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
```

AWS Tools for PowerShell 中的管道、输出和迭代

管道

PowerShell 鼓励用户将 cmdlet 连接到[管道](#)中，将一个 cmdlet 的输出定向到下一个 cmdlet 的输入。以下示例展示了在使用 AWS Tools for PowerShell 时的这种行为。该命令获取并停止当前默认区域中的所有 Amazon EC2 实例。

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

Cmdlet 输出

为了更好地支持管道，适用于 .NET 的 AWS SDK 响应中的一些数据可能默认被丢弃。来自 AWS Tools for PowerShell 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 AWS 服务调用中发送和接收的内容。从 AWS Tools for PowerShell V4 开始，cmdlet 可以使用 `-Select *` 参数和参数来返回整个服务响应。

Note

在 AWS Tools for PowerShell V4 之前的版本中，引入了一个名为 `$AWSHistory` 的会话变量，该变量维护了 AWS cmdlet 调用以及每个调用接收到的服务响应的记录。在 Tools for

PowerShell 的 V4 中，这个会话变量已被弃用，转而使用 `-Select *` 参数和参数，该参数可用于返回整个服务响应。此参数在本主题中描述。

 Note

这是适用于预览版中特征的预发行文档。本文档随时可能更改。

`$AWSHistory` 变量将在 AWS Tools for PowerShell 的 V5 版本中被移除。有关更多信息，请参阅博客文章 [Notice of upcoming major version 5 of AWS Tools for PowerShell](#)。

为了说明如何返回响应中的所有数据，请考虑以下示例。

第一个示例简单地返回一个 Amazon S3 存储桶列表。这是默认行为。

```
PS > Get-S3Bucket
```

CreationDate	BucketName
-----	-----
9/22/2023 10:54:35 PM	amzn-s3-demo-bucket1
9/22/2023 11:04:37 AM	amzn-s3-demo-bucket2
9/22/2023 12:54:34 PM	amzn-s3-demo-bucket3

第二个示例返回一个适用于 .NET 的 AWS SDK 响应对象。因为指定了 `-Select *`，输出包括整个 API 响应，其中包含 Buckets 属性中的存储桶集合。在此示例中，`Format-List` cmdlet 并非严格必需，但存在以确保显示所有属性。

```
PS > Get-S3Bucket -Select * | Format-List
```

```

LoggedAt      : 10/1/2023 9:45:52 AM
Buckets       : {amzn-s3-demo-bucket1, amzn-s3-demo-bucket2,
                amzn-s3-demo-bucket3}
Owner        : Amazon.S3.Model.Owner
ContinuationToken :
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength  : 0
HttpStatusCode : OK
  
```

通过分页数据进行迭代

以下部分描述了可能的各种迭代类型。

自动迭代

对于施加默认最大返回对象数量的服务 API 或支持可分页结果集的服务 API，大多数 cmdlet 实现自动迭代，这使得默认行为“翻页至完成”成为可能。在此场景中，cmdlet 代表您进行尽可能多的调用，以将完整数据集返回到管道。

在以下使用 `Get-S3Object` cmdlet 的示例中，`$result` 变量包含名为 `amzn-s3-demo-bucket1` 的存储桶中每个键的 `S3Object` 实例，这可能是一个非常大的数据集。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1
```

以下示例在自动迭代期间将每页的结果数量从默认值 1000 减少到 500。该示例执行两倍的自动迭代调用，因为每次调用返回的结果数量减半。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500
```

Note

在 AWS Tools for PowerShell V4 中，一些用于分页操作的 cmdlet 未实现自动迭代。如果一个 cmdlet 没有 `-NoAutoIteration` 参数（该参数在下一节讨论），那么它未实现自动迭代。

禁用自动迭代

如果您希望 Tools for PowerShell 仅返回第一页数据，您可以添加 `-NoAutoIteration` 参数以防止返回其他页的数据。

以下示例使用 `-NoAutoIteration` 和 `-MaxKey` 参数将返回的 `S3Object` 实例数量限制为在存储桶中找到的前 500 个。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -  
NoAutoIteration
```

要确定是否有更多可用数据但未返回，请使用 `-Select *` 参数和参数并检查 `next token` 属性中是否有值。

如果存储桶中的对象超过 500 个，以下示例返回 `$true`，否则返回 `$false`。

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -  
NoAutoIteration -Select *  
PS > $null -eq $result.NextMarker
```

Note

next token 响应属性的名称和 cmdlet 参数因 cmdlet 而异。有关详细信息，请参阅每个 cmdlet 的帮助文档。

手动迭代

以下示例使用 [do](#) 循环从存储桶返回所有 S3 对象，该循环在每次迭代后评估条件。do 循环执行迭代，直到 `Get-S3Object` 将 `$result.NextMarker` 设置为 `$null`，指示没有更多分页数据剩余。循环的输出被赋值给 `$s3objects` 变量。

```
$s3objects = do  
{  
    $splatParams = @{  
        BucketName = 'amzn-s3-demo-bucket1'  
        MaxKey = 500  
        Marker = $result.NextMarker  
        NoAutoIteration = $true  
        Select = '*'  
    }  
    $result = Get-S3Object @splatParams  
  
    $result.S3Objects  
}  
while ($null -ne $result.NextMarker)
```

此示例使用 PowerShell [splatting](#) 以避免因内联声明参数和参数而导致代码行过长。

凭证和配置文件解析

凭证搜索顺序

运行命令时，AWS 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AWS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AWS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 AWS 配置文件。
- b. default 共享凭证文件中的 AWS 配置文件。
- c. AWS PS Default 开发工具包存储中的 AWS 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 [适用于 .NET 的 AWS SDK](#)。

如果此搜索未能找到指定的凭证，则该命令会引发异常。

有关用户和角色的其它信息

要在 AWS 上运行 Tools for PowerShell 命令，您需要有适合您的任务的用户、权限集和服务角色的某种组合。

您创建的特定用户、权限集和服务角色以及使用它们的方式，将取决于您的要求。下面的一些附加信息介绍了可能使用它们的原因以及如何创建它们。

用户和权限集

尽管可以使用具有长期凭证的 IAM 用户账户来访问 AWS 服务，但这已不再是最佳实践，应予以避免。即使在开发过程中，最佳做法也是在 AWS IAM Identity Center 中创建用户和权限集，并使用身份源提供的临时凭证。

对于开发，您可以使用自己创建的或在[配置工具身份验证](#)中提供的用户。如果您拥有适当的 AWS 管理控制台 权限，还可以为该用户创建具有最低权限的不同权限集，或者创建专用于开发项目的新用户，同时提供具有最低权限的权限集。您选择的行动方案（如果有）取决于您的情况。

有关这些用户和权限集以及如何创建它们的更多信息，请参阅《AWS 开发工具包和工具参考指南》中的[身份验证和访问](#)以及《AWS IAM Identity Center 用户指南》中的[入门](#)。

服务角色

您可以设置 AWS 服务角色来代表用户访问 AWS 服务。如果有多人远程运行您的应用程序，则这种访问方式是合适的；例如，在您为此目的创建的 Amazon EC2 实例上。

创建服务角色的过程因情况而异，但基本上如下所示。

1. 登录 AWS 管理控制台，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 角色，然后选择 创建角色。
3. 选择 AWS 服务，找到并选择 EC2（示例），然后选择 EC2 使用案例（示例）。
4. 选择下一步，然后为您的应用程序将使用的 AWS 服务选择[适当的策略](#)。

Warning

请勿选择 AdministratorAccess 策略，因为该策略允许对您账户中的几乎所有内容具有读写权限。

5. 选择下一步。输入角色名称、描述和您想要的任何标签。

您可以在 [IAM 用户指南](#) 中的 [使用 AWS 资源标签控制访问](#) 中，找到标签相关的信息。

6. 选择创建角色。

在 [《IAM 用户指南》](#) 中的 [IAM 身份（用户、用户组和角色）](#) 中，您可以找到有关 IAM 角色的高级信息。在 [IAM 角色](#) 主题中查找有关角色的详细信息。

使用旧版凭证

本部分中的主题提供有关在不使用 AWS IAM Identity Center 的情况下使用长期或短期凭证的信息。

Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供者的联合身份验证，例如 [AWS IAM Identity Center](#)。

Note

这些主题中的信息适用于需要手动获取和管理短期或长期凭证的情况。有关短期和长期凭证的更多信息，请参阅 AWS SDKs 和工具参考指南中的 [其他身份验证方式](#)。

要了解最佳安全实践，请使用 AWS IAM Identity Center，如中所述 [配置工具身份验证](#)。

有关凭证的重要警告和指南

有关凭证的警告

- 请勿使用您账户的根凭证访问 AWS 资源。这些凭证可提供不受限的账户访问且难以撤销。
- 请勿在命令或脚本中按字面输入访问密钥或凭证信息。否则，会产生意外泄露凭证的风险。
- 请注意，存储在共享 AWS credentials 文件中的任何凭据都以纯文本形式存储。

有关安全管理凭证的更多指南

有关如何安全管理 AWS 证书的一般性讨论，请参阅 IAM 用户指南中的 [AWS 安全证书](#) [AWS 一般参考](#) 和 [《IAM 用户指南》](#) 中的 [安全最佳实践和用例](#)。除了上述讨论内容外，请考虑以下事项：

- 创建其他用户，例如 IAM Identity Center 中的用户，并使用这些用户的凭证，而不是使用您的 AWS 根用户凭证。如有必要，可以撤销其他用户的凭证，或者这些凭证本来就是临时的。此外，您可以对每个用户应用仅允许访问某些资源和操作的策略，从而采取最低权限的立场。
- 对于 Amazon Elastic Container Service (Amazon ECS)，使用 [适用于任务的 IAM 角色](#)。
- 对于在 Amazon EC2 实例上运行的应用程序，使用 [IAM 角色](#)。

主题

- [使用 AWS 凭证](#)
- [中的共享凭据 AWS Tools for PowerShell](#)

使用 AWS 凭证

每个 AWS Tools for PowerShell 命令都必须包含一组 AWS 凭据，这些凭据用于对相应的 Web 服务请求进行加密签名。您可以为每条命令、每个会话或所有会话指定凭证。

Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供者的联合身份验证，例如 [AWS IAM Identity Center](#)。

Note

本主题中的信息适用于需要手动获取和管理短期或长期凭证的情况。有关短期和长期凭证的更多信息，请参阅 AWS SDKs 和工具参考指南中的 [其他身份验证方式](#)。

要了解最佳安全实践，请使用 AWS IAM Identity Center，如中所述 [配置工具身份验证](#)。

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，AWS Tools for PowerShell 将检索关联的凭证。有关如何安全管理 AWS 证书的一般性讨论，请参阅中的 [管理 AWS 访问密钥的最佳实践](#) [Amazon Web Services 一般参考](#)。

Note

您需要一个 AWS 帐户才能获取凭证并使用 AWS Tools for PowerShell。要创建 AWS 帐户，请参阅 [入门：你是首次 AWS 使用吗？](#) 在《AWS 帐户管理 参考指南》中。

主题

- [凭证存储位置](#)
- [管理配置文件](#)
- [指定凭证](#)

- [凭证搜索顺序](#)
- [中的凭证处理 AWS Tools for PowerShell Core](#)

凭证存储位置

AWS Tools for PowerShell 可以使用两个凭证存储中的任何一个：

- S AWS DK 存储，用于加密您的凭据并将其存储在您的主文件夹中。在 Windows 中，此存储位于：`C:\Users\username\AppData\Local\AWSToolkit\RegisteredAccounts.json`。

[适用于 .NET 的 AWS SDK](#) 和 [Toolkit for Visual Studio](#) 也可以使用 AWS 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`

AWS SDKs 和 AWS Command Line Interface 也可以使用凭据文件。如果您在 AWS 用户环境之外运行脚本，请确保将包含您的凭据的文件复制到所有用户帐户（本地系统和用户）都可以访问您的凭据的位置。

管理配置文件

配置文件允许您使用引用不同的凭据集 AWS Tools for PowerShell。您可以使用 AWS Tools for PowerShell cmdlet 在 SD AWS K 商店中管理您的个人资料。您也可以使用 [Toolkit for Visual Studio](#) 或使用 [适用于 .NET 的 AWS SDK](#) 以编程方式，在 AWS 开发工具包存储中管理配置文件。有关如何在凭证文件中管理配置文件的说明，请参阅[管理 AWS 访问密钥的最佳实践](#)。

添加新的配置文件

要向 S AWS DK 存储区添加新的配置文件，请运行命令 `Set-AWSCredential`。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 AWS SDK 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。

为了向后兼容，已弃用的命令 AWSCredential C [lear-](#) 仍然可用，但 Remove-AWSCredentialProfile 它是首选。

指定凭证

可通过多种方式指定凭证。首选的方法是识别配置文件，而不是将字面凭据合并到命令行中。AWS Tools for PowerShell 使用[凭据搜索顺序中描述的搜索顺序查找](#)配置文件。

在 Windows 上，存储在 AWS SDK 存储中的 AWS 凭据使用登录的 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 AWS Tools for PowerShell 版本中的 `-StoredCredentials` 参数。为了实现向后兼容性，仍支持 `-StoredCredentials`。

默认配置文件（推荐）

如果您的凭据存储在名为的配置文件中，则所有 AWS SDKs 和管理工具都可以在您的本地计算机上自动找到您的凭证 default。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 AWS 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` 允许您为每个 PowerShell 会话指定默认配置文件，但是 cmdlet 会从您的自定义名称配置文件加载凭据，但会使用命名的配置文件覆盖 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，并且您想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

默认凭据包含在 AWS SDK 存储中的 default 配置文件名称下。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，则 PowerShell 会自动从实例配置文件中获取 AWS 证书和区域信息。您不需要运行 `Initialize-AWSDefaultConfiguration`。无需在使用实例配置文件启动的 EC2 实例上运行 `Initialize-AWSDefaultConfiguration` cmdlet，因为它使用的实例配置文件数据与默认情况下 PowerShell 已使用的实例配置文件数据相同。

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 1.1 之前的 Windows PowerShell 工具版本中，`Set-AWSCredential` cmdlet 无法正常运行，并且会覆盖 “” 指定的配置文件。MyProfileName 我们建议使用最新版本的 Windows 工具 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 AWS 区域](#)。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 AWS 共享凭据文件位于用户的主文件夹中（`C:\Users\username\.aws` 在 Windows 或 Linux `~/.aws` 上）。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

如果您在通常未登录的时间内运行 PowerShell 脚本 AWS（例如，您在正常工作时间之外将 PowerShell 脚本作为计划任务运行），请在指定要使用的配置文件时添加该 `-ProfileLocation` 参数，并将该值设置为存储您的凭据的文件的完整路径。为确保 AWS Tools for PowerShell 脚本使用正确的账户凭据运行，每当脚本在不使用账户的上下文或进程中运行时，都应添加 `-ProfileLocation` 参数。AWS 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，AWS Tools for PowerShell 按以下顺序搜索凭据。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，则该命令将在 AWS SDK 存储中查找指定的配置文件，如果该配置文件不存在，则在默认位置从 AWS 共享凭据文件中查找指定的配置文件。
- 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. AWS SDK 商店中的default个人资料。
 - b. AWS 共享凭据文件中的default个人资料。
 - c. AWS SDK 商店中的AWS PS Default个人资料。
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 [适用于 .NET 的 AWS SDK](#)。

如果此搜索未能找到指定的凭证，则该命令会引发异常。

中的凭证处理 AWS Tools for PowerShell Core

中的 Cmdlet 在运行时 AWS Tools for PowerShell Core 接受 AWS 访问和密钥或凭据配置文件的名称，类似于。AWS Tools for Windows PowerShell当它们在 Windows 上运行时，这两个模块都能够访问 适用于 .NET 的 AWS SDK 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 AWS Tools for PowerShell 搜索凭据配置文件的第一个文件，也是 AWS Tools for PowerShell 存储凭据配置文件的文件。有关 适用于 .NET 的 AWS SDK 凭据存储文件的更多信息，请参阅[配置 AWS 凭据](#)。Windows 工具 PowerShell 模块目前不支持将凭据写入其他文件或位置。

两个模块都可以从 other AWS SDKs 和 ther 使用的 AWS 共享凭据文件中读取配置文件 AWS CLI。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。

SDK 凭据存储区使用 Windows 加密以加密形式保存您的凭证。APIs APIs 它们在其他平台上不可用，因此该 AWS Tools for PowerShell Core 模块仅使用 AWS 共享凭据文件，并支持将新的凭据配置文件写入共享凭据文件。

以下使用 **Set-AWSCredential** cmdlet 的示例脚本显示了在 Windows 上使用命令行管理程序或AWSPower命令行管理程序处理凭据配置文件的选项。AWSPower NetCore模块。

```
# 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
```

以下示例显示了AWSPower命令行管理程序的行为。NetCoreLinux 或 macOS 操作系统上的模块。

```
# 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
```

中的共享凭据 AWS Tools for PowerShell

Windows 工具 PowerShell 支持使用 AWS 共享凭据文件，类似于 AWS CLI 和其他 AWS SDKs。Windows 工具 PowerShell 现在支持读取和写入 .NET assume role 凭据文件和 AWS 共享凭据文件 session、以及凭据配置文件。basic 新的 Amazon.Runtime.CredentialManagement 命名空间支持此功能。

Warning

为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证，而是使用与身份提供者的联合身份验证，例如 [AWS IAM Identity Center](#)。

Note

本主题中的信息适用于需要手动获取和管理短期或长期凭证的情况。有关短期和长期凭证的更多信息，请参阅 AWS SDKs 和工具参考指南中的 [其他身份验证方式](#)。

要了解最佳安全实践，请使用 AWS IAM Identity Center，如中所述 [配置工具身份验证](#)。

[已添加到凭据相关的 cmdlet 中的以下参数 \(初始化-AWSDefault 配置、新建和设置- \) 支持新的配置文件类型和对 AWS 共享凭证文件的访问权限。AWSCredential AWSCredential](#) 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 角色与配合使用 AWS Tools for PowerShell

AWS 共享凭证文件支持其他类型的访问权限。例如，您可以使用 IAM 角色而不是 IAM 用户的长期证书来访问您的 AWS 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。当您告诉 AWS Tools for PowerShell 使用指定角色的配置文件时，会 AWS Tools for PowerShell 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	说明
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。第三方在担任分配的角色时必须

参数名称	说明
	须将 ExternalId 作为参数包括在内。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您的 AWS 资源的访问权限时使用外部 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 AWS 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。
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 Windows PowerShell 服务工具 cmdlet 配合使用，请在命令中添加 -ProfileName 公共参数以引用该角色配置文件。以下示例使用在上一个示例中定义的角色配置文件来访问 [Get-S3Bucket](#) cmdlet。AWS Tools for PowerShell 在中查找证书 my_source_profile，使用这些证书代表用户 AssumeRole 进行调用，然后使用这些临时角色证书进行调用 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	-AccessKey
这些是 IAM 用户的长期凭证	-SecretKey
会话： 这些是您手动检索的 IAM 角色的短期证书，例如直接调用 Use -STSRole cmdlet。	-AccessKey -SecretKey -SessionToken
角色： 这些是 AWS Tools for PowerShell 为您检索的 IAM 角色的短期凭证。	-SourceProfile -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。添加该 `-ProfileLocation` 参数可控制适用于 Windows 的工具是 PowerShell 使用共享凭据文件还是 .NET 凭据文件。下表描述了该参数在 Windows 工具中的工作原理 PowerShell。

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。如果找不到个人资料，请在以下位置搜索 AWS 共享凭据文件 (<i>user's home directory</i>) <code>\.aws\credentials</code> 。
AWS 共享凭据文件格式的文件路径	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\user\.aws\credentials -ProfileName
    basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\user\.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` 参数以返回凭据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

要删除凭据配置文件，请运行新的 `Remove-Profile cmd AWSCredential let. C@@@ lea-` 已 AWSCredential 被弃用，但仍可用于向后兼容。

重要提示

只有“[初始化-AWSDefault 配置](#)” `AWSCredential`、“[新建](#)”和“[设置](#)” `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。

AWS Tools for Windows PowerShell 的功能

本部分中的某些主题提供了关于 Windows PowerShell 工具功能的信息，您在创建项目和脚本时可能需要参考这些信息。本节中的其它主题提供了有关配置这些工具的高级方法、您的环境和项目的信息。请确保您首先已[安装](#)并[设置](#)这些工具。

有关为特定 AWS 服务开发软件及相关代码示例的信息，请参阅[使用 AWS 服务](#)。有关其他代码示例，请参阅 [PowerShell V4 代码示例工具](#)。

主题

- [可观测性](#)

可观测性

可观测性是指可以根据系统发出的数据推断出其当前状态的程度。发出的数据通常称为遥测。有关使用 AWS 服务时遥测的更多信息，请参阅[《适用于 .NET 的 AWS SDK 开发人员指南》](#)中的[可观测性](#)。

以下代码展示了如何在 AWS Tools for PowerShell 中启用可观测性的示例。

```
<#
    This is an example of generating telemetry for AWS Tools for PowerShell.
    Each cmdlet that interacts with an Amazon Web Service creates a trace containing
    spans
    for underlying processes and AWS SDK for .NET operations.
    This example is written using PowerShell 7 and .NET 8.
    It requires the installation of the .NET CLI tool.
    Note that implementation varies by the exporter/endpoint, which is not specified in
    this example.
    For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
#>

# Set this value to a common folder path on your computer for local development of code
# repositories.
$devProjectsPath = [System.IO.Path]::Join('C:', 'Dev', 'Repos')

# If these values are changed, update the hardcoded method invocation toward the end of
# this script.
# Values must follow constraints for namespaces and classes.
$telemetryProjectName = 'ExampleAWSPowerShellTelemetryImplementation'
$serviceName = 'ExamplePowerShellService'
```



```
# This example supposes that the OTLP exporter requires these two properties,
# but some exporters require different properties or no properties.
$telemetryEndPoint = 'https://example-endpoint-provider.io'
$telemetryHeaders = 'x-example-header=abc123'

$dllsPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName, 'bin',
    'Release', 'net8.0', 'publish')

$telemetryProjectPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName)

# This script is designed to recreate the example telemetry project each time it's
# executed.
Remove-Item -Path $telemetryProjectPath -Recurse -Force -ErrorAction 'SilentlyContinue'
$null = New-Item -Path $devProjectsPath -Name $telemetryProjectName -ItemType
    'Directory'

<#
    Create and build a C#-based .NET 8 project that implements
    OpenTelemetry Instrumentation for the AWS Tools for PowerShell.
#>

Set-Location -Path $telemetryProjectPath

dotnet new classlib

# Other exporters are available.
# For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
dotnet add package OpenTelemetry.Exporter.OpenTelemetryProtocol
dotnet add package OpenTelemetry.Instrumentation.AWS

$classContent = @"
using OpenTelemetry;
using OpenTelemetry.Resources;
using OpenTelemetry.Trace;

namespace Example.Telemetry;

public class AWSToolsForPowerShellTelemetry
{
    public static void InitializeAWSInstrumentation()
    {
        Sdk.CreateTracerProviderBuilder()
            .ConfigureResource(e => e.AddService("$ServiceName"))
    }
}
```

```

        .AddAWSInstrumentation()
        // Exporters vary so options might need to be changed or omitted.
        .AddOtlpExporter(options =>
        {
            options.Endpoint = new Uri("$telemetryEndPoint");
            options.Headers = "$telemetryHeaders";
        })
        .Build();
    }
}
"@

$csFilePath = [System.IO.Path]::Join($telemetryProjectPath, ($serviceName + '.cs'))
Set-Content -Path $csFilePath -Value $classContent

dotnet build
dotnet publish -c Release

<#
    Add additional modules here for any other cmdlets that you require.
    Beyond this point, additional AWS Tools for PowerShell modules will fail to import
    due to conflicts with the AWS SDK for .NET assemblies that are added next.
#>

Import-Module -Name 'AWS.Tools.Common'
Import-Module -Name 'AWS.Tools.DynamoDBv2'

# Load assemblies for the telemetry project, excluding the AWS SDK for .NET assemblies
# that were already loaded by importing AWS Tools for PowerShell modules.
$dlls = (Get-ChildItem $dllsPath -Filter *.dll -Recurse ).FullName
$AWSSDKAssembliesAlreadyLoaded =
    [Threading.Thread]::GetDomain().GetAssemblies().Location | Where-Object {$_ -like
        '*AWSSDK*' } | Split-Path -Leaf
$dlls.Where{$AWSSDKAssembliesAlreadyLoaded -notcontains ($_ | Split-Path -
    Leaf)}.ForEach{Add-Type -Path $_}

# Invoke the method defined earlier in this script.
[Example.Telemetry.AWSToolsForPowerShellTelemetry]::InitializeAWSInstrumentation()

<#
    Now telemetry will be exported for AWS Tools for PowerShell cmdlets
    that are invoked directly or indirectly.
    Execute this cmdlet or execute your own PowerShell script.

```

```
#>
```

```
Get-DDBTable -TableName 'DotNetTests-HashTable' -Region 'us-east-1'
```

使用中的 AWS 服务 AWS Tools for PowerShell

本节提供了使用 AWS Tools for PowerShell 访问 AWS 服务的示例。这些示例有助于演示如何使用 cmdlet 执行实际 AWS 任务。这些示例依赖于工具提供 PowerShell 的 cmdlet。要查看有哪些 cmdlet 可用，请参阅 [AWS Tools for PowerShell Cmdlet 参考](#)。

PowerShell 文件串联编码

中的某些 cmdle AWS Tools for PowerShell t 会编辑您现有的文件或记录。AWS 例如 Edit-R53ResourceRecordSet，它调用了 Amazon Route 53 [ChangeResourceRecordSets](#) 的 API。

在 PowerShell 5.1 或更早版本中编辑或连接文件时，将以 UTF-16 而不是 UTF-8 对输出进行 PowerShell 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。

为避免将文件输出转换为 UTF-16，您可以通过管道将命令传送到 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 AWS CLI 命令，则同样的行为适用。您可以通过管道将 AWS CLI 命令的输出传送到 PowerShell 控制台 Out-File 中。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 C PowerShell ore）会自动保留串联文件输出的 UTF-8 编码。

PowerShell 工具返回的对象

为了在原生 PowerShell 环境中 AWS Tools for PowerShell 更有用，AWS Tools for PowerShell cmdlet 返回的对象是 .NET 对象，而不是通常从软件开发工具包中相应 API 返回的 JSON 文本对象。AWS 例如，Get-S3Bucket 发出 Buckets 集合，而不是 Amazon S3 JSON 响应对象。可以

将 Buckets 馆藏放入 PowerShell 管道中，并以适当的方式与之互动。同样，Get-EC2Instance 发出一个 Reservation .NET 对象集合，而不是 DescribeEC2Instances JSON 结果对象。这种行为是设计使 AWS Tools for PowerShell 体验与惯用语 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](#)

本节演示了启动 Amazon EC2 实例所需的步骤，包括如何：

- 检索 Amazon 系统映像列表 (AMIs)。
- 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](#)

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。

[AWS Lambda 和 AWS Tools for PowerShell](#)

本节简要概述了适用于 PowerShell 模块的 AWS Lambda 工具，并描述了设置该模块所需的步骤。

[Amazon SNS 和 Amazon SQS](#)

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](#)

此部分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

- [开始使用 AWS Tools for Windows PowerShell](#)

主题

- [Amazon S3 与 Tools for Windows PowerShell](#)
- [Amazon EC2 与 Tools for Windows PowerShell](#)
- [AWS Lambda 和 AWS Tools for PowerShell](#)
- [亚马逊 SQS、亚马逊 SNS 和适用于 Windows 的工具 PowerShell](#)
- [CloudWatch 来自 AWS Tools for Windows PowerShell](#)
- [在 cmdlet 中使用 ClientConfig 参数](#)

Amazon S3 与 Tools for Windows PowerShell

在本节中，我们通过使用 Amazon S3 和 CloudFront 的 AWS 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 [AWS 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

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [在 Amazon S3 上托管静态网站](#)
- [Amazon S3 控制台](#)

主题

- [创建 Amazon S3 存储桶，验证它的区域，然后删除它（可选）](#)
- [将 Amazon S3 存储桶配置为网站并启用日志记录](#)
- [将对象上传到 Amazon S3 存储桶](#)
- [删除 Amazon S3 对象和存储桶](#)
- [将内联文本内容上传到 Amazon S3](#)

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

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [放置存储桶 \(Amazon S3 服务参考 \)](#)
- [AWS PowerShell 亚马逊 S3 的区域](#)

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](#)。

```
PS > Write-S3BucketWebsite -BucketName website-example -  
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument  
error.html  
RequestId      : A1813E27995FFDDD  
AmazonId2      : T7hlD0eLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xA1I  
ResponseStream :  
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}  
Metadata       : {}  
ResponseXml    :
```

另请参阅

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [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，与适用于 .NET 的 AWS SDK 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

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [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 AWS 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 AWS Tools for PowerShell 会提示您进行确认。

另请参阅

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [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 amzn-s3-demo-bucket -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 amzn-s3-demo-bucket -key myobject.txt -content @"
>> line 1
>> line 2
>> line 3
>> "e
>>

PS > # Specifying content from a variable: (note final newline needed to end in-line
here-string)
PS > $x = @"
>> line 1
>> line 2
>> line 3
>> "e
>>

PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content $x
```

Amazon EC2 与 Tools for Windows PowerShell

您可以使用 AWS 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 [开始使用 AWS Tools for Windows PowerShell](#)。

主题

- [创建密钥对](#)
- [使用 Windows PowerShell 创建安全组](#)
- [使用 Windows 查找亚马逊计算机映像 PowerShell](#)
- [使用 Windows PowerShell 启动 Amazon EC2 实例](#)

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----
                MIIIEogIBAAKCAQEAKK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                Mz6btoxPcE7EMeH1wySUP8nouAS9xbl9l7+VkD74bN9KmNcPa/Mu...
                Zyn4vVe0Q5il/MpkrRogHq0B0rigeTeV5Yc3lv00RFFPu0Kz4kcm...
                w3Jg8dKsWn0pl0pX7V3sRC02KgJIbejQUvBFGi50QK9bm4tXBIEC...
                daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                iuskGkcvgWkcFQkLmRHROdpPb+0dFsZtjHZDpMVfMA9tT8EdbkEF...
                3SrNeqZPsxJJIX0odb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                GGLlFEgB95KjGIk7zEv2Q7K6s+DHclrDeMZwa7KFNRZuCuX7jssC...
                x098abxMr3o3TNU6p1ZYRJEQ0oJr0W+kC+/8SWb8NIwFLtwhmJEy...
                1BX9X8WFX/A8VLHrT1elrKmlkNECgYEAwltkV1p0JAFhz9p7ZFEv...
                vvVsPaF0Ev9bk9pqhx269PB50x2KokwCagDMMaYvasWobuLmNu/1...
                lmwRx7KTeQ7W1J30LgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                63g6N6rk2FkHZX1E62BgbewUd3eZOS05Ip4VUdvtGcuc8/qa+e5C...
                KXgyt9n164pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                gYALEI7m1jJKpHWAes0hiemw5VmKyIZpzGstSJsFStER1AjiETDH...
                YAtnI4J8dRyP9I7B0V0n3wNfIjk85gi1/00c+j8S65giLAfndWGR...
                9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvUkePbUDTD...
                AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hk0SmWhlZGWikJ5VizBf...
                drkBr/vTKVRMTi3lVFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                TTld5N6opg0XAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                x302duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzc0ajwieNrbl7c...
                -----END RSA PRIVATE KEY-----

```

KeyMaterial 成员存储密钥对的私有密钥。公有密钥存储在AWS中。您无法从AWS检索公有密钥，但是，您可以通过将私有密钥的 KeyFingerprint 与AWS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 KeyFingerprint 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 AWS Tools for PowerShell 或 适用于 .NET 的 AWS SDK 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 AWS 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 AWS 中。

使用 Windows PowerShell 创建安全组

您可以使用 AWS Tools for PowerShell 来创建和配置安全组。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

- [先决条件](#)
- [为 EC2-VPC 创建安全组](#)

先决条件

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-VPC 创建安全组

 Warning

EC2-Classic 已于 2022 年 8 月 15 日停用。我们建议您从 EC2-Classic 迁移到 VPC。有关更多信息，请参阅博客文章 [EC2-Classic Networking is Retiring - Here's How to Prepare](#)。

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 查找亚马逊计算机映像 PowerShell

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。但是，IDs AWS Windows 版经常 AMIs 更改，因为 AWS AMIs 提供了最新的更新和安全增强功能。您可以使用[Get-EC2Image](#)和 [Get-EC2ImageByName](#)cmdlet 来查找当前 Windows AMIs 并获取它们。IDs

主题

- [Get-EC2Image](#)
- [Get-EC2ImageByName](#)

Get-EC2Image

Get-EC2Imagecmdlet 会检索您可以 AMIs 使用的列表。

将-Owner参数与数组值一起使用，amazon，self AMIs 以便仅Get-EC2Image检索属于 Amazon 或您的数组。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，使用以下筛选器仅显示 Windows AMIs。

```
$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
```

以下是 cmdlet AMIs 返回的命令的示例；上一个命令的实际输出为许多命令提供了信息。AMIs

```
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-EC2ImageByNamescmdlet 允许您 AMIs 根据感兴趣的服务器配置类型筛选 AWS Windows 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](#)
- [使用 Windows PowerShell 创建安全组](#)
- [使用 Windows PowerShell 查找 Amazon Machine Image](#)

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 VPC 中启动实例](#)
- [在 VPC 中启动 Spot 实例](#)

在 VPC 中启动实例

Warning

EC2-Classic 已于 2022 年 8 月 15 日停用。我们建议您从 EC2-Classic 迁移到 VPC。有关更多信息，请参阅博客文章 [EC2-Classic Networking is Retiring - Here's How to Prepare](#)。

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

```

AWS Lambda 和 AWS Tools for PowerShell

通过使用 [AWSLambdaPSCore](#) 模块，您可以使用 .NET PowerShell Core 2.1 运行时在 Core 6.0 中开发 AWS Lambda 函数。PowerShell 开发人员可以使用 Lambda 在 PowerShell 环境中管理 AWS 资源和编写自动化脚本。PowerShell Lambda 中的支持允许您运行 PowerShell 脚本或函数来响应任何 Lambda 事件，例如 Amazon S3 事件或亚马逊计划事件。CloudWatch 该 AWSLambdaPSCore 模块是一个单独的 AWS 模块 PowerShell；它不是其中的一部分 AWS Tools for PowerShell，安装该 AWSLambdaPSCore 模块也不会安装 AWS Tools for PowerShell。

安装 AWSLambdaPSCore 模块后，您可以使用任何可用的 PowerShell cmdlet (或开发自己的模块) 来创作无服务器函数。AWS Lambda Tools for PowerShell 模块包括用于 PowerShell 基于无服务器的应用程序的项目模板以及用于向其发布项目的工具。AWS

AWSLambdaPSCore 所有支持 Lambda 的地区都提供模块支持。有关支持的区域的更多信息，请参阅 [AWS 区域列表](#)。

先决条件

在安装和使用该 AWSLambdaPSCore 模块之前，需要执行以下步骤。有关这些步骤的更多详细信息，请参阅《AWS Lambda 开发人员指南》中的 [设置开发环境](#)。PowerShell

- 安装正确的版本 PowerShell — Lambda 对 PowerShell 的支持基于跨平台 PowerShell 酷睿 6.0 版本。你可以在 Windows、Linux 或 PowerShell Mac 上开发 Lambda 函数。如果你至少没有 PowerShell 安装此版本的，则可以在 [Microsoft PowerShell 文档网站上](#) 找到相关说明。
- 安装 .NET Core 2.1 SDK — 由于 PowerShell 核心基于 .NET 内核，因此 Lambda 支持对 .NET 核心和 Lambda 函数 PowerShell 使用相同的 .NET Core 2.1 Lambda 运行时。PowerShell Lambda PowerShell 发布 cmdlet 使用 .NET Core 2.1 软件开发工具包创建 Lambda 部署包。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

完成先决条件后，就可以安装该 AWSLambdaPSCore 模块了。在 PowerShell 核心会话中运行以下命令。

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

您已经准备好开始在中开发 Lambda 函数了。PowerShell 有关如何入门的更多信息，请参阅 [开发人员指南 PowerShell 中的创作 Lambda 函数 AWS Lambda 的编程模型](#)。

另请参阅

- [在开发者博客上宣布对 Core 的 Lambda S PowerShell support AWS 支持](#)
- [AWSLambdaPSCore PowerShell 画廊网站上的模块](#)
- [设置 PowerShell 开发环境](#)
- [AWS 适用于 Powershell 的 Lambda 工具已开启 GitHub](#)
- [AWS Lambda 控制台](#)

亚马逊 SQS、亚马逊 SNS 和适用于 Windows 的工具 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
- 授予 IAM 用户或 AWS 账户发布到 SNS 主题和读取 SQS 队列消息的权限。
- 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 myTopic 执行 sns:Publish 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 myQueue 执行 sqs:ReceiveMessage 和 sqs:DeleteMessage 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" :
        "11E17A2+X0uJZnw3TlgcXz4C4KPLXZxbxoEMIirelh13u/oxkWmz5+9tJKFMns1Z0qQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1b03hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
vyuGkJReV710JWPHnCN
+qTE2lId2RPkF0eGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3Q06u9i9qBFt0ekJFZNJp6Avu05hIklb4yo
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 :
AQEB2vvk1e5c0KFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJb17F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHn18S1bQa0R/kD+Saq40Ivfj8x3M40h1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
+HmXdkax2Wd+9AxrHlQZV5ur1MoByKWwBDbSqoYJTJquCc10gWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
        sHN12776axknhg3j9K/Xwj54DixdsegrnrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2Msey0eM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==

```

CloudWatch 来自 AWS Tools for Windows PowerShell

本节演示如何使用适用于 Windows 的工具将自定义指标数据发布 PowerShell 到的示例 CloudWatch。

此示例假设您已为 PowerShell 会话设置了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 代码初始化 CloudWatch MetricDatum 对象并将其发布到服务。您可以通过导航到 [CloudWatch 控制台](#) 来查看此操作的结果。

```
$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
```

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

- [使用中的 AWS 服务 AWS Tools for PowerShell](#)
- [AmazonCloudWatchClient. PutMetricData](#) (.NET 开发工具包参考)
- [MetricDatum](#) (服务 API 参考)
- [亚马逊 CloudWatch 控制台](#)

在 cmdlet 中使用 ClientConfig 参数

当您连接到某个服务时，`ClientConfig` 参数可用于指定某些配置设置。此参数大多数可能的属性都是在 [Amazon.Runtime.ClientConfig](#) 类中定义的，该类继承到 AWS 服务的 API 中。有关简单继承的示例，请参阅 [Amazon.Keyspaces.AmazonKeyspacesConfig](#) 类。此外，一些服务定义了仅适用于该服务的附加属性。有关已定义的其他属性的示例，请参阅 [Amazon.S3.AmazonS3Config](#) 类，特别是 `ForcePathStyle` 属性。

使用 ClientConfig 参数

要使用 `ClientConfig` 参数，您可以在命令行上将其指定为 `ClientConfig` 对象，也可以使用 PowerShell splatting 方法将参数值集合作为一个单元传递给命令。以下示例中显示了这些方法。这些示例假设已安装并导入 `AWS.Tools.S3` 模块，并且您拥有具有适当权限的 [default] 凭据配置文件。

定义一个 `ClientConfig` 对象

```
$s3Config = New-Object -TypeName Amazon.S3.AmazonS3Config
$s3Config.ForcePathStyle = $true
$s3Config.Timeout = [TimeSpan]::FromMilliseconds(150000)
Get-S3Object -BucketName <BUCKET_NAME> -ClientConfig $s3Config
```

使用 PowerShell splating 添加 **ClientConfig** 属性

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

使用未定义的属性

使用 PowerShell splatting 时，如果您指定的 ClientConfig 属性不存在，则直到运行时 AWS Tools for PowerShell 才会检测到错误，此时它会返回异常。修改上面的示例：

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
        UndefinedProperty="Value"
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

此示例生成类似以下内容的例外：

```
Cannot bind parameter 'ClientConfig'. Cannot create object of type
"Amazon.S3.AmazonS3Config". The UndefinedProperty property was not found for the
Amazon.S3.AmazonS3Config object.
```

指定 AWS 区域

您可以使用 `ClientConfig` 参数为命令设置 AWS 区域。区域是通过 `RegionEndpoint` 属性设置的。AWS Tools for PowerShell 根据以下优先级计算要使用的区域：

1. `-Region` 参数
2. `ClientConfig` 参数中传递的区域
3. PowerShell 会话状态
4. 共享 AWS config 文件
5. 环境变量
6. Amazon EC2 实例元数据 (如果启用)。

PowerShell V4 代码示例工具

本主题中的代码示例向您展示了如何将 AWS Tools for PowerShell V4 与 AWS。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

某些服务包含其他示例类别，这些类别说明如何利用特定于服务的库或函数。

Services

- [使用适用 PowerShell 于 V4 的工具的 ACM 示例](#)
- [使用适用于 PowerShell V4 的工具的应用程序 Auto Scaling 示例](#)
- [WorkSpaces 使用适用于 PowerShell V4 的工具的应用示例](#)
- [使用适用于 PowerShell V4 的工具的 Aurora 示例](#)
- [使用适用于 PowerShell V4 的工具的 Auto Scaling 示例](#)
- [AWS Budgets 使用适用于 PowerShell V4 的工具的示例](#)
- [AWS Cloud9 使用适用于 PowerShell V4 的工具的示例](#)
- [CloudFormation 使用适用于 PowerShell V4 的工具的示例](#)
- [CloudFront 使用适用于 PowerShell V4 的工具的示例](#)
- [CloudTrail 使用适用于 PowerShell V4 的工具的示例](#)
- [CloudWatch 使用适用于 PowerShell V4 的工具的示例](#)
- [CodeCommit 使用适用于 PowerShell V4 的工具的示例](#)
- [CodeDeploy 使用适用于 PowerShell V4 的工具的示例](#)
- [CodePipeline 使用适用于 PowerShell V4 的工具的示例](#)
- [使用适用于 V4 的工具的 Amazon Cognito 身份示例 PowerShell](#)
- [AWS Config 使用适用于 PowerShell V4 的工具的示例](#)
- [使用 V4 工具的 De PowerShell vice Farm 示例](#)
- [Directory Service 使用适用于 PowerShell V4 的工具的示例](#)

- [AWS DMS 使用适用于 PowerShell V4 的工具的示例](#)
- [使用适用于 V4 的工具的 DynamoDB 示例 PowerShell](#)
- [使用适用于 PowerShell V4 的工具的亚马逊 EC2 示例](#)
- [使用适用 PowerShell 于 V4 的工具的 Amazon ECR 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon ECS 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon EFS 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon EKS 示例](#)
- [Elastic Load Balancing-使用适用于 PowerShell V4 的工具的版本 1 示例](#)
- [Elastic Load Balancing-使用适用于 PowerShell V4 的工具的版本 2 示例](#)
- [使用适用于 PowerShell V4 的工具的亚马逊 FSx 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon Glacier 示例](#)
- [AWS Glue 使用适用于 PowerShell V4 的工具的示例](#)
- [AWS Health 使用适用于 PowerShell V4 的工具的示例](#)
- [使用适用于 PowerShell V4 的工具的 IAM 示例](#)
- [使用适用于 V4 的工具的 Kinesis 示例 PowerShell](#)
- [使用适用于 V4 的工具的 Lambda 示例 PowerShell](#)
- [使用适用于 PowerShell V4 的工具的 Amazon 机器学习示例](#)
- [使用 V4 版工具的 Macie 示例 PowerShell](#)
- [AWS 价目表 使用适用于 PowerShell V4 的工具的示例](#)
- [使用适用于 PowerShell V4 的工具的 Resource Groups 示例](#)
- [Resource Groups 使用适用于 V4 的 PowerShell 工具标记 API 示例](#)
- [使用 PowerShell V4 工具的 Route 53 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon S3 示例](#)
- [使用适用于 V4 的工具的 Security Hub CSPM 示例 PowerShell](#)
- [使用适用于 PowerShell V4 的工具的 Amazon SES 示例](#)
- [使用适用 PowerShell 于 V4 的工具的 Amazon SES API v2 示例](#)
- [使用适用于 V4 的工具的 Amazon SNS 示例 PowerShell](#)
- [使用适用于 V4 的工具的亚马逊 SQS 示例 PowerShell](#)
- [AWS STS 使用适用于 PowerShell V4 的工具的示例](#)
- [支持 使用适用于 PowerShell V4 的工具的示例](#)

- [使用 PowerShell V4 版工具的 Systems Manager 示例](#)
- [使用适用于 PowerShell V4 的工具的 Amazon Translate 示例](#)
- [AWS WAFV2 使用适用于 PowerShell V4 的工具的示例](#)
- [WorkSpaces 使用适用于 PowerShell V4 的工具的示例](#)

使用适用 PowerShell 于 V4 的工具的 ACM 示例

以下代码示例向您展示了如何使用带有 ACM 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-ACMCertificate

以下代码示例演示了如何使用 Get-ACMCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例演示如何使用证书的 ARN 返回证书及其链。

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetCertificate](#) 中的。

Get-ACMCertificateDetail

以下代码示例演示了如何使用 Get-ACMCertificateDetail。

适用于 PowerShell V4 的工具

示例 1：返回指定证书的详细信息。

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

输出：

```
CertificateArn      : arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt           : 1/21/2016 5:55:59 PM
DomainName          : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy             : {}
IssuedAt            : 1/1/0001 12:00:00 AM
Issuer              :
KeyAlgorithm         : RSA-2048
NotAfter            : 1/1/0001 12:00:00 AM
NotBefore           : 1/1/0001 12:00:00 AM
RevocationReason     :
RevokedAt           : 1/1/0001 12:00:00 AM
Serial              :
SignatureAlgorithm    : SHA256WITHRSA
Status              : PENDING_VALIDATION
Subject             : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeCertificate](#) 中的。

Get-ACMCertificateList

以下代码示例演示了如何使用 Get-ACMCertificateList。

适用于 PowerShell V4 的工具

示例 1：检索所有证书的列表 ARNs 以及每个证书的域名。cmdlet 将自动分页以检索所有。ARNs 要手动控制分页，请使用 -MaxItem 参数控制每次服务调 ARNs 用返回的证书数量，使用 -NextToken 参数表示每次调用的起点。

```
Get-ACMCertificateList
```

输出：

```
CertificateArn
DomainName
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
www.example.com
```

示例 2：检索所有证书的列表，ARNs 其中证书状态与所提供的状态相匹配。

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

示例 3：此示例返回 us-east-1 区域中密钥类型为 RSA_2048 且扩展密钥用法或用途为 CODE_SIGNING 的所有证书的列表。您可以在过滤 ListCertificates 器 API 参考主题中找到这些筛选参数的值：https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html。

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

输出：

```
CertificateArn
DomainName
-----
-----
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-af8d-2133d8f30zzz
*.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-a734-800430c80zzz
nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-bad559525zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-e7ca-44c5-803e-24d9f2f36zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-1241-4b71-80b1-090305a62zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-f94617c99zzz
```

```
arn:aws:acm:us-east-1:8xxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz  
  
arn:aws:acm:us-east-1:8xxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-a714-2d277d3eezzz  
*.route53docs.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListCertificates](#) 中的。

New-ACMCertificate

以下代码示例演示了如何使用 New-ACMCertificate。

适用于 PowerShell V4 的工具

示例 1：创建新证书。该服务返回新证书的 ARN。

```
New-ACMCertificate -DomainName "www.example.com"
```

输出：

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

示例 2：创建新证书。该服务返回新证书的 ARN。

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName  
"example.com","www.example.net"
```

输出：

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RequestCertificate](#) 中的。

Remove-ACMCertificate

以下代码示例演示了如何使用 Remove-ACMCertificate。

适用于 PowerShell V4 的工具

示例 1：删除由提供的 ARN 和关联的私有密钥标识的证书。在继续操作之前，cmdlet 将提示您进行确认；添加 -Force 开关可禁止确认。

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteCertificate](#) 中的。

Send-ACMValidationEmail

以下代码示例演示了如何使用 Send-ACMValidationEmail。

适用于 PowerShell V4 的工具

示例 1：请求发送用于验证“www.example.com”的域所有权的电子邮件。如果您的 shell ConfirmPreference 的 \$ 设置为“中”或更低，cmdlet 将在继续操作之前提示您进行确认。添加 -Force 开关可禁止确认提示。

```
$params = @{  
    CertificateArn="arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"  
    Domain="www.example.com"  
    ValidationDomain="example.com"  
}  
Send-ACMValidationEmail @params
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResendValidationEmail](#) 中的。

使用适用于 PowerShell V4 的工具的应用程序 Auto Scaling 示例

以下代码示例向您展示了如何使用带有 Application Auto Scaling 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-AASScalableTarget

以下代码示例演示了如何使用 Add-AASScalableTarget。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 注册或更新可扩展目标。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterScalableTarget](#) 中的。

Get-AASScalableTarget

以下代码示例演示了如何使用 Get-AASScalableTarget。

适用于 PowerShell V4 的工具

示例 1：此示例将提供有关指定命名空间中 Application Auto Scaling 可扩展目标的信息。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

输出：

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
```

```

RoleARN          : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScalableTargets](#) 中的。

Get-AASScalingActivity

以下代码示例演示了如何使用 Get-AASScalingActivity。

适用于 PowerShell V4 的工具

示例 1：提供有关前六周指定命名空间中扩展活动的描述性信息。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

输出：

```

ActivityId       : 2827409f-b639-4cdb-a957-8055d5d07434
Cause            : monitor alarm Appstream2-MyFleet-default-scale-in Alarm in state
                  ALARM triggered policy default-scale-in
Description      : Setting desired capacity to 2.
Details          :
EndTime          : 12/14/2019 11:32:49 AM
ResourceId       : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace : appstream
StartTime        : 12/14/2019 11:32:14 AM
StatusCode       : Successful
StatusMessage    : Successfully set desired capacity to 2. Change successfully
                  fulfilled by appstream.

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScalingActivities](#) 中的。

Get-AASScalingPolicy

以下代码示例演示了如何使用 Get-AASScalingPolicy。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 描述了指定服务命名空间的 Application Auto Scaling 扩展策略。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

输出：

```
Alarms                                : {Appstream2-LabFleet-default-scale-out-
Alarm}
CreationTime                          : 9/3/2019 2:48:15 AM
PolicyARN                             : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                     policyName/default-scale-out
PolicyName                            : default-scale-out
PolicyType                            : StepScaling
ResourceId                            : fleet/LabFleet
ScalableDimension                     : appstream:fleet:DesiredCapacity
ServiceNamespace                      : appstream
StepScalingPolicyConfiguration        :
  Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                          : 9/3/2019 2:48:15 AM
PolicyARN                             : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                     policyName/default-scale-in
PolicyName                            : default-scale-in
PolicyType                            : StepScaling
ResourceId                            : fleet/LabFleet
ScalableDimension                     : appstream:fleet:DesiredCapacity
ServiceNamespace                      : appstream
StepScalingPolicyConfiguration        :
  Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScalingPolicies](#) 中的。

Get-AASScheduledAction

以下代码示例演示了如何使用 Get-AASScheduledAction。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了为自动扩缩组计划但尚未运行的操作或尚未达到其结束时间的操作。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

输出：

```
CreationTime      : 12/22/2019 9:25:52 AM
EndTime           : 1/1/0001 12:00:00 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule          : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                  /WeekDaysFleetScaling
ScheduledActionName : WeekDaysFleetScaling
ServiceNamespace   : appstream
StartTime           : 1/1/0001 12:00:00 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScheduledActions](#) 中的。

Remove-AASScalableTarget

以下代码示例演示了如何使用 Remove-AASScalableTarget。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 取消注册 Application Auto Scaling 可扩展目标。取消注册可扩展目标会删除与其关联的扩展策略。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension  
appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

输出：

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

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeregisterScalableTarget](#) 中的。

Remove-AASScalingPolicy

以下代码示例演示了如何使用 Remove-AASScalingPolicy。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定扩展策略。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-out"  
-ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteScalingPolicy](#) 中的。

Remove-AASScheduledAction

以下代码示例演示了如何使用 Remove-AASScheduledAction。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定计划操作。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName  
WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension  
appstream:fleet:DesiredCapacity
```

输出：

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

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteScheduledAction](#) 中的。

Set-AASScalingPolicy

以下代码示例演示了如何使用 Set-AASScalingPolicy。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新策略。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
-PolicyType StepScaling -ResourceId fleet/MyFleet -ScalableDimension
appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
-StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
MetricIntervalUpperBound = 0}
```

输出：

```
Alarms      PolicyARN
-----
{}          arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutScalingPolicy](#) 中的。

Set-AASScheduledAction

以下代码示例演示了如何使用 Set-AASScheduledAction。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新计划操作。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutScheduledAction](#) 中的。

WorkSpaces 使用适用于 PowerShell V4 的工具的应用示例

以下代码示例向您展示了如何使用带 WorkSpaces 应用程序的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-APSResourceTag

以下代码示例演示了如何使用 Add-APSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例向资源添加 AppStream 资源标签

```
Add-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest -Tag @{StackState='Test'} -Select ^Tag
```

输出：

Name	Value
----	----
StackState	Test

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResource](#) 中的。

Copy-APSIImage

以下代码示例演示了如何使用 Copy-APSIImage。

适用于 PowerShell V4 的工具

示例 1：此示例将映像复制到其它区域

```
Copy-APSIImage -DestinationImageName TestImageCopy -DestinationRegion us-west-2 -SourceImageName Powershell
```

输出：

```
TestImageCopy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CopyImage](#) 中的。

Disable-APSUser

以下代码示例演示了如何使用 Disable-APSUser。

适用于 PowerShell V4 的工具

示例 1：此示例禁用 USERPOOL 中的用户

```
Disable-APSUser -AuthenticationType USERPOOL -Username TestUser@lab.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DisableUser](#) 中的。

Enable-APSUser

以下代码示例演示了如何使用 Enable-APSUser。

适用于 PowerShell V4 的工具

示例 1：此示例在 USERPOOL 中启用禁用的用户

```
Enable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableUser](#) 中的。

Get-APSAssociatedFleetList

以下代码示例演示了如何使用 Get-APSAssociatedFleetList。

适用于 PowerShell V4 的工具

示例 1：此示例显示与堆栈关联的实例集

```
Get-APSAssociatedFleetList -StackName PowershellStack
```

输出：

```
PowershellFleet
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListAssociatedFleets](#) 中的。

Get-APSAssociatedStackList

以下代码示例演示了如何使用 Get-APSAssociatedStackList。

适用于 PowerShell V4 的工具

示例 1：此示例显示与实例集关联的堆栈

```
Get-APSAssociatedStackList -FleetName PowershellFleet
```

输出：

```
PowershellStack
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListAssociatedStacks](#) 中的。

Get-APSDirectoryConfigList

以下代码示例演示了如何使用 Get-APSDirectoryConfigList。

适用于 PowerShell V4 的工具

示例 1：此示例显示了在中创建的目录配置 AppStream

```
Get-APSDirectoryConfigList | Select DirectoryName,
    OrganizationalUnitDistinguishedNames, CreatedTime
```

输出：

```
DirectoryName  OrganizationalUnitDistinguishedNames  CreatedTime
-----
Test.com       {OU=AppStream,DC=Test,DC=com}         9/6/2019 10:56:40 AM
contoso.com    {OU=AppStream,OU=contoso,DC=contoso,DC=com}  8/9/2019 9:08:50 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeDirectoryConfigs](#) 中的。

Get-APSFleetList

以下代码示例演示了如何使用 Get-APSFleetList。

适用于 PowerShell V4 的工具

示例 1：此示例显示实例集的详细信息

```
Get-APSFleetList -Name Test
```


输出：

```

Arn                : arn:aws:appstream:us-east-1:1234567890:fleet/Test
ComputeCapacityStatus : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime        : 9/12/2019 5:00:45 PM
Description         : Test
DisconnectTimeoutInSeconds : 900
DisplayName         : Test
DomainJoinInfo      :
EnableDefaultInternetAccess : False
FleetErrors         : {}
FleetType           : ON_DEMAND
IamRoleArn          :
IdleDisconnectTimeoutInSeconds : 900
ImageArn            : arn:aws:appstream:us-east-1:1234567890:image/Test
ImageName           : Test
InstanceType        : stream.standard.medium
MaxUserDurationInSeconds : 57600
Name                : Test
State               : STOPPED
VpcConfig           : Amazon.AppStream.Model.VpcConfig

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeFleets](#) 中的。

Get-APSIImageBuilderList

以下代码示例演示了如何使用 Get-APSIImageBuilderList。

适用于 PowerShell V4 的工具

示例 1：此示例显示了以下内容的详细信息 ImageBuilder

```
Get-APSIImageBuilderList -Name TestImage
```

输出：

```

AccessEndpoints      : {}
AppstreamAgentVersion : 06-19-2019
Arn                  : arn:aws:appstream:us-east-1:1234567890:image-builder/
TestImage
CreatedTime          : 1/14/2019 4:33:05 AM

```

```

Description           :
DisplayName            : TestImage
DomainJoinInfo        :
EnableDefaultInternetAccess : False
IamRoleArn            :
ImageArn              : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors    : {}
InstanceType          : stream.standard.large
Name                  : TestImage
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform              : WINDOWS
State                 : STOPPED
StateChangeReason     :
VpcConfig             : Amazon.AppStream.Model.VpcConfig

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeImageBuilders](#) 中的。

Get-APSImageList

以下代码示例演示了如何使用 Get-APSImageList。

适用于 PowerShell V4 的工具

示例 1：此示例显示私有 AppStream 图片

```
Get-APSImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
arn
```

输出：

DisplayName	ImageBuilderName	Visibility	Arn
-----	-----	-----	---
OfficeApps	OfficeApps	PRIVATE	arn:aws:appstream:us-east-1:123456789012:image/OfficeApps
SessionScriptV2	SessionScriptTest	PRIVATE	arn:aws:appstream:us-east-1:123456789012:image/SessionScriptV2

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeImages](#) 中的。

Get-APSIImagePermission

以下代码示例演示了如何使用 Get-APSIImagePermission。

适用于 PowerShell V4 的工具

示例 1：此示例显示共享图像的 AppStream 图像权限

```
Get-APSIImagePermission -Name Powershell | select SharedAccountId,
@{n="AllowFleet";e={$_.ImagePermissions.AllowFleet}},
@{n="AllowImageBuilder";e={$_.ImagePermissions.AllowImageBuilder}}
```

输出：

```
SharedAccountId AllowFleet AllowImageBuilder
-----
123456789012      True      True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribelImagePermissions](#) 中的。

Get-APSSessionList

以下代码示例演示了如何使用 Get-APSSessionList。

适用于 PowerShell V4 的工具

示例 1：此示例显示实例集的会话列表

```
Get-APSSessionList -FleetName PowershellFleet -StackName PowershellStack
```

输出：

```
AuthenticationType      : API
ConnectionState         : CONNECTED
FleetName                : PowershellFleet
Id                      : d8987c70-4394-4324-a396-2d485c26f2a2
MaxExpirationTime       : 12/27/2019 4:54:07 AM
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
StackName               : PowershellStack
StartTime               : 12/26/2019 12:54:12 PM
```

```
State           : ACTIVE
UserId          : Test
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeSessions](#) 中的。

Get-APSSStackList

以下代码示例演示了如何使用 Get-APSSStackList。

适用于 PowerShell V4 的工具

示例 1：此示例显示 AppStream 堆栈列表

```
Get-APSSStackList | Select DisplayName, Arn, CreatedTime
```

输出：

DisplayName	Arn CreatedTime
-----	---

PowershellStack	arn:aws:appstream:us-east-1:123456789012:stack/
PowershellStack	4/24/2019 8:49:29 AM
SessionScriptTest	arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest	9/12/2019 3:23:12 PM

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeStacks](#) 中的。

Get-APSTagsForResourceList

以下代码示例演示了如何使用 Get-APSTagsForResourceList。

适用于 PowerShell V4 的工具

示例 1：此示例显示 AppStream 资源上的标签

```
Get-APSTagsForResourceList -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest
```

输出：

```
Key          Value
---          -
StackState Test
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListTagsForResource](#) 中的。

Get-APSUsageReportSubscription

以下代码示例演示了如何使用 Get-APSUsageReportSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例显示 AppStreamUsageReport 配置详细信息

```
Get-APSUsageReportSubscription
```

输出：

```
LastGeneratedReportDate S3BucketName                      Schedule
SubscriptionErrors
-----
1/1/0001 12:00:00 AM    appstream-logs-us-east-1-123456789012-sik1hnxe DAILY    {}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeUsageReportSubscriptions](#) 中的。

Get-APSUser

以下代码示例演示了如何使用 Get-APSUser。

适用于 PowerShell V4 的工具

示例 1：此示例显示处于启用状态的用户列表

```
Get-APSUser -AuthenticationType USERPOOL | Select-Object UserName,
AuthenticationType, Enabled
```

输出：

UserName	AuthenticationType	Enabled
-----	-----	-----
foo1@contoso.com	USERPOOL	True
foo2@contoso.com	USERPOOL	True
foo3@contoso.com	USERPOOL	True
foo4@contoso.com	USERPOOL	True
foo5@contoso.com	USERPOOL	True

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeUsers](#) 中的。

Get-APSTUserStackAssociation

以下代码示例演示了如何使用 Get-APSTUserStackAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例显示分配给堆栈的用户列表

```
Get-APSTUserStackAssociation -StackName PowershellStack
```

输出：

AuthenticationType	SendEmailNotification	StackName	UserName
-----	-----	-----	-----
USERPOOL	False	PowershellStack	TestUser1@lab.com
USERPOOL	False	PowershellStack	TestUser2@lab.com

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeUserStackAssociations](#) 中的。

New-APSDirectoryConfig

以下代码示例演示了如何使用 New-APSDirectoryConfig。

适用于 PowerShell V4 的工具

示例 1：此示例在中创建目录配置 AppStream

```
New-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso\ServiceAccount
-ServiceAccountCredentials_AccountPassword MyPass -DirectoryName contoso.com -
OrganizationalUnitDistinguishedName "OU=AppStream,OU=Contoso,DC=Contoso,DC=com"
```

输出：

```
CreatedTime          DirectoryName OrganizationalUnitDistinguishedNames
ServiceAccountCredentials
-----
12/27/2019 11:00:30 AM contoso.com {OU=AppStream,OU=Contoso,DC=Contoso,DC=com}
Amazon.AppStream.Model.ServiceAccountCredentials
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateDirectoryConfig](#) 中的。

New-APSFleet

以下代码示例演示了如何使用 New-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个新 AppStream 舰队

```
New-APSFleet -ComputeCapacity_DesiredInstance 1 -InstanceType stream.standard.medium
-Name TestFleet -DisplayName TestFleet -FleetType ON_DEMAND -
EnableDefaultInternetAccess $True -VpcConfig_SubnetIds "subnet-123ce32","subnet-
a1234cfd" -VpcConfig_SecurityGroupIds sg-4d012a34 -ImageName SessionScriptTest -
Region us-west-2
```

输出：

```
Arn                  : arn:aws:appstream:us-west-2:123456789012:fleet/
TestFleet
ComputeCapacityStatus : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime          : 12/27/2019 11:24:42 AM
Description           :
DisconnectTimeoutInSeconds : 900
DisplayName           : TestFleet
DomainJoinInfo        :
EnableDefaultInternetAccess : True
```

```

FleetErrors           : {}
FleetType             : ON_DEMAND
IamRoleArn            :
IdleDisconnectTimeoutInSeconds : 0
ImageArn              : arn:aws:appstream:us-west-2:123456789012:image/
SessionScriptTest
ImageName             : SessionScriptTest
InstanceType          : stream.standard.medium
MaxUserDurationInSeconds : 57600
Name                  : TestFleet
State                 : STOPPED
VpcConfig             : Amazon.AppStream.Model.VpcConfig

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateFleet](#) 中的。

New-APSImageBuilder

以下代码示例演示了如何使用 New-APSImageBuilder。

适用于 PowerShell V4 的工具

示例 1：此示例在中创建了 Image Builder AppStream

```

New-APSImageBuilder -InstanceType stream.standard.medium -Name TestIB -DisplayName
TestIB -ImageName AppStream-WinServer2012R2-12-12-2019 -EnableDefaultInternetAccess
$True -VpcConfig_SubnetId subnet-a1234cfd -VpcConfig_SecurityGroupIds sg-2d012a34 -
Region us-west-2

```

输出：

```

AccessEndpoints       : {}
AppstreamAgentVersion : 12-16-2019
Arn                   : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime           : 12/27/2019 11:39:24 AM
Description           :
DisplayName            : TestIB
DomainJoinInfo        :
EnableDefaultInternetAccess : True
IamRoleArn            :
ImageArn              : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors    : {}

```



```
InstanceType      : stream.standard.medium
Name              : TestIB
NetworkAccessConfiguration :
Platform          : WINDOWS
State             : PENDING
StateChangeReason :
VpcConfig         : Amazon.AppStream.Model.VpcConfig
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateImageBuilder](#) 中的。

New-APSIImageBuilderStreamingURL

以下代码示例演示了如何使用 `New-APSIImageBuilderStreamingURL`。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个有效期为 2 小时的 ImageBuilder 直播网址

```
New-APSIImageBuilderStreamingURL -Name TestIB -Validity 7200 -Region us-west-2
```

输出：

Expires	StreamingURL
----- 12/27/2019 1:49:13 PM	https://appstream2.us-west-2.aws.amazon.com/authenticate?parameters=eyJ0eXBBIjoiQURNSU4iLCJleHBpcmVzIjoimTU3NzQ1NDU1MyIsImF3c0FjY291bnRJCi6IjM5MzQwM

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) 中的 CreateImageBuilderStreaming 网址](#)。

New-APSSStack

以下代码示例演示了如何使用 New-APSSStack。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个新 AppStream 堆栈

```
New-APSSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
$True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```

输出：

```
AccessEndpoints      : {}
ApplicationSettings  : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                  : arn:aws:appstream:us-west-2:123456789012:stack/TestStack
CreatedTime          : 12/27/2019 12:34:19 PM
Description          :
DisplayName          : TestStack
EmbedHostDomains     : {}
FeedbackURL          :
Name                 : TestStack
RedirectURL          :
StackErrors          : {}
StorageConnectors    : {}
UserSettings         : {Amazon.AppStream.Model.UserSetting,
                        Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
                        Amazon.AppStream.Model.UserSetting}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateStack](#) 中的。

New-APSSstreamingURL

以下代码示例演示了如何使用 New-APSSstreamingURL。

适用于 PowerShell V4 的工具

示例 1：此示例创建堆栈的流式传输 URL

```
New-APSSstreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
TestUser
```

输出：

```
Expires              StreamingURL
-----
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M...
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的 [CreateStreamingURL](#) 网址。

New-APSUsageReportSubscription

以下代码示例演示了如何使用 New-APSUsageReportSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例启用 AppStream 使用情况报告

```
New-APSUsageReportSubscription
```

输出：

S3BucketName	Schedule
-----	-----
appstream-logs-us-east-1-123456789012-sik2hnxe	DAILY

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateUsageReportSubscription](#) 中的。

New-APSUser

以下代码示例演示了如何使用 New-APSUser。

适用于 PowerShell V4 的工具

示例 1：此示例在 USERPOOL 中创建一个用户

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -  
LastName 'aws' -Select ^UserName
```

输出：

```
Test@lab.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateUser](#) 中的。

Register-APSFleet

以下代码示例演示了如何使用 Register-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例使用堆栈注册实例集

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AssociateFleet](#) 中的。

Register-APSUserStackBatch

以下代码示例演示了如何使用 Register-APSUserStackBatch。

适用于 PowerShell V4 的工具

示例 1：此示例将堆栈分配给 USERPOOL 中的用户

```
Register-APSUserStackBatch -UserStackAssociation  
@{AuthenticationType="USERPOOL";SendEmailNotification=  
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [BatchAssociateUserStack](#) 中的。

Remove-APSDirectoryConfig

以下代码示例演示了如何使用 Remove-APSDirectoryConfig。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 AppStream 目录配置

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-APSDirectoryConfig (DeleteDirectoryConfig)" on  
target "contoso.com".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): A
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteDirectoryConfig](#) 中的。

Remove-APSFleet

以下代码示例演示了如何使用 Remove-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例移除已删除的 AppStream 舰队

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

输出：

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

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteFleet](#) 中的。

Remove-APSIImage

以下代码示例演示了如何使用 Remove-APSIImage。

适用于 PowerShell V4 的工具

示例 1：此示例删除映像

```
Remove-APSIImage -Name TestImage -Region us-west-2
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSIImage (DeleteImage)" on target "TestImage".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): A

Applications           : {}
AppstreamAgentVersion  : LATEST
Arn                    : arn:aws:appstream:us-west-2:123456789012:image/
TestImage
BaseImageArn           :
CreatedTime            : 12/27/2019 1:34:10 PM
Description            :
DisplayName            : TestImage
ImageBuilderName       :
ImageBuilderSupported  : True
ImagePermissions       :
Name                   : TestImage
Platform               : WINDOWS
PublicBaseImageReleasedDate : 6/12/2018 12:00:00 AM
State                  : AVAILABLE
StateChangeReason      :
Visibility              : PRIVATE
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteImage](#) 中的。

Remove-APSIImageBuilder

以下代码示例演示了如何使用 Remove-APSIImageBuilder。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 ImageBuilder

```
Remove-APSIImageBuilder -Name TestIB -Region us-west-2
```

输出：

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

```

AccessEndpoints           : {}
AppstreamAgentVersion     : 12-16-2019
Arn                       : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime               : 12/27/2019 11:39:24 AM
Description               :
DisplayName                : TestIB
DomainJoinInfo            :
EnableDefaultInternetAccess : True
IamRoleArn                :
ImageArn                  : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors        : {}
InstanceType              : stream.standard.medium
Name                      : TestIB
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                  : WINDOWS
State                     : DELETING
StateChangeReason         :
VpcConfig                 : Amazon.AppStream.Model.VpcConfig

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteImageBuilder](#) 中的。

Remove-APSIImagePermission

以下代码示例演示了如何使用 Remove-APSIImagePermission。

适用于 PowerShell V4 的工具

示例 1：此示例移除映像的权限

```
Remove-APSIImagePermission -Name Powershell -SharedAccountId 123456789012
```

输出：

```

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

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteImagePermissions](#) 中的。

Remove-APSResourceTag

以下代码示例演示了如何使用 Remove-APSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例从资源中移除 AppStream 资源标签

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest -TagKey StackState
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSResourceTag (UntagResource)" on target
"arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): A
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagResource](#) 中的。

Remove-APSStack

以下代码示例演示了如何使用 Remove-APSStack。

适用于 PowerShell V4 的工具

示例 1：此示例删除堆栈

```
Remove-APSStack -Name TestStack -Region us-west-2
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSStack (DeleteStack)" on target "TestStack".
```



```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): A
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteStack](#) 中的。

Remove-APSUsageReportSubscription

以下代码示例演示了如何使用 Remove-APSUsageReportSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例禁用 AppStream 使用情况报告订阅

```
Remove-APSUsageReportSubscription
```

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteUsageReportSubscription](#) 中的。

Remove-APSUser

以下代码示例演示了如何使用 Remove-APSUser。

适用于 PowerShell V4 的工具

示例 1：此示例从 USERPOOL 中删除用户

```
Remove-APSUser -UserName TestUser@lab.com -AuthenticationType USERPOOL
```

输出：

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-APSUser (DeleteUser)" on target "TestUser@lab.com".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): A
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteUser](#) 中的。

Revoke-APSSession

以下代码示例演示了如何使用 Revoke-APSSession。

适用于 PowerShell V4 的工具

示例 1：此示例撤消了与舰队的会话 AppStream

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ExpireSession](#) 中的。

Start-APSFleet

以下代码示例演示了如何使用 Start-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例启动实例集

```
Start-APSFleet -Name PowershellFleet
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartFleet](#) 中的。

Start-APSIImageBuilder

以下代码示例演示了如何使用 Start-APSIImageBuilder。

适用于 PowerShell V4 的工具

示例 1：此示例启动 ImageBuilder

```
Start-APSIImageBuilder -Name TestImage
```

输出：

```
AccessEndpoints           : {}
AppstreamAgentVersion     : 06-19-2019
Arn                       : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime               : 1/14/2019 4:33:05 AM
Description               :
DisplayName               : TestImage
DomainJoinInfo            :
EnableDefaultInternetAccess : False
IamRoleArn                :
ImageArn                  : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors        : {}
InstanceType              : stream.standard.large
Name                     : TestImage
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                  : WINDOWS
State                     : PENDING
StateChangeReason          :
VpcConfig                 : Amazon.AppStream.Model.VpcConfig
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartImageBuilder](#) 中的。

Stop-APSFleet

以下代码示例演示了如何使用 Stop-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例停止实例集

```
Stop-APSFleet -Name PowershellFleet
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopFleet](#) 中的。

Stop-APSImageBuilder

以下代码示例演示了如何使用 Stop-APSImageBuilder。

适用于 PowerShell V4 的工具

示例 1：此示例停止 ImageBuilder

```
Stop-APSIImageBuilder -Name TestImage
```

输出：

```
AccessEndpoints           : {}
AppstreamAgentVersion     : 06-19-2019
Arn                       : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime               : 1/14/2019 4:33:05 AM
Description               :
DisplayName               : TestImage
DomainJoinInfo            :
EnableDefaultInternetAccess : False
IamRoleArn                :
ImageArn                  : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors        : {}
InstanceType              : stream.standard.large
Name                     : TestImage
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                 : WINDOWS
State                    : STOPPING
StateChangeReason         :
VpcConfig                 : Amazon.AppStream.Model.VpcConfig
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [StopImageBuilder](#) 中的。

Unregister-APSFleet

以下代码示例演示了如何使用 Unregister-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例从堆栈中取消注册实例集

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DisassociateFleet](#) 中的。

Unregister-APSUserStackBatch

以下代码示例演示了如何使用 Unregister-APSUserStackBatch。

适用于 PowerShell V4 的工具

示例 1：此示例将用户从分配的堆栈中移除

```
Unregister-APSUserStackBatch -UserStackAssociation
@{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [BatchDisassociateUserStack](#) 中的。

Update-APSDirectoryConfig

以下代码示例演示了如何使用 Update-APSDirectoryConfig。

适用于 PowerShell V4 的工具

示例 1：此示例更新了在中创建的目录配置 AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
-DirectoryName contoso.com -OrganizationalUnitDistinguishedName
"OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

输出：

```
CreatedTime          DirectoryName OrganizationalUnitDistinguishedNames
ServiceAccountCredentials
-----
-----
12/27/2019 3:50:02 PM contoso.com {OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com}
Amazon.AppStream.Model.ServiceAccountCredentials
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateDirectoryConfig](#) 中的。

Update-APSFleet

以下代码示例演示了如何使用 Update-APSFleet。

适用于 PowerShell V4 的工具

示例 1：此示例更新实例集的属性

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSeconds 950
```

输出：

```
Arn                                : arn:aws:appstream:us-east-1:123456789012:fleet/
PowershellFleet
ComputeCapacityStatus             : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                       : 4/24/2019 8:39:41 AM
Description                       : PowershellFleet
DisconnectTimeoutInSeconds        : 950
DisplayName                       : PowershellFleet
DomainJoinInfo                   :
EnableDefaultInternetAccess       : True
FleetErrors                       : {}
FleetType                        : ON_DEMAND
IamRoleArn                       :
IdleDisconnectTimeoutInSeconds    : 900
ImageArn                         : arn:aws:appstream:us-east-1:123456789012:image/
Powershell
ImageName                        : Powershell
InstanceType                     : stream.standard.medium
MaxUserDurationInSeconds         : 57600
Name                             : PowershellFleet
State                            : STOPPED
VpcConfig                       : Amazon.AppStream.Model.VpcConfig
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateFleet](#) 中的。

Update-APSIImagePermission

以下代码示例演示了如何使用 Update-APSIImagePermission。

适用于 PowerShell V4 的工具

示例 1：此示例与其他账户共享 AppStream 图片

```
Update-APSIImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateImagePermissions](#) 中的。

Update-APSSStack

以下代码示例演示了如何使用 Update-APSSStack。

适用于 PowerShell V4 的工具

示例 1：此示例更新（启用）堆栈上的应用程序设置持久性和主文件夹

```
Update-APSSStack -Name PowershellStack -ApplicationSettings_Enabled $True
-ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
@{ConnectorType="HOMEFOLDERS"}
```

输出：

```
AccessEndpoints      : {}
ApplicationSettings  : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                  : arn:aws:appstream:us-east-1:123456789012:stack/PowershellStack
CreatedTime          : 4/24/2019 8:49:29 AM
Description           : PowershellStack
DisplayName           : PowershellStack
EmbedHostDomains     : {}
FeedbackURL          :
Name                  : PowershellStack
RedirectURL           :
StackErrors           : {}
StorageConnectors    : {Amazon.AppStream.Model.StorageConnector,
                        Amazon.AppStream.Model.StorageConnector}
```

```
UserSettings      : {Amazon.AppStream.Model.UserSetting,  
    Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,  
    Amazon.AppStream.Model.UserSetting}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateStack](#) 中的。

使用适用于 PowerShell V4 的工具的 Aurora 示例

以下代码示例向您展示了如何使用带有 Aurora 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-RDSOrderableDBInstanceOption

以下代码示例演示了如何使用 Get-RDSOrderableDBInstanceOption。

适用于 PowerShell V4 的工具

示例 1：此示例列出了支持 AWS 区域特定数据库实例类的数据库引擎版本。

```
$params = @{  
    Engine = 'aurora-postgresql'  
    DBInstanceClass = 'db.r5.large'  
    Region = 'us-east-1'  
}  
Get-RDSOrderableDBInstanceOption @params
```

示例 2：此示例列出了 AWS 区域特定数据库引擎版本支持的数据库实例类。


```
$params = @{  
    Engine = 'aurora-postgresql'  
    EngineVersion = '13.6'  
    Region = 'us-east-1'  
}  
Get-RDSOrderableDBInstanceOption @params
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) 中的 [DescribeOrderableDBInstance选项](#)。

使用适用于 PowerShell V4 的工具的 Auto Scaling 示例

以下代码示例向您展示了如何使用带有 Auto Scaling 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-ASLoadBalancer

以下代码示例演示了如何使用 Add-ASLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的负载均衡器附加到指定的自动扩缩组。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachLoadBalancers](#) 中的。

Complete-ASLifecycleAction

以下代码示例演示了如何使用 Complete-ASLifecycleAction。

适用于 PowerShell V4 的工具

示例 1：此示例完成了指定的生命周期操作。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -  
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CompleteLifecycleAction](#) 中的。

Disable-ASMetricsCollection

以下代码示例演示了如何使用 Disable-ASMetricsCollection。

适用于 PowerShell V4 的工具

示例 1：此示例禁用对指定自动扩缩组指定指标的监控。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",  
"GroupMaxSize")
```

示例 2：此示例禁用对指定自动扩缩组所有指标的监控。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DisableMetricsCollection](#) 中的。

Dismount-ASInstance

以下代码示例演示了如何使用 Dismount-ASInstance。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的实例与指定的自动扩缩组分离，并减少所需容量以使 Auto Scaling 不启动替换实例。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached in
                      response to a user request, shrinking
                      the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details               : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/20/2015 2:34:59 PM
StatusCode            : InProgress
StatusMessage         :
```

示例 2：此示例将在不减少所需容量的情况下将指定的实例与指定的自动扩缩组分离。Auto Scaling 会启动一个替换实例。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached in
                      response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details               : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              :
Progress              : 50
StartTime             : 11/20/2015 2:34:59 PM
StatusCode            : InProgress
StatusMessage         :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachInstances](#) 中的。

Dismount-ASLoadBalancer

以下代码示例演示了如何使用 Dismount-ASLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的负载均衡器与指定的自动扩缩组分离。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachLoadBalancers](#) 中的。

Enable-ASMetricsCollection

以下代码示例演示了如何使用 Enable-ASMetricsCollection。

适用于 PowerShell V4 的工具

示例 1：此示例启用对指定自动扩缩组指定指标的监控。

```
Enable-ASMetricsCollection -Metric @("GroupMinSize", "GroupMaxSize") -  
AutoScalingGroupName my-asg -Granularity 1Minute
```

示例 2：此示例启用对指定自动扩缩组所有指标的监控。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableMetricsCollection](#) 中的。

Enter-ASStandby

以下代码示例演示了如何使用 Enter-ASStandby。

适用于 PowerShell V4 的工具

示例 1：此示例将指定实例置于备用模式并减少所需容量，以使 Auto Scaling 不启动替换实例。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649  
AutoScalingGroupName : my-asg  
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to  
                      standby in response to a user request,  
                      shrinking the capacity from 2 to 1.  
Description          : Moving EC2 instance to Standby: i-95b8484f  
Details              : {"Availability Zone":"us-west-2b","Subnet  
                      ID":"subnet-5264e837"}  
EndTime              :  
Progress             : 50  
StartTime            : 11/22/2015 7:48:06 AM  
StatusCode           : InProgress  
StatusMessage        :
```

示例 2：此示例在不减少所需容量的情况下将指定实例置于备用模式。Auto Scaling 会启动一个替换实例。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -  
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649  
AutoScalingGroupName : my-asg  
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to  
                      standby in response to a user request.  
Description          : Moving EC2 instance to Standby: i-95b8484f  
Details              : {"Availability Zone":"us-west-2b","Subnet  
                      ID":"subnet-5264e837"}  
EndTime              :  
Progress             : 50  
StartTime            : 11/22/2015 7:48:06 AM
```

```

StatusCode      : InProgress
StatusMessage   :

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnterStandby](#) 中的。

Exit-ASStandby

以下代码示例演示了如何使用 Exit-ASStandby。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的实例移出备用模式。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

输出：

```

ActivityId      : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause           : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out of
                  standby in response to a user
                  request, increasing the capacity from 1 to 2.
Description     : Moving EC2 instance out of Standby: i-95b8484f
Details         : {"Availability Zone":"us-west-2b","Subnet
                  ID":"subnet-5264e837"}
EndTime        :
Progress        : 30
StartTime       : 11/22/2015 7:51:21 AM
StatusCode      : PreInService
StatusMessage   :

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ExitStandby](#) 中的。

Get-ASAccountLimit

以下代码示例演示了如何使用 Get-ASAccountLimit。

适用于 PowerShell V4 的工具

示例 1：此示例描述了您的 AWS 账户的 Auto Scaling 资源限制。

```
Get-ASAccountLimit
```

输出：

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAccountLimits](#) 中的。

Get-ASAdjustmentType

以下代码示例演示了如何使用 Get-ASAdjustmentType。

适用于 PowerShell V4 的工具

示例 1：此示例描述 Auto Scaling 所支持的调整类型。

```
Get-ASAdjustmentType
```

输出：

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAdjustmentTypes](#) 中的。

Get-ASAutoScalingGroup

以下代码示例演示了如何使用 Get-ASAutoScalingGroup。

适用于 PowerShell V4 的工具

示例 1：此示例列出自动扩缩组的名称。

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

输出：

```
AutoScalingGroupName
-----
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

示例 2：此示例描述指定的自动扩缩组。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

输出：

```
AutoScalingGroupARN      : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                           f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName     : my-asg-1
AvailabilityZones        : {us-west-2b, us-west-2a}
CreatedTime              : 3/1/2015 9:05:31 AM
DefaultCooldown          : 300
DesiredCapacity          : 2
EnabledMetrics           : {}
HealthCheckGracePeriod   : 300
HealthCheckType          : EC2
Instances                : {my-1c}
LaunchConfigurationName  : my-1c
LoadBalancerNames        : {}
MaxSize                  : 0
MinSize                  : 0
PlacementGroup           :
Status                   :
SuspendedProcesses       : {}
Tags                     : {}
TerminationPolicies      : {Default}
VPCZoneIdentifier        : subnet-e4f33493,subnet-5264e837
```


示例 3：此示例描述指定的两个自动扩缩组。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @"my-asg-1", "my-asg-2")
```

示例 4：此示例描述指定自动扩缩组的 Auto Scaling 实例。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

示例 5：此示例描述所有自动扩缩组。

```
Get-ASAutoScalingGroup
```

示例 6：此示例描述了指定 LaunchTemplate 的 Auto Scaling 组。此示例假设“实例购买选项”设置为“遵照启动模板”。如果此选项设置为“合并购买选项和实例类型”，则 LaunchTemplate 可以使用“进行访问MixedInstancesPolicy。LaunchTemplate“财产。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

输出：

LaunchTemplateId	LaunchTemplateName	Version
-----	-----	-----
lt-06095fd619cb40371	test-launch-template	\$Default

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAutoScalingGroups](#) 中的。

Get-ASAutoScalingInstance

以下代码示例演示了如何使用 Get-ASAutoScalingInstance。

适用于 PowerShell V4 的工具

示例 1：此示例列出了您的 IDs Auto Scaling 实例。

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

输出：

```
InstanceId
-----
i-12345678
i-87654321
i-abcd1234
```

示例 2：此示例描述指定的 Auto Scaling 实例。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

输出：

```
AutoScalingGroupName      : my-asg
AvailabilityZone           : us-west-2b
HealthStatus              : HEALTHY
InstanceId                 : i-12345678
LaunchConfigurationName   : my-lc
LifecycleState             : InService
```

示例 3：此示例描述指定的两个 Auto Scaling 实例。

```
Get-ASAutoScalingInstance -InstanceId @( "i-12345678", "i-87654321" )
```

示例 4：此示例描述指定自动扩缩组的 Auto Scaling 实例。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

示例 5：此示例描述所有 Auto Scaling 实例。

```
Get-ASAutoScalingInstance
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAutoScalingInstances](#) 中的。

Get-ASAutoScalingNotificationType

以下代码示例演示了如何使用 Get-ASAutoScalingNotificationType。

适用于 PowerShell V4 的工具

示例 1：此示例列出 Auto Scaling 支持的通知类型。

```
Get-ASAutoScalingNotificationType
```

输出：

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAutoScalingNotificationTypes](#) 中的。

Get-ASLaunchConfiguration

以下代码示例演示了如何使用 Get-ASLaunchConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例列出启动配置的名称。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

输出：

```
LaunchConfigurationName
-----
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

示例 2：此示例描述指定的启动配置。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

输出：

```
AssociatePublicIpAddress      : True
BlockDeviceMappings           : {/dev/xvda}
ClassicLinkVPCId              :
ClassicLinkVPCSecurityGroups  : {}
CreatedTime                   : 12/12/2014 3:22:08 PM
EbsOptimized                  : False
IamInstanceProfile            :
ImageId                       : ami-043a5034
InstanceMonitoring             : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                  : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN        : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                              e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName       : my-lc-1
PlacementTenancy              :
RamdiskId                    :
SecurityGroups                : {sg-67ef0308}
SpotPrice                    :
UserData                      :
```

示例 3：此示例描述指定的两种启动配置。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

示例 4：此示例描述所有启动配置。

```
Get-ASLaunchConfiguration
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeLaunchConfigurations](#) 中的。

Get-ASLifecycleHook

以下代码示例演示了如何使用 Get-ASLifecycleHook。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的生命周期挂钩。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

输出：

```
AutoScalingGroupName : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition    : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

示例 2：此示例描述指定自动扩缩组的所有生命周期挂钩。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

示例 3：此示例描述所有自动扩缩组的所有生命周期挂钩。

```
Get-ASLifecycleHook
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeLifecycleHooks](#) 中的。

Get-ASLifecycleHookType

以下代码示例演示了如何使用 Get-ASLifecycleHookType。

适用于 PowerShell V4 的工具

示例 1：此示例列出 Auto Scaling 支持的生命周期挂钩类型。

```
Get-ASLifecycleHookType
```

输出：

```
autoscaling:EC2_INSTANCE_LAUNCHING
```

```
auto-scaling:EC2_INSTANCE_TERMINATING
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeLifecycleHookTypes](#) 中的。

Get-ASLoadBalancer

以下代码示例演示了如何使用 Get-ASLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定自动扩缩组的负载均衡器。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

输出：

LoadBalancerName	State
-----	----
my-lb	Added

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeLoadBalancers](#) 中的。

Get-ASMetricCollectionType

以下代码示例演示了如何使用 Get-ASMetricCollectionType。

适用于 PowerShell V4 的工具

示例 1：此示例列出 Auto Scaling 支持的指标收集类型。

```
(Get-ASMetricCollectionType).Metrics
```

输出：

Metric

GroupMinSize

```

GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances

```

示例 2：此示例列出相应的粒度。

```
(Get-ASMetricCollectionType).Granularities
```

输出：

```

Granularity
-----
1Minute

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMetricCollectionTypes](#) 中的。

Get-ASNotificationConfiguration

以下代码示例演示了如何使用 Get-ASNotificationConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例描述与指定自动扩缩组关联的通知操作。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

输出：

```

AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType      : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN              : arn:aws:sns:us-west-2:123456789012:my-topic

```

示例 2：此示例描述与所有自动扩缩组关联的通知操作。

```
Get-ASNotificationConfiguration
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeNotificationConfigurations](#) 中的。

Get-ASPolicy

以下代码示例演示了如何使用 Get-ASPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定自动扩缩组的所有策略。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

输出：

```
AdjustmentType      : ChangeInCapacity
Alarms              : {}
AutoScalingGroupName : my-asg
Cooldown            : 0
EstimatedInstanceWarmup : 0
MetricAggregationType :
MinAdjustmentMagnitude : 0
MinAdjustmentStep    : 0
PolicyARN           : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                    : autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName          : myScaleInPolicy
PolicyType          : SimpleScaling
ScalingAdjustment    : -1
StepAdjustments      : {}
```

示例 2：此示例描述指定自动扩缩组的指定策略。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
"myScaleInPolicy")
```


示例 3：此示例描述所有自动扩缩组的所有策略。

```
Get-ASPolicy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribePolicies](#) 中的。

Get-ASScalingActivity

以下代码示例演示了如何使用 Get-ASScalingActivity。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定自动扩缩组过去六周的扩缩活动。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

输出：

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set group
                      desired capacity changing the desired
                      capacity from 1 to 2. At 2015-11-22T15:45:34Z an instance
                      was started in response to a difference
                      between desired and actual capacity, increasing the capacity
                      from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress              : 100
StartTime             : 11/22/2015 7:45:35 AM
StatusCode            : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
                      AutoScalingGroup changing the desired capacity
                      from 0 to 1. At 2015-11-20T22:57:58Z an instance was
                      started in response to a difference betwe
```

```

        en desired and actual capacity, increasing the capacity from
    0 to 1.
Description      : Launching a new EC2 instance: i-93633f9b
Details          : {"Availability Zone":"us-west-2b","Subnet
    ID":"subnet-5264e837"}
EndTime         : 11/20/2015 2:58:32 PM
Progress        : 100
StartTime       : 11/20/2015 2:57:59 PM
StatusCode      : Successful
StatusMessage   :

```

示例 2：此示例描述指定的扩缩活动。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

示例 3：此示例描述所有自动扩缩组过去六周的扩缩活动。

```
Get-ASScalingActivity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeScalingActivities](#) 中的。

Get-ASScalingProcessType

以下代码示例演示了如何使用 Get-ASScalingProcessType。

适用于 PowerShell V4 的工具

示例 1：此示例列出 Auto Scaling 支持的进程类型。

```
Get-ASScalingProcessType
```

输出：

```

ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck

```

```
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScalingProcessTypes](#) 中的。

Get-ASScheduledAction

以下代码示例演示了如何使用 Get-ASScheduledAction。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定自动扩缩组的计划扩缩操作。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

输出：

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence            :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
2c3af3a4d6:autoScalingGroupName/my-asg:scheduledActionName/
myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

示例 2：此示例描述指定的计划扩缩操作。

```
Get-ASScheduledAction -ScheduledActionName @"(myScheduledScaleOut",
"myScheduledScaleIn")
```

示例 3：此示例描述在指定时间开始的计划扩缩操作。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

示例 4：此示例描述在指定时间结束的计划扩缩操作。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

示例 5：此示例描述所有自动扩缩组的计划扩缩操作。

```
Get-ASScheduledAction
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeScheduledActions](#) 中的。

Get-ASTag

以下代码示例演示了如何使用 Get-ASTag。

适用于 PowerShell V4 的工具

示例 1：此示例描述键值为“myTag”或“myTag2”的标签。筛选器名称的可能值为“、auto-scaling-group 'key'、'value' 和 'propagate-at-launch'”。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2")} )
```

输出：

```
Key           : myTag2
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value          : myTagValue2

Key           : myTag
PropagateAtLaunch : True
ResourceId     : my-asg
ResourceType   : auto-scaling-group
Value          : myTagValue
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 为过滤器参数创建过滤器。

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

示例 3：此示例描述所有自动扩缩组的所有标签。

```
Get-ASTag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

Get-ASTerminationPolicyType

以下代码示例演示了如何使用 Get-ASTerminationPolicyType。

适用于 PowerShell V4 的工具

示例 1：此示例列出 Auto Scaling 支持的终止策略。

```
Get-ASTerminationPolicyType
```

输出：

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTerminationPolicyTypes](#) 中的。

Mount-ASInstance

以下代码示例演示了如何使用 Mount-ASInstance。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的实例附加到指定的自动扩缩组。Auto Scaling 会自动增加自动扩缩组的所需容量。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachInstances](#) 中的。

New-ASAutoScalingGroup

以下代码示例演示了如何使用 New-ASAutoScalingGroup。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的名称和属性创建自动扩缩组。默认所需容量为最小大小。因此，此自动扩缩组启动两个实例，指定的两个可用区中各一个。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -  
MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateAutoScalingGroup](#) 中的。

New-ASLaunchConfiguration

以下代码示例演示了如何使用 New-ASLaunchConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为“my-lc”的启动配置。使用此启动配置的 Auto Scaling 组启动的 EC2 实例使用指定的实例类型、AMI、安全组 and IAM 角色。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -  
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateLaunchConfiguration](#) 中的。

Remove-ASAutoScalingGroup

以下代码示例演示了如何使用 Remove-ASAutoScalingGroup。

适用于 PowerShell V4 的工具

示例 1：如果指定的自动扩缩组没有正在运行的实例，则此示例将删除该组。在操作继续之前，系统会提示您进行确认。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on Target
"my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

示例 3：此示例删除指定的自动扩缩组并终止该组包含的所有正在运行的实例。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteAutoScalingGroup](#) 中的。

Remove-ASLaunchConfiguration

以下代码示例演示了如何使用 Remove-ASLaunchConfiguration。

适用于 PowerShell V4 的工具

示例 1：如果指定的启动配置未附加到自动扩缩组，则此示例将删除该配置。在操作继续之前，系统会提示您进行确认。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)" on
Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteLaunchConfiguration](#) 中的。

Remove-ASLifecycleHook

以下代码示例演示了如何使用 Remove-ASLifecycleHook。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定自动扩缩组的指定生命周期挂钩。在操作继续之前，系统会提示您进行确认。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
myLifecycleHook
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
"myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```


示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteLifecycleHook](#) 中的。

Remove-ASNotificationConfiguration

以下代码示例演示了如何使用 Remove-ASNotificationConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的通知操作。在操作继续之前，系统会提示您进行确认。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic"
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ASNotificationConfiguration  
(DeleteNotificationConfiguration)" on Target  
"arn:aws:sns:us-west-2:123456789012:my-topic".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN  
"arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteNotificationConfiguration](#) 中的。

Remove-ASPolicy

以下代码示例演示了如何使用 Remove-ASPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定自动扩缩组的指定策略。在操作继续之前，系统会提示您进行确认。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target "myScaleInPolicy".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeletePolicy](#) 中的。

Remove-ASScheduledAction

以下代码示例演示了如何使用 Remove-ASScheduledAction。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定自动扩缩组的指定计划操作。在操作继续之前，系统会提示您进行确认。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
"myScheduledAction"
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
"myScheduledAction".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
"myScheduledAction" -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteScheduledAction](#) 中的。

Remove-ASTag

以下代码示例演示了如何使用 Remove-ASTag。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的自动扩缩组删除指定的标签。在操作继续之前，系统会提示您进行确认。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
Key="myTag" } )
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
"Amazon.AutoScaling.Model.Tag".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

示例 2：如果您指定 Force 参数，则在操作继续之前，系统不会提示您进行确认。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
Key="myTag" } ) -Force
```

示例 3：对于 PowerShell 版本 2，必须使用 New-Object 创建 Tag 参数的标签。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTags](#) 中的。

Resume-ASProcess

以下代码示例演示了如何使用 Resume-ASProcess。

适用于 PowerShell V4 的工具

示例 1：此示例恢复指定自动扩缩组的指定 Auto Scaling 进程。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

示例 2：此示例恢复指定自动扩缩组的所有暂停 Auto Scaling 进程。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResumeProcesses](#) 中的。

Set-ASDesiredCapacity

以下代码示例演示了如何使用 Set-ASDesiredCapacity。

适用于 PowerShell V4 的工具

示例 1：此示例设置指定自动扩缩组的大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

示例 2：此示例设置指定自动扩缩组的大小，并等待冷却时间结束后再扩缩到新大小。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SetDesiredCapacity](#) 中的。

Set-ASInstanceHealth

以下代码示例演示了如何使用 Set-ASInstanceHealth。

适用于 PowerShell V4 的工具

示例 1：此示例将指定实例的状态设置为“运行状况不佳”，使其停止服务。Auto Scaling 会终止并替换该实例。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

示例 2：此示例将指定实例的状态设置为“运行状况正常”，使其保持服务状态。未遵守自动扩缩组的任何运行状况检查宽限期。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetInstanceHealth](#) 中的。

Set-ASInstanceProtection

以下代码示例演示了如何使用 Set-ASInstanceProtection。

适用于 PowerShell V4 的工具

示例 1：此示例启用指定实例的实例保护。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

示例 2：此示例禁用指定实例的实例保护。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetInstanceProtection](#) 中的。

Set-ASTag

以下代码示例演示了如何使用 Set-ASTag。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的自动扩缩组添加单个标签。标签键为“myTag”，标签值为“myTagValue”。Auto Scaling 会将此标签传播到由 Auto Scaling 组启动的后续 EC2 实例。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 为标签参数创建标签。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag  
$tag.ResourceType = "auto-scaling-group"  
$tag.ResourceId = "my-asg"  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
$tag.PropagateAtLaunch = $true  
Set-ASTag -Tag $tag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateOrUpdateTags](#) 中的。

Start-ASPolicy

以下代码示例演示了如何使用 Start-ASPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例对指定的自动扩缩组执行指定的策略。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

示例 2：此示例在等待冷却时间结束后，对指定的自动扩缩组执行指定的策略。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -  
HonorCooldown $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ExecutePolicy](#) 中的。

Stop-ASInstanceInAutoScalingGroup

以下代码示例演示了如何使用 Stop-ASInstanceInAutoScalingGroup。

适用于 PowerShell V4 的工具

示例 1：此示例终止了指定的实例，并减少其自动扩缩组的所需容量，以使 Auto Scaling 不启动替换实例。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $true
```

输出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of  
                      service in response to a user  
                      request, shrinking the capacity from 2 to 1.  
Description          : Terminating EC2 instance: i-93633f9b  
Details              : {"Availability Zone":"us-west-2b","Subnet  
                      ID":"subnet-5264e837"}  
EndTime              :  
Progress              : 0  
StartTime             : 11/22/2015 8:09:03 AM  
StatusCode            : InProgress  
StatusMessage        :
```

示例 2：此示例在不减少指定实例自动扩缩组所需容量的情况下终止该实例。Auto Scaling 会启动一个替换实例。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $false
```

输出：

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of  
                      service in response to a user
```

```

request.
Description      : Terminating EC2 instance: i-93633f9b
Details          : {"Availability Zone":"us-west-2b","Subnet
ID":"subnet-5264e837"}
EndTime         :
Progress        : 0
StartTime       : 11/22/2015 8:09:03 AM
StatusCode      : InProgress
StatusMessage   :

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [TerminateInstanceInAutoScalingGroup](#) 中的。

Suspend-ASProcess

以下代码示例演示了如何使用 Suspend-ASProcess。

适用于 PowerShell V4 的工具

示例 1：此示例暂停了指定自动扩缩组的指定 Auto Scaling 进程。

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

示例 2：此示例暂停了指定自动扩缩组的所有 Auto Scaling 进程。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SuspendProcesses](#) 中的。

Update-ASAutoScalingGroup

以下代码示例演示了如何使用 Update-ASAutoScalingGroup。

适用于 PowerShell V4 的工具

示例 1：此示例更新指定自动扩缩组的最小和最大大小。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```


示例 2：此示例更新指定自动扩缩组的默认冷却时间。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

示例 3：此示例更新指定自动扩缩组的可用区。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

示例 4：此示例更新指定的自动扩缩组以使用 Elastic Load Balancing 运行状况检查。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -  
HealthCheckGracePeriod 60
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateAutoScalingGroup](#) 中的。

Write-ASLifecycleActionHeartbeat

以下代码示例演示了如何使用 Write-ASLifecycleActionHeartbeat。

适用于 PowerShell V4 的工具

示例 1：此示例记录指定生命周期操作的心跳。这会使实例保持待处理状态，直到您完成自定义操作。

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName  
myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RecordLifecycleActionHeartbeat](#) 中的。

Write-ASLifecycleHook

以下代码示例演示了如何使用 Write-ASLifecycleHook。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的生命周期挂钩添加到指定的自动扩缩组。

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
"myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -  
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN  
"arn:aws:iam::123456789012:role/my-iam-role"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutLifecycleHook](#) 中的。

Write-ASNotificationConfiguration

以下代码示例演示了如何使用 Write-ASNotificationConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的 Auto Scaling 组配置为在启动 EC2 实例时向指定的 SNS 主题发送通知。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
"autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-  
topic"
```

示例 2：此示例将指定的 Auto Scaling 组配置为在启动或终止 EC2 实例时向指定的 SNS 主题发送通知。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
@("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -  
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutNotificationConfiguration](#) 中的。

Write-ASScalingPolicy

以下代码示例演示了如何使用 Write-ASScalingPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的策略添加到指定的自动扩缩组。指定的调整类型决定了如何解释 ScalingAdjustment 参数。使用 “ChangeInCapacity” 时，正值将按指定数量的实例增加容量，负值则按指定的实例数量减少容量。

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType  
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

输出：

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-  
e1d769fc24ef:autoScalingGroupName/my-asg  
:policyName/myScaleInPolicy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutScalingPolicy](#) 中的。

Write-ASScheduledUpdateGroupAction

以下代码示例演示了如何使用 Write-ASScheduledUpdateGroupAction。

适用于 PowerShell V4 的工具

示例 1：此示例创建或更新一次性计划操作，以在指定的开始时间更改所需容量。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName  
"myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutScheduledUpdateGroupAction](#) 中的。

AWS Budgets 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS Budgets 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

New-BGTBudget

以下代码示例演示了如何使用 New-BGTBudget。

适用于 PowerShell V4 的工具

示例 1：使用电子邮件通知创建具有指定预算和时间限制的新预算。

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @"user@domain.com"
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateBudget](#) 中的。

AWS Cloud9 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS Cloud9 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-C9EnvironmentData

以下代码示例演示了如何使用 Get-C9EnvironmentData。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定 AWS Cloud9 开发环境的信息。

```
Get-C9EnvironmentData -EnvironmentId
685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

输出：

```
Arn          : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description  : Created from CodeStar.
Id           : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle    : Amazon.Cloud9.Model.EnvironmentLifecycle
Name         : my-demo-ec2-env
OwnerArn     : arn:aws:iam::123456789012:user/MyDemoUser
Type         : ec2

Arn          : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description  :
Id           : 1980b80e5f584920801c09086667f0EX
Lifecycle    : Amazon.Cloud9.Model.EnvironmentLifecycle
Name         : my-demo-ssh-env
OwnerArn     : arn:aws:iam::123456789012:user/MyDemoUser
Type         : ssh
```

示例 2：此示例获取有关指定 AWS Cloud9 开发环境生命周期状态的信息。

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

输出：

```
FailureResource Reason Status
-----
CREATED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeEnvironments](#) 中的。

Get-C9EnvironmentList

以下代码示例演示了如何使用 Get-C9EnvironmentList。

适用于 PowerShell V4 的工具

示例 1：此示例获取可用的 AWS Cloud9 开发环境标识符列表。

```
Get-C9EnvironmentList
```

输出：

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListEnvironments](#) 中的。

Get-C9EnvironmentMembershipList

以下代码示例演示了如何使用 Get-C9EnvironmentMembershipList。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定 AWS Cloud9 开发环境的环境成员的信息。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeaeaa8a04bfde8cEX
```

输出：

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions    : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCXHEX

EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions    : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

示例 2：此示例获取有关指定 AWS Cloud9 开发环境所有者的信息。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -
Permission owner
```

输出：

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions    : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

示例 3：此示例获取有关多个 AWS Cloud9 开发环境的指定环境成员的信息。

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

输出：

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions    : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
```

```
Permissions      : owner
UserArn          : arn:aws:iam::123456789012:user/MyDemoUser
UserId           : AIDAJ3LOROMOUXTBSUGEX
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeEnvironmentMemberships](#) 中的。

Get-C9EnvironmentStatus

以下代码示例演示了如何使用 Get-C9EnvironmentStatus。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定 AWS Cloud9 开发环境的状态信息。

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

输出：

```
Message                Status
-----
Environment is ready to use ready
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeEnvironmentStatus](#) 中的。

New-C9EnvironmentEC2

以下代码示例演示了如何使用 New-C9EnvironmentEC2。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定设置创建一个 AWS Cloud9 开发环境，启动亚马逊弹性计算云 (Amazon EC2) 实例，然后从该实例连接到该环境。

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
"My demonstration development environment." -InstanceType t2.micro -OwnerArn
arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

输出：


```
ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的 [CreateEnvironmentEc2](#)。

New-C9EnvironmentMembership

以下代码示例演示了如何使用 New-C9EnvironmentMembership。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的环境成员添加到指定的 AWS Cloud9 开发环境中。

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser  
-EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Permission read-write
```

输出：

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX  
LastAccess    : 1/1/0001 12:00:00 AM  
Permissions    : read-write  
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser  
UserId        : AIDAJ3BA602FMJWCWXHEX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateEnvironmentMembership](#) 中的。

Remove-C9Environment

以下代码示例演示了如何使用 Remove-C9Environment。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定的 AWS Cloud9 开发环境。如果 Amazon EC2 实例已连接到环境，也会终止该实例。

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteEnvironment](#) 中的。

Remove-C9EnvironmentMembership

以下代码示例演示了如何使用 Remove-C9EnvironmentMembership。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的 AWS Cloud9 开发环境中删除指定的环境成员。

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteEnvironmentMembership](#) 中的。

Update-C9Environment

以下代码示例演示了如何使用 Update-C9Environment。

适用于 PowerShell V4 的工具

示例 1：此示例更改了指定现有 AWS Cloud9 开发环境的指定设置。

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Description
"My changed demonstration development environment." -Name my-changed-demo-env
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateEnvironment](#) 中的。

Update-C9EnvironmentMembership

以下代码示例演示了如何使用 Update-C9EnvironmentMembership。

适用于 PowerShell V4 的工具

示例 1：此示例更改了指定 AWS Cloud9 开发环境的指定现有环境成员的设置。

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Permission read-
only
```

输出：

```
EnvironmentId : ffd88420d4824eeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCWXHEX
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateEnvironmentMembership](#) 中的。

CloudFormation 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CloudFormation 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-CFNStack

以下代码示例演示了如何使用 Get-CFNStack。

适用于 PowerShell V4 的工具

示例 1：返回描述用户的所有堆栈的堆栈实例集合。

```
Get-CFNStack
```

示例 2：返回描述指定堆栈的堆栈实例

```
Get-CFNStack -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeStacks](#) 中的。

Get-CFNStackEvent

以下代码示例演示了如何使用 Get-CFNStackEvent。

适用于 PowerShell V4 的工具

示例 1：返回指定堆栈的所有堆栈相关事件。

```
Get-CFNStackEvent -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeStackEvents](#) 中的。

Get-CFNStackResource

以下代码示例演示了如何使用 Get-CFNStackResource。

适用于 PowerShell V4 的工具

示例 1：返回模板中按逻辑 ID DBInstance “我的” 标识的与指定堆栈关联的资源的描述。

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeStackResource](#) 中的。

Get-CFNStackResourceList

以下代码示例演示了如何使用 Get-CFNStackResourceList。

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的多达 100 个资源的 AWS 资源描述。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，它还支持手动分页结果。

```
Get-CFNStackResourceList -StackName "myStack"
```

示例 2：返回在模板中通过逻辑 ID “Ec2Instance” 标识的与指定堆栈关联的亚马逊 EC2 实例的描述。

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

示例 3：返回与堆栈关联的多达 100 个资源的描述，该堆栈包含一个由 EC2 实例 ID “i-123456” 标识的 Amazon 实例。要获取与堆栈关联的所有资源的详细信息，请使用 Get-CFNStackResourceSummary，它还支持手动分页结果。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

示例 4：返回由堆栈模板中的逻辑 ID “Ec2Instance” 标识的 Amazon EC2 实例的描述。堆栈使用其包含的资源的物理资源 ID 进行标识，在本例中也是 EC2 实例 ID 为 “i-123456” 的 Amazon 实例。也可以根据模板内容使用其他物理资源来识别堆栈，例如 Amazon S3 存储桶。

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId  
"Ec2Instance"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeStackResources](#) 中的。

Get-CFNStackResourceSummary

以下代码示例演示了如何使用 Get-CFNStackResourceSummary。

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的所有资源的描述。

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListStackResources](#) 中的。

Get-CFNStackSummary

以下代码示例演示了如何使用 Get-CFNStackSummary。

适用于 PowerShell V4 的工具

示例 1：返回所有堆栈的摘要信息。

```
Get-CFNStackSummary
```

示例 2：返回当前正在创建的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

示例 3：返回当前正在创建或更新的所有堆栈的摘要信息。

```
Get-CFNStackSummary -StackStatusFilter @"CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListStacks](#) 中的。

Get-CFNTemplate

以下代码示例演示了如何使用 Get-CFNTemplate。

适用于 PowerShell V4 的工具

示例 1：返回与指定堆栈关联的模板。

```
Get-CFNTemplate -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetTemplate](#) 中的。

Measure-CFNTemplateCost

以下代码示例演示了如何使用 Measure-CFNTemplateCost。

适用于 PowerShell V4 的工具

示例 1：返回一个带有查询字符串的 AWS 简单月度计算器网址，该字符串描述了运行模板所需的资源。模板是从指定的 Amazon S3 URL 获取的，并且应用了单个自定义参数。也可以使用“键”和“值”来指定参数，而不是“ParameterKey”和“ParameterValue”。

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/  
templatefile.template `
```

```
-Region us-west-1 `
-Parameter @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }
```

示例 2：返回一个带有查询字符串的 AWS 简单月度结算器网址，该字符串描述了运行模板所需的资源。模板是根据提供的内容和应用的自定义参数进行解析的（此示例假设模板内容会声明两个参数，'和KeyName' InstanceType'）。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
-Parameter @( @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }, `
@{ ParameterKey="InstanceType";
ParameterValue="m1.large" })
```

示例 3：使用 New-Object 生成模板参数集，并返回一个“AWS 简单月度计算器”网址，其中包含描述运行模板所需资源的查询字符串。模板是从提供的内容中解析出来的，并带有自定义参数（此示例假设模板内容会声明两个参数，KeyName'和' InstanceType'）。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter @( $p1,
$p2 )
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EstimateTemplateCost](#) 中的。

New-CFNStack

以下代码示例演示了如何使用 New-CFNStack。

适用于 PowerShell V4 的工具

示例 1：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，带有自定义参数（PK1'PK2' 和 " 代表模板内容中声明的参数名称，PV1'和' PV2 '代表这些参数的值。也可以使用

“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则不会将其回滚。

```
New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
    -DisableRollback $true
```

示例 2：使用指定名称创建新堆栈。模板是从提供的内容中解析出来的，带有自定义参数（PK1'PK2' 和 " 代表模板内容中声明的参数的名称，PV1'和' PV2 '代表这些参数的值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚。

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( $p1, $p2 ) `
    -OnFailure "ROLLBACK"
```

示例 3：使用指定名称创建新堆栈。该模板是从带有自定义参数的 Amazon S3 URL 中获取的（PK1"" 表示模板内容中声明的参数的名称，PV1"" 表示参数的值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定相同-DisableRollback \$false）。

```
New-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用指定名称创建新堆栈。该模板是从带有自定义参数的 Amazon S3 URL 中获取的（PK1"" 表示模板内容中声明的参数的名称，PV1"" 表示参数的值。也可以使用“密钥”和“值”来指

定自定义参数，而不是“ParameterKey”和“ParameterValue”。如果堆栈创建失败，则会将其回滚（与指定相同-DisableRollback \$false）。指定的通知 AENs 将收到已发布的堆栈相关事件。

```
New-CFNStack -StackName "myStack" `
              -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
              templatefile.template `
              -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
              -NotificationARN @( "arn1", "arn2" )
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateStack](#) 中的。

Remove-CFNStack

以下代码示例演示了如何使用 Remove-CFNStack。

适用于 PowerShell V4 的工具

示例 1：删除指定的堆栈。

```
Remove-CFNStack -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteStack](#) 中的。

Resume-CFNUpdateRollback

以下代码示例演示了如何使用 Resume-CFNUpdateRollback。

适用于 PowerShell V4 的工具

示例 1：继续指定堆栈的回滚，该堆栈的状态应为“UPDATE_ROLLBACK_FAILED”。如果继续回滚成功，堆栈将进入“UPDATE_ROLLBACK_COMPLETE”状态。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ContinueUpdateRollback](#) 中的。

Stop-CFNUpdateStack

以下代码示例演示了如何使用 Stop-CFNUpdateStack。

适用于 PowerShell V4 的工具

示例 1：取消指定堆栈的更新。

```
Stop-CFNUpdateStack -StackName "myStack"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CancelUpdateStack](#) 中的。

Test-CFNStack

以下代码示例演示了如何使用 Test-CFNStack。

适用于 PowerShell V4 的工具

示例 1：测试堆栈是否已达到 UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE 或 UPDATE_COMPLETE 状态之一。

```
Test-CFNStack -StackName MyStack
```

输出：

```
False
```

示例 2：测试堆栈是否已达到 UPDATE_COMPLETE 或 UPDATE_ROLLBACK_COMPLETE 的状态。

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet CFNStack 中的 [测试](#) 参考 (V4)。

Test-CFNTemplate

以下代码示例演示了如何使用 Test-CFNTemplate。

适用于 PowerShell V4 的工具

示例 1：验证指定模板的内容。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

示例 2：验证通过 Amazon S3 URL 访问的指定模板。输出详细说明了模板的功能、描述和参数。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ValidateTemplate](#) 中的。

Update-CFNStack

以下代码示例演示了如何使用 Update-CFNStack。

适用于 PowerShell V4 的工具

示例 1：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”表示模板中声明的参数的名称，“PV1”表示其值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 2：使用指定的模板和自定义参数更新堆栈“MyStack”。PK1'和' PK2 '表示模板中声明的参数的名称，'和 PV1 'PV2' 代表其请求的值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                              @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 3：使用指定的模板和自定义参数更新堆栈“MyStack”。“PK1”表示模板中声明的参数的名称，“PV2”表示其值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

示例 4：使用从 Amazon S3 获得的指定模板和自定义参数更新堆栈 “MyStack”。PK1'和' PK2 '表示模板中声明的参数的名称，' 和 PV1 'PV2' 代表其请求的值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

示例 5：使用从 Amazon S3 获取的指定模板和自定义参数更新堆栈 “MyStack”（在本示例中假设该堆栈包含 IAM 资源）。PK1'和' PK2 '表示模板中声明的参数的名称，' 和 PV1 'PV2' 代表其请求的值。也可以使用“密钥”和“值”来指定自定义参数，而不是“ParameterKey”和“ParameterValue”。包含 IAM 资源的堆栈要求您指定-Capabilities “CAPABILITY_IAM” 参数，否则更新将失败并出现“”错误。InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateStack](#) 中的。

Wait-CFNStack

以下代码示例演示了如何使用 Wait-CFNStack。

适用于 PowerShell V4 的工具

示例 1：测试堆栈是否已达到

UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE 或 UPDATE_COMPLETE 状态之一。如果堆栈未处于其中一种状态，则命令将休眠两秒钟，然后再次测试状态。重复此操作，直到堆栈达到所请求的状态之一或经过了 60 秒的默认超时时间。如果

超过了超时时间，则会引发异常。如果堆栈在超时时间内达到请求的状态之一，则会将它返回到管道。

```
$stack = Wait-CFNStack -StackName MyStack
```

示例 2：此示例等待总共 5 分钟（300 秒），以使堆栈达到任一指定的状态。在此示例中，在超时之前达到此状态，因此堆栈对象将返回到管道。

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status  
CREATE_COMPLETE,ROLLBACK_COMPLETE
```

输出：

```
Capabilities      : {CAPABILITY_IAM}  
ChangeSetId       :  
CreationTime      : 6/1/2017 9:29:33 AM  
Description       : AWS CloudFormation Sample Template  
ec2_instance_with_instance_profile: Create an EC2 instance with an associated  
instance profile. **WARNING** This template creates one or more Amazon EC2  
instances and an Amazon SQS queue. You will be billed for the  
AWS resources used if you create a stack from this template.  
DisableRollback   : False  
LastUpdatedTime   : 1/1/0001 12:00:00 AM  
NotificationARNs  : {}  
Outputs           : {}  
Parameters        : {}  
RoleARN           :  
StackId           : arn:aws:cloudformation:us-west-2:123456789012:stack/  
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1  
StackName         : MyStack  
StackStatus       : CREATE_COMPLETE  
StackStatusReason :  
Tags              : {}  
TimeoutInMinutes  : 0
```

示例 3：此示例演示堆栈在超时时间（在本例中为默认时段 60 秒）内未达到所请求状态之一的错误输出。

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

输出：

```
Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
stack MyStack in region us-west-2 to reach one of state(s):
UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~
+ CategoryInfo          : InvalidOperation:
(Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet) [Wait-CFNStack],
InvalidOperationException
+ FullyQualifiedErrorId :
InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet
```

- 有关 API 的详细信息，请参阅[等待AWS Tools for PowerShell Cmdlet 参考 \(V4\)](#)。CFNStack

CloudFront 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CloudFront 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-CFCloudFrontOriginAccessIdentity

以下代码示例演示了如何使用 Get-CFCloudFrontOriginAccessIdentity。

适用于 PowerShell V4 的工具

示例 1：此示例返回由 -Id 参数指定的特定亚马逊 CloudFront 源访问身份。尽管不需要 -Id 参数，但如果您不指定该参数，则不会返回任何结果。

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

输出：

```
CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
-----
Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetCloudFrontOriginAccessIdentity](#) 中的。

Get-CFCloudFrontOriginAccessIdentityConfig

以下代码示例演示了如何使用 Get-CFCloudFrontOriginAccessIdentityConfig。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关单个 Amazon CloudFront 源访问身份的配置信息，该身份由 -Id 参数指定。如果未指定 -Id 参数，则会出现错误。

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

输出：

```
CallerReference                                Comment
-----
mycallerreference: 2/1/2011 1:16:32 PM          Caller reference:
2/1/2011 1:16:32 PM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetCloudFrontOriginAccessIdentityConfig](#) 中的。

Get-CFCloudFrontOriginAccessIdentityList

以下代码示例演示了如何使用 Get-CFCloudFrontOriginAccessIdentityList。

适用于 PowerShell V4 的工具

示例 1：此示例返回 Amazon CloudFront 来源访问身份列表。由于 -MaxItem 参数指定的值为 2，因此结果包含两个恒等式。

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

输出：

```
IsTruncated : True
Items       : {E326XXXXXXXXXXT, E1YWXXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXXX9B
Quantity    : 2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListCloudFrontOriginAccessIdentities](#) 中的。

Get-CFDistribution

以下代码示例演示了如何使用 Get-CFDistribution。

适用于 PowerShell V4 的工具

示例 1：检索有关特定分配的信息。

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDistribution](#) 中的。

Get-CFDistributionConfig

以下代码示例演示了如何使用 Get-CFDistributionConfig。

适用于 PowerShell V4 的工具

示例 1：检索有关特定分配的配置。


```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDistributionConfig](#) 中的。

Get-CFDistributionList

以下代码示例演示了如何使用 Get-CFDistributionList。

适用于 PowerShell V4 的工具

示例 1：返回分配。

```
Get-CFDistributionList
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDistributions](#) 中的。

New-CFDistribution

以下代码示例演示了如何使用 New-CFDistribution。

适用于 PowerShell V4 的工具

示例 1：创建配置了日志和缓存的基本 CloudFront 发行版。

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
    -DistributionConfig_Enabled $true `
    -DistributionConfig_Comment "Test distribution" `
    -Origins_Item $origin `
    -Origins_Quantity 1 `
    -Logging_Enabled $true `
    -Logging_IncludeCookie $true `
    -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
    -Logging_Prefix "help/" `
```

```
-DistributionConfig_CallerReference Client1 `
-DistributionConfig_DefaultRootObject index.html `
-DefaultCacheBehavior_TargetOriginId $origin.Id `
-ForwardedValues_QueryString $true `
-Cookies_Forward all `
-WhitelistedNames_Quantity 0 `
-TrustedSigners_Enabled $false `
-TrustedSigners_Quantity 0 `
-DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
-DefaultCacheBehavior_MinTTL 1000 `
-DistributionConfig_PriceClass "PriceClass_All" `
-CacheBehaviors_Quantity 0 `
-Aliases_Quantity 0
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateDistribution](#) 中的。

New-CFInvalidation

以下代码示例演示了如何使用 New-CFInvalidation。

适用于 PowerShell V4 的工具

示例 1：此示例在 ID 为 EXAMPLNSTXAXE 的分配上创建一个新的失效。CallerReference 是用户选择的唯一 ID；在本例中，使用代表 2019 年 5 月 15 日上午 9:00 的时间戳。\$Paths 变量存储了用户不希望将其作为分配缓存一部分的图像和媒体文件的三个路径。-Paths_Quantity 参数值是在 -Paths_Item 参数中指定的路径总数。

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLNSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
3
```

输出：

Invalidation	Location
-----	-----
Amazon.CloudFront.Model.Invalidation	https://cloudfront.amazonaws.com/2018-11-05/distribution/EXAMPLNSTXAXE/invalidation/EXAMPLE8NOK9H

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateInvalidation](#) 中的。

New-CFSignedCookie

以下代码示例演示了如何使用 New-CFSignedCookie。

适用于 PowerShell V4 的工具

示例 1：使用标准策略创建指向指定资源的签名 Cookie。该 Cookie 的有效期为一年。

```
$params = @{
  "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

输出：

```
Expires
-----
[CloudFront-Expires, 1472227284]
```

示例 2：使用自定义策略创建指向指定资源的签名 Cookie。该 Cookie 将在 24 小时内生效，并将在一周后过期。

```
$start = (Get-Date).AddHours(24)
$params = @{
  "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=$start.AddDays(7)
  "ActiveFrom"=$start
}
New-CFSignedCookie @params
```

输出：

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

示例 3：使用自定义策略创建指向指定资源的签名 Cookie。该 Cookie 将在 24 小时内生效，并将在一周后过期。对资源的访问仅限于指定的 IP 范围。

```
$start = (Get-Date).AddHours(24)
$params = @{
    "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
    "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
    "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
    "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}

New-CFSignedCookie @params
```

输出：

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V4) 中的“新增-CFSigned [ookie](#)”。

New-CFSignedUrl

以下代码示例演示了如何使用 New-CFSignedUrl。

适用于 PowerShell V4 的工具

示例 1：使用标准策略创建指向指定资源的签名 url。url 的有效期为 1 小时。包含签名 url 的 System.Uri 对象发送到管道。

```
$params = @{
    "ResourceUri"="https://cdn.example.com/index.html"
```

```
"KeyId"="AKIAIOSFODNN7EXAMPLE"
"PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
"ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

示例 2：使用自定义策略创建指向指定资源的签名 url。该 url 将在 24 小时内开始生效，并将在一周后过期。

```
$start = (Get-Date).AddHours(24)
$params = @{
    "ResourceUri"="https://cdn.example.com/index.html"
    "KeyId"="AKIAIOSFODNN7EXAMPLE"
    "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
    "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

示例 3：使用自定义策略创建指向指定资源的签名 URL。该 url 将在 24 小时内开始生效，并将在一周后过期。对资源的访问仅限于指定的 IP 范围。

```
$start = (Get-Date).AddHours(24)
$params = @{
    "ResourceUri"="https://cdn.example.com/index.html"
    "KeyId"="AKIAIOSFODNN7EXAMPLE"
    "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
    "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) 中的 [新建 CFSigned 网址](#)。

CloudTrail 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CloudTrail 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Find-CTEvent

以下代码示例演示了如何使用 Find-CTEvent。

适用于 PowerShell V4 的工具

示例 1：返回过去七天内发生的所有事件。默认情况下，cmdlet 会自动发出多个调用以传送所有事件，当服务指示没有更多数据可用时退出。

```
Find-CTEvent
```

示例 2：返回过去七天内发生的所有事件，且指定的区域不是当前 Shell 默认区域。

```
Find-CTEvent -Region eu-central-1
```

示例 3：返回与 RunInstances API 调用关联的所有事件。

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";  
    AttributeValue="RunInstances" }
```

示例 4：返回前 5 个可用事件。

```
Find-CTEvent -MaxResult 5
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [LookupEvents](#) 中的。

Get-CTTrail

以下代码示例演示了如何使用 Get-CTTrail。

适用于 PowerShell V4 的工具

示例 1：返回与您账户的当前区域关联的所有跟踪的设置。

```
Get-CTTrail
```

示例 2：返回指定跟踪的设置。

```
Get-CTTrail -TrailNameList trail1, trail2
```

示例 3：返回在当前 Shell 默认区域以外的区域 [在本例中为法兰克福 (eu-central-1) 区域] 中创建的指定跟踪的设置。

```
Get-CTTrail -TrailNameList trailABC, trailDEF -Region eu-central-1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrails](#) 中的。

Get-CTTrailStatus

以下代码示例演示了如何使用 Get-CTTrailStatus。

适用于 PowerShell V4 的工具

示例 1：返回名为 “myExampleTrail” 的跟踪的状态信息。返回的数据包括有关传输错误、Amazon SNS 和 Amazon S3 错误以及跟踪的开始和停止日志记录时间的信息。此示例假设跟踪是在与当前 Shell 默认区域相同的区域中创建的。

```
Get-CTTrailStatus -Name myExampleTrail
```

示例 2：返回在当前 Shell 默认区域以外的区域 [在本例中为法兰克福 (eu-central-1) 区域] 中创建的跟踪的状态信息。

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetTrailStatus](#) 中的。

New-CTTrail

以下代码示例演示了如何使用 New-CTTrail。

适用于 PowerShell V4 的工具

示例 1：创建将使用存储桶“amzn-s3-demo-bucket”来存储日志文件的跟踪。

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

示例 2：创建将使用存储桶“amzn-s3-demo-bucket”来存储日志文件的跟踪。表示日志的 S3 对象的通用键前缀为“mylogs”。当新日志传送到存储桶时，系统将向 SNS 主题“mlog-deliverytopic”发送通知。此示例使用 splatting 向 cmdlet 提供参数值。

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTrail](#) 中的。

Remove-CTTrail

以下代码示例演示了如何使用 Remove-CTTrail。

适用于 PowerShell V4 的工具

示例 1：删除指定的跟踪。在运行此命令之前，系统会提示您进行确认。要禁止确认，请添加 -Force 开关参数。

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTrail](#) 中的。

Start-CTLogging

以下代码示例演示了如何使用 Start-CTLogging。

适用于 PowerShell V4 的工具

示例 1：开始为名为 “myExampleTrail” 的跟踪记录 AWS API 调用和日志文件传输。此示例假设跟踪是在与当前 Shell 默认区域相同的区域中创建的。

```
Start-CTLogging -Name myExampleTrail
```

示例 2：开始记录在当前外壳默认区域（在本例中为法兰克福 (eu-central-1) 区域）中创建的跟踪 AWS API 调用和日志文件传输。

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartLogging](#) 中的。

Stop-CTLogging

以下代码示例演示了如何使用 Stop-CTLogging。

适用于 PowerShell V4 的工具

示例 1：暂停记录名为 “myExampleTrail” 的跟踪的 AWS API 调用和日志文件传输。此示例假设跟踪是在与当前 Shell 默认区域相同的区域中创建的。

```
Stop-CTLogging -Name myExampleTrail
```

示例 2：暂停记录在当前外壳默认区域（在本例中为法兰克福 (eu-central-1) 区域）中创建的跟踪 AWS API 调用和日志文件传输。

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopLogging](#) 中的。

Update-CTTrail

以下代码示例演示了如何使用 Update-CTTrail。

适用于 PowerShell V4 的工具

示例 1：更新指定的跟踪，以便记录全局服务事件（例如来自 IAM 的服务事件），并将后续日志文件的通用键前缀更改为“globallogs”。

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -S3KeyPrefix "globallogs"
```

示例 2：更新指定的跟踪，以便将有关新日志传输的通知发送到指定的 SNS 主题。

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

示例 3：更新指定的跟踪，以便将日志传送到其它存储桶。

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateTrail](#) 中的。

CloudWatch 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CloudWatch 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-CWAlarm

以下代码示例演示了如何使用 Get-CWAlarm。

适用于 PowerShell V4 的工具

示例 1：返回所有警报，包括来自的复合警报和指标警报 CloudWatch。

```
Get-CWAlarm -MaxRecords 1
```

输出：

CompositeAlarms	MetricAlarms	NextToken
-----	-----	-----
	{MetricAlarms-01}	NextToken-01
	{MetricAlarms-02}	NextToken-02
	{MetricAlarms-03}	NextToken-03

示例 2：仅返回将 AlarmType 参数设置为 CloudWatch 后的复合警报数据 CompositeAlarms。

```
Get-CWAlarm -AlarmType 'CompositeAlarms'
```

输出：

CompositeAlarms	MetricAlarms	NextToken
-----	-----	-----
{CompositeAlarms-01}		
{CompositeAlarms-02}		
{CompositeAlarms-03}		

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeAlarms](#) 中的。

Get-CWDashboard

以下代码示例演示了如何使用 Get-CWDashboard。

适用于 PowerShell V4 的工具

示例 1：返回指定控制面板的正文 arn。

```
Get-CWDashboard -DashboardName Dashboard1
```

输出：

DashboardArn	DashboardBody
-----	-----
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1	{...

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDashboard](#) 中的。

Get-CWDashboardList

以下代码示例演示了如何使用 Get-CWDashboardList。

适用于 PowerShell V4 的工具

示例 1：返回您账户的控制面板集合。

```
Get-CWDashboardList
```

输出：

DashboardArn	DashboardName	LastModified	Size
-----	-----	-----	----
arn:...	Dashboard1	7/6/2017 8:14:15 PM	252

示例 2：返回名称以前缀“dev”开头的账户的控制面板集合。

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDashboards](#) 中的。

Remove-CWDashboard

以下代码示例演示了如何使用 Remove-CWDashboard。

适用于 PowerShell V4 的工具

示例 1：删除指定的控制面板，继续操作前提示确认。要绕过确认，请在命令中添加 -Force 开关。

```
Remove-CWDashboard -DashboardName Dashboard1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteDashboards](#) 中的。

Write-CWDashboard

以下代码示例演示了如何使用 Write-CWDashboard。

适用于 PowerShell V4 的工具

示例 1：创建或更新名为“Dashboard1”的控制面板，以包含两个并排的指标小部件。

```
$dashBody = @"
{
  "widgets":[
    {
      "type":"metric",
      "x":0,
      "y":0,
      "width":12,
      "height":6,
      "properties":{"
        "metrics":[
          [
            "AWS/EC2",
            "CPUUtilization",
            "InstanceId",
            "i-012345"
          ]
        ],
        "period":300,
        "stat":"Average",
        "region":"us-east-1",
        "title":"EC2 Instance CPU"
      }
    },
    {
      "type":"metric",
      "x":12,
      "y":0,
      "width":12,
      "height":6,
      "properties":{"
        "metrics":[
```

```

        [
            "AWS/S3",
            "BucketSizeBytes",
            "BucketName",
            "amzn-s3-demo-bucket"
        ],
        "period":86400,
        "stat":"Maximum",
        "region":"us-east-1",
        "title":"amzn-s3-demo-bucket bytes"
    }
}
]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody

```

示例 2：创建或更新控制面板，将描述控制面板的内容通过管道传输到 cmdlet 中。

```

$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutDashboard](#) 中的。

Write-CWMetricData

以下代码示例演示了如何使用 Write-CWMetricData。

适用于 PowerShell V4 的工具

示例 1：创建新 MetricDatum 对象并将其写入 Amazon Web Serv CloudWatch 指标。

```

### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum

```

```
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutMetricData](#) 中的。

CodeCommit 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CodeCommit 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-CCBranch

以下代码示例演示了如何使用 Get-CCBranch。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定存储库的指定分支的信息。

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

输出：

BranchName	CommitId
-----	-----

```
MyNewBranch                                7763222d...561fc9c9
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetBranch](#) 中的。

Get-CCBranchList

以下代码示例演示了如何使用 Get-CCBranchList。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定存储库的分支名称的列表。

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

输出：

```
master
MyNewBranch
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListBranches](#) 中的。

Get-CCRepository

以下代码示例演示了如何使用 Get-CCRepository。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定存储库的信息。

```
Get-CCRepository -RepositoryName MyDemoRepo
```

输出：

```
AccountId           : 80398EXAMPLE
Arn                 : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp       : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh        : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate       : 9/8/2015 3:21:33 PM
DefaultBranch      :
```



```
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetRepository](#) 中的。

Get-CCRepositoryBatch

以下代码示例演示了如何使用 Get-CCRepositoryBatch。

适用于 PowerShell V4 的工具

示例 1：此示例确认已找到和未找到哪些指定的存储库。

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

输出：

Repositories	RepositoriesNotFound
-----	-----
{MyDemoRepo, MyNewRepo}	{AMissingRepo}

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [BatchGetRepositories](#) 中的。

Get-CCRepositoryList

以下代码示例演示了如何使用 Get-CCRepositoryList。

适用于 PowerShell V4 的工具

示例 1：此示例按存储库名称的升序列出所有存储库。

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

输出：

RepositoryId	RepositoryName

```

-----
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListRepositories](#) 中的。

New-CCBranch

以下代码示例演示了如何使用 New-CCBranch。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定存储库的指定名称和指定的提交 ID 创建一个新分支。

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
7763222d...561fc9c9
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateBranch](#) 中的。

New-CCRepository

以下代码示例演示了如何使用 New-CCRepository。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的名称和指定的描述创建新的存储库。

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
repository for demonstration purposes."
```

输出：

```

AccountId           : 80398EXAMPLE
Arn                 : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp        : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh          : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo

```

```
CreationDate      : 9/18/2015 4:13:25 PM
DefaultBranch     :
LastModifiedDate  : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId      : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName    : MyDemoRepo
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateRepository](#) 中的。

Remove-CCRepository

以下代码示例演示了如何使用 Remove-CCRepository。

适用于 PowerShell V4 的工具

示例 1：此示例强制删除指定的存储库。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除存储库。

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

输出：

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRepository](#) 中的。

Update-CCDefaultBranch

以下代码示例演示了如何使用 Update-CCDefaultBranch。

适用于 PowerShell V4 的工具

示例 1：此示例将指定存储库的默认分支更改为指定的分支。

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateDefaultBranch](#) 中的。

Update-CCRepositoryDescription

以下代码示例演示了如何使用 Update-CCRepositoryDescription。

适用于 PowerShell V4 的工具

示例 1：此示例更改指定存储库的描述。

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription  
"This is an updated description."
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateRepositoryDescription](#) 中的。

Update-CCRepositoryName

以下代码示例演示了如何使用 Update-CCRepositoryName。

适用于 PowerShell V4 的工具

示例 1：此示例更改指定存储库的名称。

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateRepositoryName](#) 中的。

CodeDeploy 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CodeDeploy 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 操作

操作

Add-CDOnPremiseInstanceTag

以下代码示例演示了如何使用 Add-CDOnPremiseInstanceTag。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的本地实例添加具有指定键和值的本地实例标签。

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
"Value" = "CodeDeployDemo-OnPrem"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddTagsToOnPremisesInstances](#) 中的。

Get-CDApplication

以下代码示例演示了如何使用 Get-CDApplication。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定应用程序的信息。

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

输出：

ApplicationId LinkedToGitHub	ApplicationName	CreateTime
----- -----	-----	-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE 9:49:48 PM False	CodeDeployDemoApplication	7/20/2015

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetApplication](#) 中的。

Get-CDApplicationBatch

以下代码示例演示了如何使用 `Get-CDApplicationBatch`。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定应用程序的信息。

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,  
CodePipelineDemoApplication
```

输出：

ApplicationId	ApplicationName	CreateTime
----- ----- LinkedToGitHub	-----	-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE 9:49:48 PM False	CodeDeployDemoApplication	7/20/2015
1ecfd602-62f1-4038-8f0d-06688EXAMPLE 5:53:26 PM False	CodePipelineDemoApplication	8/13/2015

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) BatchGetApplications](#) 中的。

Get-CDApplicationList

以下代码示例演示了如何使用 `Get-CDApplicationList`。

适用于 PowerShell V4 的工具

示例 1：此示例获取可用应用程序的列表。

Get-CDApplicationList

输出：

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListApplications](#) 中的。

Get-CDApplicationRevision

以下代码示例演示了如何使用 Get-CDApplicationRevision。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定应用程序版本的信息。

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
-S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

输出：

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
RegisterTime = 07/20/2015 23:46:42
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetApplicationRevision](#) 中的。

Get-CDApplicationRevisionList

以下代码示例演示了如何使用 Get-CDApplicationRevisionList。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定应用程序的可用修订版的信息。

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
CodeDeployDemoApplication -Deployed Ignore)) {
>> If ($revision.RevisionType -Eq "S3") {
>> Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
+ ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
$revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>> }
```

```
>> If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
    $revision.GitHubLocation.CommitId + ", Repository = " +
    $revision.GitHubLocation.Repository)
>> }
>> }
>>
```

输出：

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListApplicationRevisions](#) 中的。

Get-CDDeployment

以下代码示例演示了如何使用 Get-CDDeployment。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定部署的摘要信息。

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

输出：

```
ApplicationName      : CodeDeployDemoApplication
CompleteTime         : 7/23/2015 11:26:04 PM
CreateTime           : 7/23/2015 11:24:43 PM
Creator              : user
DeploymentConfigName  : CodeDeployDefault.OneAtATime
DeploymentGroupName   : CodeDeployDemoFleet
DeploymentId          : d-QZMRGSTEX
DeploymentOverview    : Amazon.CodeDeploy.Model.DeploymentOverview
Description           :
ErrorInformation      :
IgnoreApplicationStopFailures : False
Revision             : Amazon.CodeDeploy.Model.RevisionLocation
```



```
StartTime          : 1/1/0001 12:00:00 AM
Status             : Succeeded
```

示例 2：此示例获取有关参与指定部署的实例的状态信息。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

输出：

```
Failed      : 0
InProgress  : 0
Pending     : 0
Skipped     : 0
Succeeded   : 3
```

示例 3：此示例获取有关指定部署的应用程序修订版的信息。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

输出：

```
Bucket      : amzn-s3-demo-bucket
BundleType  : zip
ETag        : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key         : clzfqEX
Version     :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDeployment](#) 中的。

Get-CDDeploymentBatch

以下代码示例演示了如何使用 Get-CDDeploymentBatch。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定部署的信息。

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

输出：

```

ApplicationName      : CodeDeployDemoApplication
CompleteTime         : 7/23/2015 11:26:04 PM
CreateTime           : 7/23/2015 11:24:43 PM
Creator              : user
DeploymentConfigName  : CodeDeployDefault.OneAtATime
DeploymentGroupName  : CodeDeployDemoFleet
DeploymentId          : d-QZMRGSTEX
DeploymentOverview    : Amazon.CodeDeploy.Model.DeploymentOverview
Description           :
ErrorInformation      :
IgnoreApplicationStopFailures : False
Revision             : Amazon.CodeDeploy.Model.RevisionLocation
StartTime            : 1/1/0001 12:00:00 AM
Status               : Succeeded

ApplicationName      : CodePipelineDemoApplication
CompleteTime         : 7/23/2015 6:07:30 PM
CreateTime           : 7/23/2015 6:06:29 PM
Creator              : user
DeploymentConfigName  : CodeDeployDefault.OneAtATime
DeploymentGroupName  : CodePipelineDemoFleet
DeploymentId          : d-RR0T5KTEX
DeploymentOverview    : Amazon.CodeDeploy.Model.DeploymentOverview
Description           :
ErrorInformation      :
IgnoreApplicationStopFailures : False
Revision             : Amazon.CodeDeploy.Model.RevisionLocation
StartTime            : 1/1/0001 12:00:00 AM
Status               : Succeeded

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [BatchGetDeployments](#) 中的。

Get-CDDeploymentConfig

以下代码示例演示了如何使用 Get-CDDeploymentConfig。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定部署配置的摘要信息。

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

输出：

CreateTime	DeploymentConfigId	DeploymentConfigName
MinimumHealthyHosts		
-----	-----	-----

10/3/2014 4:32:30 PM	518a3950-d034-46a1-9d2c-3c949EXAMPLE	ThreeQuartersHealthy
Amazon.CodeDeploy.Model.MinimumHealthyHosts		

示例 2：此示例获取有关指定部署配置的定义的信息。

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
ThreeQuartersHealthy).MinimumHealthyHosts)
```

输出：

Type	Value
----	-----
FLEET_PERCENT	75

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetDeploymentConfig](#) 中的。

Get-CDDeploymentConfigList

以下代码示例演示了如何使用 Get-CDDeploymentConfigList。

适用于 PowerShell V4 的工具

示例 1：此示例获取可用部署配置列表。

```
Get-CDDeploymentConfigList
```

输出：

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListDeploymentConfigs](#) 中的。

Get-CDDeploymentGroup

以下代码示例演示了如何使用 Get-CDDeploymentGroup。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定部署组的信息。

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -  
DeploymentGroupName CodeDeployDemoFleet
```

输出：

```
ApplicationName           : CodeDeployDemoApplication  
AutoScalingGroups        : {}  
DeploymentConfigName      : CodeDeployDefault.OneAtATime  
DeploymentGroupId         : 7d7c098a-b444-4b27-96ef-22791EXAMPLE  
DeploymentGroupName       : CodeDeployDemoFleet  
Ec2TagFilters             : {Name}  
OnPremisesInstanceTagFilters : {}  
ServiceRoleArn           : arn:aws:iam::80398EXAMPLE:role/  
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-09MWP7XTL8EX  
TargetRevision           : Amazon.CodeDeploy.Model.RevisionLocation
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetDeploymentGroup](#) 中的。

Get-CDDeploymentGroupList

以下代码示例演示了如何使用 Get-CDDeploymentGroupList。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定应用程序的部署组的列表。

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

输出：

ApplicationName	DeploymentGroups
NextToken	
-----	-----

CodeDeployDemoApplication	{CodeDeployDemoFleet, CodeDeployProductionFleet}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListDeploymentGroups](#) 中的。

Get-CDDeploymentInstance

以下代码示例演示了如何使用 Get-CDDeploymentInstance。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定部署的指定实例的信息。

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

输出：

```
DeploymentId      : d-QZMRGSTEX
InstanceId        : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt     : 7/23/2015 11:25:24 PM
LifecycleEvents   : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status            : Succeeded
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetDeploymentInstance](#) 中的。

Get-CDDeploymentInstanceList

以下代码示例演示了如何使用 Get-CDDeploymentInstanceList。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定部署 IDs 的实例列表。

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

输出：

```
i-254e22EX  
i-274e22EX  
i-3b4e22EX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDeploymentInstances](#) 中的。

Get-CDDeploymentList

以下代码示例演示了如何使用 Get-CDDeploymentList。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定应用程序和部署 IDs 组的部署列表。

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName  
CodeDeployDemoFleet
```

输出：

```
d-QZMRGSTEX  
d-RR0T5KTEX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDeployments](#) 中的。

Get-CDOnPremiseInstance

以下代码示例演示了如何使用 Get-CDOnPremiseInstance。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定本地实例的信息。

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

输出：

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn      : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn     : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName    : AssetTag12010298EX
RegisterTime    : 4/3/2015 6:36:24 PM
Tags            : {Name}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetOnPremisesInstance](#) 中的。

Get-CDOnPremiseInstanceBatch

以下代码示例演示了如何使用 Get-CDOnPremiseInstanceBatch。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定本地实例的信息。

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

输出：

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn      : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn     : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName    : AssetTag12010298EX-2
RegisterTime    : 4/3/2015 6:38:52 PM
Tags            : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn      : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn     : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName    : AssetTag12010298EX
RegisterTime    : 4/3/2015 6:36:24 PM
Tags            : {Name}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [BatchGetOnPremisesInstances](#) 中的。

Get-CDOnPremiseInstanceList

以下代码示例演示了如何使用 Get-CDOnPremiseInstanceList。

适用于 PowerShell V4 的工具

示例 1：此示例获取可用本地实例名称的列表。

```
Get-CDOnPremiseInstanceList
```

输出：

```
AssetTag12010298EX  
AssetTag12010298EX-2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListOnPremisesInstances](#) 中的。

New-CDApplication

以下代码示例演示了如何使用 New-CDApplication。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定名称创建新的应用程序。

```
New-CDApplication -ApplicationName MyNewApplication
```

输出：

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateApplication](#) 中的。

New-CDDeployment

以下代码示例演示了如何使用 New-CDDeployment。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的部署配置和应用程序修订版为指定的应用程序和部署组创建新部署。

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

输出：

```
d-ZHR0G7UEX
```

示例 2：此示例说明如何指定 EC2 实例标签组，实例必须通过这些标签进行标识，才能将其包含在 blue/green 部署的替换环境中。

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
-S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

输出：

```
d-ZHR0G7UEX
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateDeployment](#) 中的。

New-CDDeploymentConfig

以下代码示例演示了如何使用 New-CDDeploymentConfig。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的名称和行为创建新的部署配置。

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

输出：

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateDeploymentConfig](#) 中的。

New-CDDeploymentGroup

以下代码示例演示了如何使用 New-CDDeploymentGroup。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的应用程序创建具有指定名称、自动扩缩组、部署配置、标签和服务角色的部署组。

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

输出：

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

示例 2：此示例说明如何指定 EC2 实例标签组，实例必须通过这些标签进行标识，才能将其包含在 blue/green 部署的替换环境中。

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

输出：

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateDeploymentGroup](#) 中的。

Register-CDApplicationRevision

以下代码示例演示了如何使用 Register-CDApplicationRevision。

适用于 PowerShell V4 的工具

示例 1：此示例在指定的 Amazon S3 位置为指定的应用程序注册应用程序修订版。

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket  
    amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-  
    codedeploy_linux-master.zip -Revision_RevisionType S3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterApplicationRevision](#) 中的。

Register-CDOnPremiseInstance

以下代码示例演示了如何使用 Register-CDOnPremiseInstance。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的名称和 IAM 用户注册本地实例。

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/  
    CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterOnPremisesInstance](#) 中的。

Remove-CDApplication

以下代码示例演示了如何使用 Remove-CDApplication。

适用于 PowerShell V4 的工具

示例 1：此示例删除具有指定名称的应用程序。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除应用程序。

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteApplication](#) 中的。

Remove-CDDeploymentConfig

以下代码示例演示了如何使用 Remove-CDDeploymentConfig。

适用于 PowerShell V4 的工具

示例 1：此示例删除具有指定名称的部署配置。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除部署配置。

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteDeploymentConfig](#) 中的。

Remove-CDDeploymentGroup

以下代码示例演示了如何使用 Remove-CDDeploymentGroup。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定应用程序的具有指定名称的部署组。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除部署组。

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName  
MyNewDeploymentGroup
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteDeploymentGroup](#) 中的。

Remove-CDOnPremiseInstanceTag

以下代码示例演示了如何使用 Remove-CDOnPremiseInstanceTag。

适用于 PowerShell V4 的工具

示例 1：此示例为具有指定名称的本地实例删除指定的标签。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除标签。

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =  
    "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveTagsFromOnPremisesInstances](#) 中的。

Stop-CDDeployment

以下代码示例演示了如何使用 Stop-CDDeployment。

适用于 PowerShell V4 的工具

示例 1：此示例尝试停止具有指定部署 ID 的部署。

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

输出：

Status	StatusMessage
-----	-----
Pending	Stopping Pending. Stopping to schedule commands in the deployment instances

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopDeployment](#) 中的。

Unregister-CDOnPremiseInstance

以下代码示例演示了如何使用 Unregister-CDOnPremiseInstance。

适用于 PowerShell V4 的工具

示例 1：此示例取消注册具有指定名称的本地实例。

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeregisterOnPremisesInstance](#) 中的。

Update-CDApplication

以下代码示例演示了如何使用 Update-CDApplication。

适用于 PowerShell V4 的工具

示例 1：此示例更改指定应用程序的名称。

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName  
MyNewApplication-2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateApplication](#) 中的。

Update-CDDeploymentGroup

以下代码示例演示了如何使用 Update-CDDeploymentGroup。

适用于 PowerShell V4 的工具

示例 1：此示例更改指定应用程序的指定部署组的名称。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -  
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName  
MyNewDeploymentGroup-2
```

示例 2：此示例说明如何指定 EC2 实例标签组，实例必须通过这些标签进行标识，才能将其包含在 blue/green 部署的替换环境中。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -  
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName  
MyNewDeploymentGroup-2 -Ec2TagSetList  
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateDeploymentGroup](#) 中的。

CodePipeline 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 CodePipeline 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Confirm-CPJob

以下代码示例演示了如何使用 Confirm-CPJob。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定作业的状态。

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

输出：

```
Value
-----
InProgress
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AcknowledgeJob](#) 中的。

Disable-CPStageTransition

以下代码示例演示了如何使用 Disable-CPStageTransition。

适用于 PowerShell V4 的工具

示例 1：此示例对指定管道中的指定阶段禁用入站转换。

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling temporarily." -StageName Beta -TransitionType Inbound
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DisableStageTransition](#) 中的。

Enable-CPStageTransition

以下代码示例演示了如何使用 Enable-CPStageTransition。

适用于 PowerShell V4 的工具

示例 1：此示例对指定管道中的指定阶段启用入站转换。

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -TransitionType Inbound
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [EnableStageTransition](#) 中的。

Get-CPActionType

以下代码示例演示了如何使用 Get-CPActionType。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定所有者的所有可用操作的信息。

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
    Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
    $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version = " +
    $actionType.Id.Version + ":")
    Write-Output ("  ActionConfigurationProperties:")
    ForEach ($acp in $actionType.ActionConfigurationProperties) {
        Write-Output ("    For " + $acp.Name + ":")
        Write-Output ("      Description = " + $acp.Description)
        Write-Output ("      Key = " + $acp.Key)
```



```

        Write-Output ("        Queryable = " + $acp.Queryable)
        Write-Output ("        Required = " + $acp.Required)
        Write-Output ("        Secret = " + $acp.Secret)
    }
    Write-Output ("    InputArtifactDetails:")
    Write-Output ("        MaximumCount = " +
$actionType.InputArtifactDetails.MaximumCount)
    Write-Output ("        MinimumCount = " +
$actionType.InputArtifactDetails.MinimumCount)
    Write-Output ("    OutputArtifactDetails:")
    Write-Output ("        MaximumCount = " +
$actionType.OutputArtifactDetails.MaximumCount)
    Write-Output ("        MinimumCount = " +
$actionType.OutputArtifactDetails.MinimumCount)
    Write-Output ("    Settings:")
    Write-Output ("        EntityUrlTemplate = " + $actionType.Settings.EntityUrlTemplate)
    Write-Output ("        ExecutionUrlTemplate = " +
$actionType.Settings.ExecutionUrlTemplate)
}

```

输出：

```

For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
ActionConfigurationProperties:
  For ApplicationName:
    Description = The AWS Elastic Beanstalk Application name
    Key = True
    Queryable = False
    Required = True
    Secret = False
  For EnvironmentName:
    Description = The AWS Elastic Beanstalk Environment name
    Key = True
    Queryable = False
    Required = True
    Secret = False
InputArtifactDetails:
  MaximumCount = 1
  MinimumCount = 1
OutputArtifactDetails:
  MaximumCount = 0
  MinimumCount = 0
Settings:

```

```

    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
    ActionConfigurationProperties:
        For ApplicationName:
            Description = The AWS CodeDeploy Application name
            Key = True
            Queryable = False
            Required = True
            Secret = False
        For DeploymentGroupName:
            Description = The AWS CodeDeploy Deployment Group name
            Key = True
            Queryable = False
            Required = True
            Secret = False
    InputArtifactDetails:
        MaximumCount = 1
        MinimumCount = 1
    OutputArtifactDetails:
        MaximumCount = 0
        MinimumCount = 0
    Settings:
        EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
applications/{Config:ApplicationName}/deployment-groups/{Config:DeploymentGroupName}
        ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListActionTypes](#) 中的。

Get-CPActionableJobList

以下代码示例演示了如何使用 Get-CPActionableJobList。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定操作类别、所有者、提供者、版本和查询参数的所有可操作作业的信息。

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
-ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
@{"ProjectName" = "MyProjectName"}
```

输出：

AccountId	Data	Id
----- -----	----	--
80398EXAMPLE f57a0EXAMPLE	Amazon.CodePipeline.Model.JobData 3	0de392f5-712d-4f41-ace3-

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PollForJobs](#) 中的。

Get-CPJobDetail

以下代码示例演示了如何使用 Get-CPJobDetail。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定作业的一般信息。

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

输出：

AccountId	Data	Id
----- -----	----	--
80398EXAMPLE f570dc12-5ef3-44bc-945a-6e133EXAMPLE	Amazon.CodePipeline.Model.JobData	

示例 2：此示例获取有关指定作业的详细信息。

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output (" AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output (" Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
```

```

    $value = $jobData.ActionConfiguration.$key
    Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
    $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
    Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
    Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)

```

输出：

```

For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp

```

```
OutputArtifacts:
  MyAppBuild
PipelineContext:
  Name = Build
  PipelineName = CodePipelineDemo
  Stage = Build
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetJobDetails](#) 中的。

Get-CPPipeline

以下代码示例演示了如何使用 Get-CPPipeline。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定管道的一般信息。

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

输出：

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

示例 2：此示例获取有关指定管道的详细信息。

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
}
```

```

    ForEach ($action in $stage.Actions) {
        Write-Output ("        Name = " + $action.Name)
    Write-Output ("        Category = " + $action.ActionTypeId.Category)
    Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
    Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
    Write-Output ("        Version = " + $action.ActionTypeId.Version)
    Write-Output ("        Configuration:")
    ForEach ($key in $action.Configuration.Keys) {
        $value = $action.Configuration.$key
        Write-Output ("            " + $key + " = " + $value)
    }
    Write-Output ("        InputArtifacts:")
    ForEach ($ia in $action.InputArtifacts) {
        Write-Output ("            " + $ia.Name)
    }
    ForEach ($oa in $action.OutputArtifacts) {
        Write-Output ("            " + $oa.Name)
    }
    Write-Output ("        RunOrder = " + $action.RunOrder)
    }
}

```

输出：

```

Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
    Location = amzn-s3-demo-bucket
    Type = S3
Stages:
    Name = Source
    Actions:
        Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
            Branch = master
            OAuthToken = ****
            Owner = my-user-name
            Repo = MyRepoName

```

```

    InputArtifacts:
        MyApp
    RunOrder = 1
Name = Build
    Actions:
        Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
            ProjectName = MyProjectName
        InputArtifacts:
            MyApp
            MyAppBuild
        RunOrder = 1
Name = Beta
    Actions:
        Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
            ApplicationName = CodePipelineDemoApplication
            DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
            MyAppBuild
        RunOrder = 1
Name = TestStage
    Actions:
        Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
            ProjectName = MyJenkinsProjectName
        InputArtifacts:
            MyAppBuild
        RunOrder = 1

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPipeline](#) 中的。

Get-CPPipelineList

以下代码示例演示了如何使用 Get-CPPipelineList。

适用于 PowerShell V4 的工具

示例 1：此示例获取可用管道的列表。

```
Get-CPPipelineList
```

输出：

Created	Name	Updated	Version
-----	----	-----	-----
8/13/2015 10:17:54 PM	CodePipelineDemo	8/13/2015 10:17:54 PM	3
7/8/2015 2:41:53 AM	MyFirstPipeline	7/22/2015 9:06:37 PM	7

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListPipelines](#) 中的。

Get-CPPipelineState

以下代码示例演示了如何使用 Get-CPPipelineState。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关指定管道的阶段的一般信息。

```
Get-CPPipelineState -Name CodePipelineDemo
```

输出：

```
Created       : 8/13/2015 10:17:54 PM
PipelineName  : CodePipelineDemo
PipelineVersion : 1
StageStates   : {Source, Build, Beta, TestStage}
Updated       : 8/13/2015 10:17:54 PM
```

示例 2：此示例获取有关指定管道的阶段的详细信息。

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
```



```

Write-Output ("For " + $stageState.StageName + ":")
Write-Output ("  InboundTransitionState:")
Write-Output ("    DisabledReason = " +
$stageState.InboundTransitionState.DisabledReason)
Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
Write-Output ("    LastChangedAt = " +
$stageState.InboundTransitionState.LastChangedAt)
Write-Output ("    LastChangedBy = " +
$stageState.InboundTransitionState.LastChangedBy)
Write-Output ("  ActionStates:")
ForEach ($actionState in $stageState.ActionStates) {
  Write-Output ("    For " + $actionState.ActionName + ":")
Write-Output ("      CurrentRevision:")
  Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
Write-Output ("        RevisionChangeId = " +
$actionState.CurrentRevision.RevisionChangeId)
Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
Write-Output ("        EntityUrl = " + $actionState.EntityUrl)
Write-Output ("        LatestExecution:")
  Write-Output ("          ErrorDetails:")
  Write-Output ("            Code = " +
$actionState.LatestExecution.ErrorDetails.Code)
Write-Output ("            Message = " +
$actionState.LatestExecution.ErrorDetails.Message)
Write-Output ("            ExternalExecutionId = " +
$actionState.LatestExecution.ExternalExecutionId)
Write-Output ("            ExternalExecutionUrl = " +
$actionState.LatestExecution.ExternalExecutionUrl)
Write-Output ("            LastStatusChange = " +
$actionState.LatestExecution.LastStatusChange)
Write-Output ("            PercentComplete = " +
$actionState.LatestExecution.PercentComplete)
Write-Output ("            Status = " + $actionState.LatestExecution.Status)
Write-Output ("            Summary = " + $actionState.LatestExecution.Summary)
Write-Output ("            RevisionUrl = " + $actionState.RevisionUrl)
  }
}

```

输出：

```

For Source:
  InboundTransitionState:
    DisabledReason =

```

```

    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
  For Build:
    InboundTransitionState:
      DisabledReason =
      Enabled = True
      LastChangedAt = 01/01/0001 00:00:00
      LastChangedBy =
    ActionStates:
      For Build:
        CurrentRevision:
          Created =
          RevisionChangeId =
          RevisionId =
        EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
        LatestExecution:
          ErrorDetails:
            Code = TimeoutError
            Message = The action failed because a job worker exceeded its time limit.
  If this is a custom action, make sure that the job worker is configured correctly.
    ExternalExecutionId =
    ExternalExecutionUrl =
    LastStatusChange = 07/21/2015 00:29:29
    PercentComplete = 0
    Status = Failed

```

```

        Summary =
        RevisionUrl =
    For Beta:
        InboundTransitionState:
            DisabledReason =
            Enabled = True
            LastChangedAt = 01/01/0001 00:00:00
            LastChangedBy =
        ActionStates:
            For CodePipelineDemoFleet:
                CurrentRevision:
                    Created =
                    RevisionChangeId =
                    RevisionId =
                    EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
                LatestExecution:
                    ErrorDetails:
                        Code =
                        Message =
                        ExternalExecutionId = d-D5LTCZXEX
                        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
                        LastStatusChange = 07/08/2015 22:07:42
                        PercentComplete = 0
                        Status = Succeeded
                        Summary = Deployment Succeeded
                    RevisionUrl =
    For TestStage:
        InboundTransitionState:
            DisabledReason =
            Enabled = True
            LastChangedAt = 01/01/0001 00:00:00
            LastChangedBy =
        ActionStates:
            For MyJenkinsTestAction25:
                CurrentRevision:
                    Created =
                    RevisionChangeId =
                    RevisionId =
                    EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
                LatestExecution:
                    ErrorDetails:
                        Code =

```

```

    Message =
    ExternalExecutionId = 5
    ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
    LastStatusChange = 07/08/2015 22:09:03
    PercentComplete = 0
    Status = Succeeded
    Summary = Finished
    RevisionUrl =

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPipelineState](#) 中的。

New-CPCustomActionType

以下代码示例演示了如何使用 New-CPCustomActionType。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定属性创建新的自定义操作。

```

New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
= "The name of the build project must be provided when this action is added
to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
= $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
-Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1

```

输出：

```

ActionConfigurationProperties : {ProjectName}
Id                           : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails        : Amazon.CodePipeline.Model.ArtifactDetails
Settings                     : Amazon.CodePipeline.Model.ActionTypeSettings

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateCustomActionType](#) 中的。

New-CPPipeline

以下代码示例演示了如何使用 New-CPPipeline。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定设置创建新的管道。

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageAction.OutputArtifact = New-Object
    Amazon.CodePipeline.Model.OutputArtifact
$sourceStageAction.OutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
    "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageAction.OutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageAction.InputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageAction.InputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
    "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
    "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageAction.InputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)
```

```
$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

输出：

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreatePipeline](#) 中的。

Remove-CPCustomActionType

以下代码示例演示了如何使用 Remove-CPCustomActionType。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的自定义操作。在继续操作之前，该命令将提示进行确认。添加 -Force 参数可在不提示的情况下删除自定义操作。

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteCustomActionType](#) 中的。

Remove-CPPipeline

以下代码示例演示了如何使用 Remove-CPPipeline。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的管道。在继续操作之前，该命令将提示进行确认。添加 `-Force` 参数可在不提示的情况下删除管道。

```
Remove-CCPipeline -Name CodePipelineDemo
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeletePipeline](#) 中的。

Start-CCPipelineExecution

以下代码示例演示了如何使用 `Start-CCPipelineExecution`。

适用于 PowerShell V4 的工具

示例 1：此示例开始运行指定的管道。

```
Start-CCPipelineExecution -Name CodePipelineDemo
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartPipelineExecution](#) 中的。

Update-CCPipeline

以下代码示例演示了如何使用 `Update-CCPipeline`。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的设置更新指定的现有管道。

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageAction.OutputArtifact = New-Object
    Amazon.CodePipeline.Model.OutputArtifact
$sourceStageAction.OutputArtifact.Name = "MyApp"
```

```

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
  "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
  "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
  "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline

```

输出：

```

ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2

```


- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdatePipeline](#) 中的。

使用适用于 V4 的工具的 Amazon Cognito 身份示例 PowerShell

以下代码示例向您展示了如何使用带有 Amazon Cognito Identity 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-CGIIIdentityPool

以下代码示例演示了如何使用 Get-CGIIIdentityPool。

适用于 PowerShell V4 的工具

示例 1：按身份池的 ID 检索有关该身份池的信息。

```
Get-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

输出：

```
LoggedAt                : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName    :
IdentityPoolId           : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName         : CommonTests1
OpenIdConnectProviderARNs : {}
SupportedLoginProviders   : {}
ResponseMetadata         : Amazon.Runtime.ResponseMetadata
ContentLength            : 142
```

```
HttpStatusCode          : OK
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeIdentityPool](#) 中的。

Get-CGIIdentityPoolList

以下代码示例演示了如何使用 Get-CGIIdentityPoolList。

适用于 PowerShell V4 的工具

示例 1：检索现有身份池的列表。

```
Get-CGIIdentityPoolList
```

输出：

IdentityPoolId	IdentityPoolName
-----	-----
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1	CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2	Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3	CommonTests13

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListIdentityPools](#) 中的。

Get-CGIIdentityPoolRole

以下代码示例演示了如何使用 Get-CGIIdentityPoolRole。

适用于 PowerShell V4 的工具

示例 1：获取有关特定身份池的角色的信息。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

输出：

```
LoggedAt          : 8/12/2015 4:33:51 PM
```

```

IdentityPoolId    : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles              : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata  : Amazon.Runtime.ResponseMetadata
ContentLength     : 165
HttpStatusCode    : OK

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetIdentityPoolRoles](#) 中的。

New-CGIIdentityPool

以下代码示例演示了如何使用 New-CGIIdentityPool。

适用于 PowerShell V4 的工具

示例 1：创建允许未经身份验证的身份的新身份池。

```

New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
CommonTests13

```

输出：

```

LoggedAt          : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName :
IdentityPoolId     : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName   : CommonTests13
OpenIdConnectProviderARNs : {}
SupportedLoginProviders : {}
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 136
HttpStatusCode     : OK

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateIdentityPool](#) 中的。

Remove-CGIIdentityPool

以下代码示例演示了如何使用 Remove-CGIIdentityPool。

适用于 PowerShell V4 的工具

示例 1：删除特定身份池。

```
Remove-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteIdentityPool](#) 中的。

Set-CGIIIdentityPoolRole

以下代码示例演示了如何使用 Set-CGIIIdentityPoolRole。

适用于 PowerShell V4 的工具

示例 1：将特定的身份池配置为具有未经身份验证的 IAM 角色。

```
Set-CGIIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/CommonTests1Role" }
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetIdentityPoolRoles](#) 中的。

Update-CGIIIdentityPool

以下代码示例演示了如何使用 Update-CGIIIdentityPool。

适用于 PowerShell V4 的工具

示例 1：更新某些身份池属性，在本例中为身份池的名称。

```
Update-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

输出：

```
LoggedAt : 8/12/2015 4:53:33 PM
```

```
AllowUnauthenticatedIdentities : False
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                 : NewPoolName
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders          : {}
ResponseMetadata                 : Amazon.Runtime.ResponseMetadata
ContentLength                    : 135
HttpStatusCode                   : OK
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateIdentityPool](#) 中的。

AWS Config 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS Config 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-CFGResourceTag

以下代码示例演示了如何使用 Add-CFGResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的标签与资源 ARN 相关联，在本例中为 config-rule/config-rule-16iyn0。

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResource](#) 中的。

Get-CFGAggregateComplianceByConfigRuleList

以下代码示例演示了如何使用 Get-CFGAggregateComplianceByConfigRuleList。

适用于 PowerShell V4 的工具

示例 1：此示例从给定配置规则 ConfigurationAggregator 的“kaju”筛选和规则的“合规性”expands/returns 中获取详细信息。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
-Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

输出：

ComplianceContributorCount	ComplianceType
-----	-----
Amazon.ConfigService.Model.ComplianceContributorCount	NON_COMPLIANT

示例 2：此示例从给定内容中获取详细信息 ConfigurationAggregator，针对给定账户筛选聚合器中涵盖的所有区域，并进一步返回所有规则的合规性。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
@{N="Compliance";E={$_.Compliance.ComplianceType}}
```

输出：

ConfigRuleName	Compliance
-----	-----
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK	NON_COMPLIANT
ec2-instance-no-public-ip	NON_COMPLIANT
desired-instance-type	NON_COMPLIANT

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeAggregateComplianceByConfigRules](#) 中的。

Get-CFGAggregateComplianceDetailsByConfigRule

以下代码示例演示了如何使用 Get-CFGAggregateComplianceDetailsByConfigRule。

适用于 PowerShell V4 的工具

示例 1：此示例返回评估结果，为给定账户、聚合器、区域和 AWS 配置规则的 Config 规则 “desired-instance-type” 选择带有资源 ID 和资源类型的输出，这些输出处于 “合规” 状态

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
EvaluationResultQualifier
```

输出：

ConfigRuleName	ResourceId	ResourceType
-----	-----	-----
desired-instance-type	i-0f1bf2f34c5678d12	AWS::EC2::Instance
desired-instance-type	i-0fd12dd3456789123	AWS::EC2::Instance

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAggregateComplianceDetailsByConfigRule](#) 中的。

Get-CFGAggregateConfigRuleComplianceSummary

以下代码示例演示了如何使用 Get-CFGAggregateConfigRuleComplianceSummary。

适用于 PowerShell V4 的工具

示例 1：此示例返回给定聚合器的不合规规则数。

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

输出：

CapExceeded	CappedCount
-----	-----

False	5

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetAggregateConfigRuleComplianceSummary](#) 中的。

Get-CFGAggregateDiscoveredResourceCount

以下代码示例演示了如何使用 Get-CFGAggregateDiscoveredResourceCount。

适用于 PowerShell V4 的工具

示例 1：此示例返回针对区域 us-east-1 筛选的给定聚合器的资源计数。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

输出：

GroupByKey	GroupedResourceCounts	NextToken	TotalDiscoveredResources
-----	-----	-----	-----
	{}		455

示例 2：此示例返回针对给定聚合器的已筛选区域按 RESOURCE_TYPE 分组的资源计数。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
Select-Object -ExpandProperty GroupedResourceCounts
```

输出：

GroupName	ResourceCount
-----	-----
AWS::CloudFormation::Stack	12
AWS::CloudFront::Distribution	1
AWS::CloudTrail::Trail	1
AWS::DynamoDB::Table	1
AWS::EC2::EIP	2
AWS::EC2::FlowLog	2
AWS::EC2::InternetGateway	4
AWS::EC2::NatGateway	2


```

AWS::EC2::NetworkAcl          4
AWS::EC2::NetworkInterface    12
AWS::EC2::RouteTable          13
AWS::EC2::SecurityGroup       18
AWS::EC2::Subnet              16
AWS::EC2::VPC                 4
AWS::EC2::VPCEndpoint         2
AWS::EC2::VPCPeeringConnection 1
AWS::IAM::Group               2
AWS::IAM::Policy              51
AWS::IAM::Role                78
AWS::IAM::User                7
AWS::Lambda::Function         3
AWS::RDS::DBSecurityGroup     1
AWS::S3::Bucket               3
AWS::SSM::AssociationCompliance 107
AWS::SSM::ManagedInstanceInventory 108

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAggregateDiscoveredResourceCounts](#) 中的。

Get-CFGAggregateDiscoveredResourceList

以下代码示例演示了如何使用 `Get-CFGAggregateDiscoveredResourceList`。

适用于 PowerShell V4 的工具

示例 1：此示例返回在“Ireland”聚合器中聚合的给定资源类型的资源标识符。有关资源类型列表，请查看 https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TConfigServiceResourceType.html&t ConfigService ocid=Amazon_。ResourceType

```

Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)

```

输出：

```

ResourceId      : arn:aws:autoscaling:eu-
west-1:123456789012:autoScalingGroup:12e3b4fc-1234-1234-
a123-1d2ba3c45678:autoScalingGroupName/asg-1
ResourceName    : asg-1
ResourceType    : AWS::AutoScaling::AutoScalingGroup
SourceAccountId : 123456789012

```

```
SourceRegion      : eu-west-1
```

示例 2：此示例返回使用区域 us-east-1 进行筛选的给定聚合器的名为“default”的资源类型 **AwsEC2SecurityGroup**。

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

输出：

```
ResourceId        : sg-01234bd5dbfa67c89
ResourceName      : default
ResourceType      : AWS::EC2::SecurityGroup
SourceAccountId   : 123456789102
SourceRegion      : us-east-1

ResourceId        : sg-0123a4ebbf56789be
ResourceName      : default
ResourceType      : AWS::EC2::SecurityGroup
SourceAccountId   : 123456789102
SourceRegion      : us-east-1

ResourceId        : sg-4fc1d234
ResourceName      : default
ResourceType      : AWS::EC2::SecurityGroup
SourceAccountId   : 123456789102
SourceRegion      : us-east-1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListAggregateDiscoveredResources](#) 中的。

Get-CFGAggregateResourceConfig

以下代码示例演示了如何使用 Get-CFGAggregateResourceConfig。

适用于 PowerShell V4 的工具

示例 1：此示例返回给定聚合资源的配置项并展开配置。

```
(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
```

```
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json
```

输出：

```
{"description":"default VPC security group","groupName":"default","ipPermissions":
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[{"groupId":"sg-4fc1d234","userId":"123456789012"}],"ipv4Ranges":
[],"ipRanges":[]},{ "fromPort":3389,"ipProtocol":"tcp","ipv6Ranges":
[],"prefixListIds":[],"toPort":3389,"userIdGroupPairs":[],"ipv4Ranges":
[{"cidrIp":"54.240.197.224/29","description":"office subnet"},
{"cidrIp":"72.21.198.65/32","description":"home pc"}],"ipRanges":
["54.240.197.224/29","72.21.198.65/32"]}], "ownerId":"123456789012","groupId":"sg-4fc1d234",
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[],"ipv4Ranges":[{"cidrIp":"0.0.0.0/0"}],"ipRanges":["0.0.0.0/0"]}], "tags":
[], "vpcId":"vpc-2d1c2e34"}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V4\)](#) 中的 [GetAggregateResourceconfig-service](#)。

Get-CFGAggregateResourceConfigBatch

以下代码示例演示了如何使用 Get-CFGAggregateResourceConfigBatch。

适用于 PowerShell V4 的工具

示例 1：此示例获取给定聚合器中存在的（已标识）资源的当前配置项。

```
$resIdentifier=[Amazon.ConfigService.Model.AggregateResourceIdentifier]@{
    ResourceId= "i-012e3cb4df567e8aa"
    ResourceName = "arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa"
    ResourceType = [Amazon.ConfigService.ResourceType]::AWSEC2Instance
    SourceAccountId = "123456789012"
    SourceRegion = "eu-west-1"
}

Get-CFGAggregateResourceConfigBatch -ResourceIdentifier $resIdentifier -
ConfigurationAggregatorName raju
```

输出：

```
BaseConfigurationItems UnprocessedResourceIdentifiers
-----
{} {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V4) 中的 [BatchGetAggregateResourceconfig-service](#)。

Get-CFGAggregationAuthorizationList

以下代码示例演示了如何使用 Get-CFGAggregationAuthorizationList。

适用于 PowerShell V4 的工具

示例 1：此示例检索授予聚合器的授权。

```
Get-CFGAggregationAuthorizationList
```

输出：

```
AggregationAuthorizationArn
  AuthorizedAccountId AuthorizedAwsRegion CreationTime
-----
arn:aws:config-service:eu-west-1:123456789012:aggregation-
authorization/123456789012/eu-west-1 123456789012 eu-west-1
8/26/2019 12:55:27 AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeAggregationAuthorizations](#) 中的。

Get-CFGComplianceByConfigRule

以下代码示例演示了如何使用 Get-CFGComplianceByConfigRule。

适用于 PowerShell V4 的工具

示例 1：此示例检索规则的合规性详细信息，该规则当前没有评估结果 ebs-optimized-instance，因此它返回 INSUFKIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

输出：

```
ComplianceContributorCount ComplianceType
-----
INSUFFICIENT_DATA
```

示例 2：此示例返回规则 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK 的不合规资源数量。

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

输出：

```
CapExceeded CappedCount
-----
False      2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeComplianceByConfigRule](#) 中的。

Get-CFGComplianceByResource

以下代码示例演示了如何使用 Get-CFGComplianceByResource。

适用于 PowerShell V4 的工具

示例 1：此示例检查“COMPLIANT”合规类型的 **AWS::SSM::ManagedInstanceInventory** 资源类型。

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
AWS::SSM::ManagedInstanceInventory
```

输出：

```
Compliance                      ResourceId                      ResourceType
-----

```

```
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
AWS::SSM::ManagedInstanceInventory
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeComplianceByResource](#) 中的。

Get-CFGComplianceDetailsByConfigRule

以下代码示例演示了如何使用 Get-CFGComplianceDetailsByConfigRule。

适用于 PowerShell V4 的工具

示例 1：此示例获取规则的评估结果 access-keys-rotated并返回按合规类型分组的输出

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-Object ComplianceType
```

输出：

Count	Name	Group
-----	-----	-----
2	COMPLIANT	{Amazon.ConfigService.Model.EvaluationResult, Amazon.ConfigService.Model.EvaluationResult}
5	NON_COMPLIANT	{Amazon.ConfigService.Model.EvaluationResult, Amazon.ConfigService.Model.EvaluationResult, Amazon.ConfigService.Model.EvaluationRes...

示例 2：此示例查询合规资源的规则 access-keys-rotated的合规性详细信息。

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated -ComplianceType COMPLIANT | ForEach-Object {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

输出：

ConfigRuleName	ResourceId	ResourceType
-----	-----	-----

```
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetComplianceDetailsByConfigRule](#) 中的。

Get-CFGComplianceDetailsByResource

以下代码示例演示了如何使用 Get-CFGComplianceDetailsByResource。

适用于 PowerShell V4 的工具

示例 1：此示例提供给定资源的评估结果。

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
'AWS::IAM::User'
```

输出：

```
Annotation           :
ComplianceType       : COMPLIANT
ConfigRuleInvokedTime : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier : Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime   : 8/25/2019 11:34:56 PM
ResultToken          :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetComplianceDetailsByResource](#) 中的。

Get-CFGComplianceSummaryByConfigRule

以下代码示例演示了如何使用 Get-CFGComplianceSummaryByConfigRule。

适用于 PowerShell V4 的工具

示例 1：此示例返回不合规的 Config 规则的数量。

```
Get-CFGComplianceSummaryByConfigRule -Select
ComplianceSummary.NonCompliantResourceCount
```

输出：

```
CapExceeded CappedCount
-----
False          9
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetComplianceSummaryByConfigRule](#) 中的。

Get-CFGComplianceSummaryByResourceType

以下代码示例演示了如何使用 Get-CFGComplianceSummaryByResourceType。

适用于 PowerShell V4 的工具

示例 1：此示例返回合规或不合规资源的数量，并将输出转换为 json。

```
Get-CFGComplianceSummaryByResourceType -Select
ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetComplianceSummaryByResourceType](#) 中的。

Get-CFGConfigRule

以下代码示例演示了如何使用 Get-CFGConfigRule。

适用于 PowerShell V4 的工具

示例 1：此示例列出账户的配置规则以及选定属性。


```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
ConfigRuleState
```

输出：

ConfigRuleName	ConfigRuleId	ConfigRuleArn
-----	-----	-----
ALB_REDIRECTION_CHECK	config-rule-12iyn3	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3	ACTIVE	
access-keys-rotated	config-rule-aospfr	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr	ACTIVE	
autoscaling-group-elb-healthcheck-required	config-rule-cn1f2x	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x	ACTIVE	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeConfigRules](#) 中的。

Get-CFGConfigRuleEvaluationStatus

以下代码示例演示了如何使用 Get-CFGConfigRuleEvaluationStatus。

适用于 PowerShell V4 的工具

示例 1：此示例返回给定配置规则的状态信息。

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

输出：

ConfigRuleArn	: arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk	
ConfigRuleId	: config-rule-kvq1wk
ConfigRuleName	: root-account-mfa-enabled
FirstActivatedTime	: 8/27/2019 8:05:17 AM
FirstEvaluationStarted	: True
LastErrorCode	:
LastErrorMessage	:
LastFailedEvaluationTime	: 1/1/0001 12:00:00 AM

```

LastFailedInvocationTime      : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime  : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime  : 12/13/2019 8:12:03 AM

ConfigRuleArn                 : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                  : config-rule-z1s23b
ConfigRuleName                : vpc-flow-logs-enabled
FirstActivatedTime            : 8/14/2019 6:23:44 AM
FirstEvaluationStarted        : True
LastErrorCode                 :
LastErrorMessage              :
LastFailedEvaluationTime      : 1/1/0001 12:00:00 AM
LastFailedInvocationTime      : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime  : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime  : 12/13/2019 7:12:01 AM

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConfigRuleEvaluationStatus](#) 中的。

Get-CFGConfigurationAggregatorList

以下代码示例演示了如何使用 Get-CFGConfigurationAggregatorList。

适用于 PowerShell V4 的工具

示例 1：此示例返回区域/账户的所有聚合器。

```
Get-CFGConfigurationAggregatorList
```

输出：

```

AccountAggregationSources      :
  {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn     : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabca1me
ConfigurationAggregatorName    : IrelandMaster
CreationTime                   : 8/25/2019 11:42:39 PM
LastUpdatedTime                : 8/25/2019 11:42:39 PM
OrganizationAggregationSource  :

AccountAggregationSources      : {}

```

```
ConfigurationAggregatorArn      : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-qubqabcd
ConfigurationAggregatorName     : raju
CreationTime                    : 8/11/2019 8:39:25 AM
LastUpdatedTime                 : 8/11/2019 8:39:25 AM
OrganizationAggregationSource   :
    Amazon.ConfigService.Model.OrganizationAggregationSource
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConfigurationAggregators](#) 中的。

Get-CFGConfigurationAggregatorSourcesStatus

以下代码示例演示了如何使用 Get-CFGConfigurationAggregatorSourcesStatus。

适用于 PowerShell V4 的工具

示例 1：此示例显示给定聚合器中源的所请求字段。

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

输出：

SourceType	LastUpdateStatus	LastUpdateTime	SourceId
-----	-----	-----	-----
ORGANIZATION	SUCCEEDED	12/31/2019 7:45:06 AM	Organization
ACCOUNT	SUCCEEDED	12/31/2019 7:09:38 AM	612641234567
ACCOUNT	SUCCEEDED	12/31/2019 7:12:53 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:18:10 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:25:17 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:25:49 AM	612641234567
ACCOUNT	SUCCEEDED	12/31/2019 7:26:11 AM	612641234567

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConfigurationAggregatorSourcesStatus](#) 中的。

Get-CFGConfigurationRecorder

以下代码示例演示了如何使用 Get-CFGConfigurationRecorder。

适用于 PowerShell V4 的工具

示例 1：此示例返回配置记录器的详细信息。

```
Get-CFGConfigurationRecorder | Format-List
```

输出：

```
Name           : default
RecordingGroup  : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConfigurationRecorders](#) 中的。

Get-CFGConfigurationRecorderStatus

以下代码示例演示了如何使用 Get-CFGConfigurationRecorderStatus。

适用于 PowerShell V4 的工具

示例 1：此示例返回配置记录器的状态。

```
Get-CFGConfigurationRecorderStatus
```

输出：

```
LastErrorCode      :
LastErrorMessage   :
LastStartTime      : 10/11/2019 10:13:51 AM
LastStatus         : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime       : 10/11/2019 10:13:46 AM
Name               : default
Recording          : True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConfigurationRecorderStatus](#) 中的。

Get-CFGConformancePack

以下代码示例演示了如何使用 Get-CFGConformancePack。

适用于 PowerShell V4 的工具

示例 1：此示例列出所有一致性包。

```
Get-CFGConformancePack
```

输出：

```
ConformancePackArn      : arn:aws:config:eu-west-1:123456789012:conformance-
pack/dono/conformance-pack-p0acq8bpz
ConformancePackId       : conformance-pack-p0acabcde
ConformancePackInputParameters : {}
ConformancePackName     : dono
CreatedBy               :
DeliveryS3Bucket        : kt-ps-examples
DeliveryS3KeyPrefix     :
LastUpdateRequestedTime : 12/31/2019 8:45:31 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeConformancePacks](#) 中的。

Get-CFGDeliveryChannel

以下代码示例演示了如何使用 Get-CFGDeliveryChannel。

适用于 PowerShell V4 的工具

示例 1：此示例检索该地区的传输通道并显示详细信息。

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
S3KeyPrefix,
@{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

输出：

Name	S3BucketName	S3KeyPrefix	DeliveryFrequency
----	-----	-----	-----

```
default config-bucket-NA my TwentyFour_Hours
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeDeliveryChannels](#) 中的。

Get-CFGResourceTag

以下代码示例演示了如何使用 Get-CFGResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例列出给定资源的关联标签

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

输出：

```
Key      Value
---      -
Version 1.3
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListTagsForResource](#) 中的。

Remove-CFGConformancePack

以下代码示例演示了如何使用 Remove-CFGConformancePack。

适用于 PowerShell V4 的工具

示例 1：此示例移除给定的一致性包，以及该包的所有规则、补救措施和评估结果。

```
Remove-CFGConformancePack -ConformancePackName dono
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-CFGConformancePack (DeleteConformancePack)" on
target "dono".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteConformancePack](#) 中的。

Write-CFGConformancePack

以下代码示例演示了如何使用 Write-CFGConformancePack。

适用于 PowerShell V4 的工具

示例 1：此示例创建一致性包，并从给定的 yaml 文件中获取模板。

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutConformancePack](#) 中的。

Write-CFGDeliveryChannel

以下代码示例演示了如何使用 Write-CFGDeliveryChannel。

适用于 PowerShell V4 的工具

示例 1：此示例更改现有传输通道的 deliveryFrequency 属性。

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutDeliveryChannel](#) 中的。

使用 V4 工具的 De PowerShell vice Farm 示例

以下代码示例向您展示了如何使用带有 Device Farm 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

New-DFUpload

以下代码示例演示了如何使用 New-DFUpload。

适用于 PowerShell V4 的工具

示例 1：此示例为安卓应用创建 Dev AWS ice Farm 上传。您可以从“新建列表” DFProject 或“获取列表”的输出中获取项目 ARN。DFProject使用新DFUpload 输出中的签名 URL 将文件上传到 Device Farm。

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn  
"arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-  
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateUpload](#) 中的。

Directory Service 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 Directory Service 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-DSIpRoute

以下代码示例演示了如何使用 Add-DSIpRoute。

适用于 PowerShell V4 的工具

示例 1：此命令移除分配给指定 Directory-id 的资源标签

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -  
UpdateSecurityGroupForDirectoryController $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddIpRoutes](#) 中的。

Add-DSResourceTag

以下代码示例演示了如何使用 Add-DSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此命令将资源标签添加到指定的 Directory-id

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddTagsToResource](#) 中的。

Approve-DSTrust

以下代码示例演示了如何使用 Approve-DSTrust。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 Trustid 调用 AWS 目录服务 VerifyTrust API 操作。

```
Approve-DSTrust -TrustId t-9067157123
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [VerifyTrust](#) 中的。

Confirm-DSSharedDirectory

以下代码示例演示了如何使用 Confirm-DSSharedDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例接受目录所有者发送的目录共享请求 AWS 账户。

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

输出：

```
CreatedDateTime      : 12/30/2019 4:20:27 AM
LastUpdatedDateTime : 12/30/2019 4:21:40 AM
OwnerAccountId       : 123456781234
OwnerDirectoryId     : d-123456ijkl
SharedAccountId       : 123456784321
SharedDirectoryId    : d-9067012345
ShareMethod          :
ShareNotes           : This is test sharing
ShareStatus          : Sharing
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AcceptSharedDirectory](#) 中的。

Connect-DSDirectory

以下代码示例演示了如何使用 Connect-DSDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例创建 AD Connector 以连接到本地目录。

```
Connect-DSDirectory -Name contoso.com -ConnectSettings_CustomerUserName
Administrator -Password $Password -ConnectSettings_CustomerDnsIp 172.31.36.96
-ShortName CONTOSO -Size Small -ConnectSettings_VpcId vpc-123459da -
ConnectSettings_SubnetId subnet-1234ccaa, subnet-5678ffbb
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ConnectDirectory](#) 中的。

Deny-DSSharedDirectory

以下代码示例演示了如何使用 Deny-DSSharedDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例拒绝从目录所有者账户中发送的目录共享请求。

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

输出：

```
d-9067012345
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RejectSharedDirectory](#) 中的。

Disable-DSDirectoryShare

以下代码示例演示了如何使用 Disable-DSDirectoryShare。

适用于 PowerShell V4 的工具

示例 1：此示例停止目录所有者和使用者账户之间的目录共享。

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -  
UnshareTarget_Type ACCOUNT
```

输出：

```
d-9067012345
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UnshareDirectory](#) 中的。

Disable-DSLDPAPS

以下代码示例演示了如何使用 Disable-DSLDPAPS。

适用于 PowerShell V4 的工具

示例 1：此示例停用对指定目录的 LDAP 安全调用。

```
Disable-DSLDAPs -DirectoryId d-123456ijkl -Type Client
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [DisableLDAPs](#)。

Disable-DSRadius

以下代码示例演示了如何使用 Disable-DSRadius。

适用于 PowerShell V4 的工具

示例 1：此示例禁用为 AD Connector 或 Microsoft AD 目录配置的 RADIUS 服务器。

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DisableRadius](#) 中的。

Disable-DSSso

以下代码示例演示了如何使用 Disable-DSSso。

适用于 PowerShell V4 的工具

示例 1：此示例为目录禁用单点登录。

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DisableSso](#) 中的。

Enable-DSDirectoryShare

以下代码示例演示了如何使用 Enable-DSDirectoryShare。

适用于 PowerShell V4 的工具

示例 1：此示例使用 Handshake 方法与另一个 AWS 账户共享您 AWS 账户中的指定目录。

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -  
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

输出：

```
d-9067012345
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ShareDirectory](#) 中的。

Enable-DSLDAPS

以下代码示例演示了如何使用 Enable-DSLDAPS。

适用于 PowerShell V4 的工具

示例 1：此示例激活特定目录的开关以始终使用 LDAP 安全调用。

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [EnableLDAPS](#)。

Enable-DSRadius

以下代码示例演示了如何使用 Enable-DSRadius。

适用于 PowerShell V4 的工具

示例 1：此示例使用为 AD Connector 或 Microsoft AD 目录提供的 RADIUS 服务器配置启用多重身份验证 (MFA)。

```
Enable-DSRadius -DirectoryId d-123456ijkl  
-RadiusSettings_AuthenticationProtocol PAP  
-RadiusSettings_DisplayLabel Radius  
-RadiusSettings_RadiusPort 1812  
-RadiusSettings_RadiusRetry 4  
-RadiusSettings_RadiusServer 10.4.185.113  
-RadiusSettings_RadiusTimeout 50  
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableRadius](#) 中的。

Enable-DSSso

以下代码示例演示了如何使用 Enable-DSSso。

适用于 PowerShell V4 的工具

示例 1：此示例为目录启用单点登录。

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableSso](#) 中的。

Get-DSCertificate

以下代码示例演示了如何使用 Get-DSCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例显示有关为受保护的 LDAP 连接注册的证书的信息。

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

输出：

```
CertificateId      : c-906731e34f
CommonName         : contoso-EC2AMAZ-CTGG2NM-CA
ExpiryDateTime     : 4/15/2025 6:34:15 PM
RegisteredDateTime : 4/15/2020 6:38:56 PM
State              : Registered
StateReason        : Certificate registered successfully.
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeCertificate](#) 中的。

Get-DSCertificateList

以下代码示例演示了如何使用 Get-DSCertificateList。

适用于 PowerShell V4 的工具

示例 1：此示例对于指定的目录，列出为受保护的 LDAP 连接注册的所有证书。

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

输出：

CertificateId	CommonName	ExpiryDateTime	State
-----	-----	-----	-----
c-906731e34f	contoso-EC2AMAZ-CTGG2NM-CA	4/15/2025 6:34:15 PM	Registered

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListCertificates](#) 中的。

Get-DSConditionalForwarder

以下代码示例演示了如何使用 Get-DSConditionalForwarder。

适用于 PowerShell V4 的工具

示例 1：此命令获取给定 Directory-id 的所有已配置条件转发服务器。

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

输出：

DnsIpAddress	RemoteDomainName	ReplicationScope
-----	-----	-----
{172.31.77.239}	contoso.com	Domain

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeConditionalForwarders](#) 中的。

Get-DSDirectory

以下代码示例演示了如何使用 Get-DSDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关属于此账户的目录的信息。

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

输出：

DirectoryId	Name	DnsIpAddrs	Type
d-123456abcd	abcd.example.com	{172.31.74.189, 172.31.13.145}	SimpleAD
d-123456efgh	wifi.example.com	{172.31.16.108, 172.31.10.56}	ADConnector
d-123456ijkl	lan2.example.com	{172.31.10.56, 172.31.16.108}	MicrosoftAD

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeDirectories](#) 中的。

Get-DSDirectoryLimit

以下代码示例演示了如何使用 Get-DSDirectoryLimit。

适用于 PowerShell V4 的工具

示例 1：此示例显示 us-east-1 区域的目录限制信息。

```
Get-DSDirectoryLimit -Region us-east-1
```

输出：

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit       : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetDirectoryLimits](#) 中的。

Get-DSDomainControllerList

以下代码示例演示了如何使用 Get-DSDomainControllerList。

适用于 PowerShell V4 的工具

示例 1：此命令获取为所提及的 directory-id 启动的域控制器的详细列表

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

输出：

```
AvailabilityZone      : us-east-1b
DirectoryId           : d-123456ijkl
DnsIpAddr             : 172.31.16.108
DomainControllerId    : dc-1234567aa6
LaunchTime            : 4/4/2019 4:53:43 AM
Status                : Active
StatusLastUpdatedDateTime : 4/24/2019 1:37:54 PM
StatusReason          :
SubnetId              : subnet-1234kkaa
VpcId                 : vpc-123459d

AvailabilityZone      : us-east-1d
DirectoryId           : d-123456ijkl
DnsIpAddr             : 172.31.10.56
DomainControllerId    : dc-1234567aa7
LaunchTime            : 4/4/2019 4:53:43 AM
Status                : Active
StatusLastUpdatedDateTime : 4/4/2019 5:14:31 AM
StatusReason          :
SubnetId              : subnet-5678ffbb
VpcId                 : vpc-123459d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeDomainControllers](#) 中的。

Get-DSEventTopic

以下代码示例演示了如何使用 Get-DSEventTopic。

适用于 PowerShell V4 的工具

示例 1：此命令显示已配置 SNS 主题的信息，以便在目录状态发生变化时得到通知。

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

输出：

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeEventTopics](#) 中的。

Get-DSIpRouteList

以下代码示例演示了如何使用 Get-DSIpRouteList。

适用于 PowerShell V4 的工具

示例 1：此命令获取在目录 IP 路由中配置的公有 IP 地址块

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

输出：

```
AddedDateTime      : 12/13/2019 12:27:22 PM
CidrIp             : 203.0.113.5/32
Description         : Public IP of On-Prem DNS Server
DirectoryId        : d-123456ijkl
IpRouteStatusMsg    : Added
IpRouteStatusReason :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListIpRoutes](#) 中的。

Get-DSLDAOSSetting

以下代码示例演示了如何使用 Get-DSLDAOSSetting。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定目录的 LDAP 安全性状态。

```
Get-DSLDAOSSetting -DirectoryId d-123456ijkl
```

输出：

```
LastUpdatedDateTime  LDAPStatus LDAPStatusReason
-----
4/15/2020 6:51:03 PM Enabled      LDAPS is enabled successfully.
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) [LDAPSSettings](#) 中的[描述](#)。

Get-DSLogSubscriptionList

以下代码示例演示了如何使用 Get-DSLogSubscriptionList。

适用于 PowerShell V4 的工具

示例 1：此命令获取指定 directory-id 的日志订阅信息

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

输出：

```
DirectoryId  LogGroupName
SubscriptionCreatedDateTime
-----
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListLogSubscriptions](#) 中的。

Get-DSResourceTag

以下代码示例演示了如何使用 Get-DSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此命令获取指定目录的所有标签。

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

输出：

```
Key      Value
---      -
myTag    myTagValue
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListTagsForResource](#) 中的。

Get-DSSchemaExtension

以下代码示例演示了如何使用 Get-DSSchemaExtension。

适用于 PowerShell V4 的工具

示例 1：此示例列出应用于某一 Microsoft AD 目录的所有架构扩展。

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

输出：

```
Description           : ManagedADSchemaExtension
DirectoryId            : d-123456ijkl
EndTime                : 4/12/2020 10:30:49 AM
SchemaExtensionId      : e-9067306643
SchemaExtensionStatus  : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartDateTime          : 4/12/2020 10:28:42 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListSchemaExtensions](#) 中的。

Get-DSSharedDirectory

以下代码示例演示了如何使用 Get-DSSharedDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例获取您 AWS 账户的共享目录

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

输出：

```
CreatedDateTime      : 12/30/2019 4:34:37 AM
LastUpdatedDateTime : 12/30/2019 4:35:22 AM
OwnerAccountId       : 123456781234
OwnerDirectoryId     : d-123456ijkl
SharedAccountId      : 123456784321
SharedDirectoryId    : d-9067012345
ShareMethod          : HANDSHAKE
ShareNotes           : This is a test Sharing
ShareStatus          : Shared
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSharedDirectories](#) 中的。

Get-DSSnapshot

以下代码示例演示了如何使用 Get-DSSnapshot。

适用于 PowerShell V4 的工具

示例 1：此命令获取有关属于此账户的指定目录快照的信息。

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

输出：

```
DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bd1234
StartTime   : 12/13/2019 6:33:01 PM
Status      : Completed
Type        : Auto

DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bb4321
StartTime   : 12/9/2019 9:48:11 PM
Status      : Completed
```

Type : Auto

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeSnapshots](#) 中的。

Get-DSSnapshotLimit

以下代码示例演示了如何使用 Get-DSSnapshotLimit。

适用于 PowerShell V4 的工具

示例 1：此命令获取指定目录的手动快照限制。

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

输出：

ManualSnapshotsCurrentCount	ManualSnapshotsLimit	ManualSnapshotsLimitReached
0	5	False

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetSnapshotLimits](#) 中的。

Get-DSTrust

以下代码示例演示了如何使用 Get-DSTrust。

适用于 PowerShell V4 的工具

示例 1：此命令获取为指定 directory-id 创建的信任关系的信息。

```
Get-DSTrust -DirectoryId d-123456abcd
```

输出：

CreatedDateTime	: 7/5/2019 4:55:42 AM
DirectoryId	: d-123456abcd
LastUpdatedDateTime	: 7/5/2019 4:56:04 AM

```
RemoteDomainName      : contoso.com
SelectiveAuth          : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection         : One-Way: Incoming
TrustId                : t-9067157123
TrustState              : Created
TrustStateReason        :
TrustType               : Forest
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrusts](#) 中的。

New-DSAlias

以下代码示例演示了如何使用 New-DSAlias。

适用于 PowerShell V4 的工具

示例 1：此命令为目录创建一个别名并将此别名分配给指定的 directory-id。

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

输出：

```
Alias      DirectoryId
-----
myorgname d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateAlias](#) 中的。

New-DSComputer

以下代码示例演示了如何使用 New-DSComputer。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个新的 Active Directory 计算机对象。

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
$Password
```

输出：

```
ComputerAttributes      ComputerId
-----
-----
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
ADMemberServer
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateComputer](#) 中的。

New-DSConditionalForwarder

以下代码示例演示了如何使用 New-DSConditionalForwarder。

适用于 PowerShell V4 的工具

示例 1：此示例在指定的 Di AWS rectory-ID 中创建条件转发器。

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress
172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateConditionalForwarder](#) 中的。

New-DSDirectory

以下代码示例演示了如何使用 New-DSDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个新的 Simple AD 目录。

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateDirectory](#) 中的。

New-DSLogSubscription

以下代码示例演示了如何使用 New-DSLogSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个订阅，用于将实时 Directory Service 域控制器安全日志转发到您的指定的 Amazon CloudWatch 日志组 AWS 账户。

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/d-123456ijkl-lan2.example.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateLogSubscription](#) 中的。

New-DSMicrosoftAD

以下代码示例演示了如何使用 New-DSMicrosoftAD。

适用于 PowerShell V4 的工具

示例 1：此示例在中创建了新的 Microsoft 广告目录 AWS Cloud。

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的 A [CreateMicrosoftD](#)。

New-DSSnapshot

以下代码示例演示了如何使用 New-DSSnapshot。

适用于 PowerShell V4 的工具

示例 1：此示例创建目录快照

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateSnapshot](#) 中的。

New-DSTrust

以下代码示例演示了如何使用 New-DSTrust。

适用于 PowerShell V4 的工具

示例 1：此示例在您的托管 Microsoft AD 目录和现有的本地 AWS Microsoft Active Directory 之间创建了全森林范围的双向信任。

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
172.31.36.96
```

输出：

```
t-9067157123
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTrust](#) 中的。

Register-DSCertificate

以下代码示例演示了如何使用 Register-DSCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例为受保护的 LDAP 连接注册证书。

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

输出：

```
c-906731e350
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterCertificate](#) 中的。

Register-DSEventTopic

以下代码示例演示了如何使用 Register-DSEventTopic。

适用于 PowerShell V4 的工具

示例 1：此示例将目录作为发布者与 SNS 主题相关联。

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterEventTopic](#) 中的。

Remove-DSConditionalForwarder

以下代码示例演示了如何使用 Remove-DSConditionalForwarder。

适用于 PowerShell V4 的工具

示例 1：此示例删除已为您的 D AWS irecotry 设置的条件转发器。

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName  
contoso.com
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteConditionalForwarder](#) 中的。

Remove-DSDirectory

以下代码示例演示了如何使用 Remove-DSDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例删除 AWS 目录服务目录（简单 AD/Microsoft AD/AD 连接器）

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteDirectory](#) 中的。

Remove-DSIpRoute

以下代码示例演示了如何使用 Remove-DSIpRoute。

适用于 PowerShell V4 的工具

示例 1：此命令从 Directory-id 的已配置 IP 路由中移除指定的 IP。

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveIpRoutes](#) 中的。

Remove-DSLogSubscription

以下代码示例演示了如何使用 Remove-DSLogSubscription。

适用于 PowerShell V4 的工具

示例 1：此命令移除指定 Directory-id 的日志订阅

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteLogSubscription](#) 中的。

Remove-DSResourceTag

以下代码示例演示了如何使用 Remove-DSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此命令移除分配给指定 Directory-id 的资源标签

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveTagsFromResource](#) 中的。

Remove-DSSnapshot

以下代码示例演示了如何使用 Remove-DSSnapshot。

适用于 PowerShell V4 的工具

示例 1：此示例移除手动创建的快照。

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteSnapshot](#) 中的。

Remove-DSTrust

以下代码示例演示了如何使用 Remove-DSTrust。

适用于 PowerShell V4 的工具

示例 1：此示例删除了您的 AWS 托管 AD 目录与外部域之间现有的信任关系。

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

输出：

```
t-9067157123
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTrust](#) 中的。

Reset-DSUserPassword

以下代码示例演示了如何使用 Reset-DSUserPassword。

适用于 PowerShell V4 的工具

示例 1：此示例重置了在 Microsoft AD AWS 托管或 Simple AD Directory ADUser 中命名的 Active Direcotry 用户的密码

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword $Password
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResetUserPassword](#) 中的。

Restore-DSFromSnapshot

以下代码示例演示了如何使用 Restore-DSFromSnapshot。

适用于 PowerShell V4 的工具

示例 1：此示例使用现有的目录快照还原目录。

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RestoreFromSnapshot](#) 中的。

Set-DSDomainControllerCount

以下代码示例演示了如何使用 Set-DSDomainControllerCount。

适用于 PowerShell V4 的工具

示例 1：此示例将指定 directory-id 的域控制器数量设置为 3。

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateNumberOfDomainControllers](#) 中的。

Start-DSSchemaExtension

以下代码示例演示了如何使用 Start-DSSchemaExtension。

适用于 PowerShell V4 的工具

示例 1：此示例对 Microsoft AD 目录应用架构扩展。

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

输出：

```
e-9067306643
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [StartSchemaExtension](#) 中的。

Stop-DSSchemaExtension

以下代码示例演示了如何使用 Stop-DSSchemaExtension。

适用于 PowerShell V4 的工具

示例 1：此示例对 Microsoft AD 目录取消正在进行的架构扩展。

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CancelSchemaExtension](#) 中的。

Unregister-DSCertificate

以下代码示例演示了如何使用 Unregister-DSCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例从系统中删除为受保护的 LDAP 连接注册的证书。

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeregisterCertificate](#) 中的。

Unregister-DSEventTopic

以下代码示例演示了如何使用 Unregister-DSEventTopic。

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的目录作为指定 SNS 主题的发布者。

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeregisterEventTopic](#) 中的。

Update-DSConditionalForwarder

以下代码示例演示了如何使用 Update-DSConditionalForwarder。

适用于 PowerShell V4 的工具

示例 1：此示例更新已为您的 AWS 目录设置的条件转发器。

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr  
172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateConditionalForwarder](#) 中的。

Update-DSRadius

以下代码示例演示了如何使用 Update-DSRadius。

适用于 PowerShell V4 的工具

示例 1：此示例更新 AD Connector 或 Microsoft AD 目录的 RADIUS 服务器信息。

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateRadius](#) 中的。

Update-DSTrust

以下代码示例演示了如何使用 Update-DSTrust。

适用于 PowerShell V4 的工具

示例 1：此示例将指定信任 ID 的 SelectiveAuth 参数从“已禁用”更新为“启用”。

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```


输出：

```
RequestId                                TrustId
-----                                -
138864a7-c9a8-4ad1-a828-eae479e85b45  t-9067157123
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateTrust](#) 中的。

AWS DMS 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS DMS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

New-DMSReplicationTask

以下代码示例演示了如何使用 New-DMSReplicationTask。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个新的 AWS 数据库迁移服务 (Database Migration Service) 复制任务，该任务使用 CdcStartTime 代替 CdcStartPosition。设置 MigrationType 为 full-load-and-cdc ""，这意味着目标表必须为空。新任务使用一个标签进行标记，该标签的键为 Stage，键值为 Test。有关此 cmdlet 使用的值的更多信息，请参阅《数据库迁移服务用户指南》[https://docs.aws.amazon.com/dms/latest/userguide/CHAP中的创建任务 \(_tasks.creating.html\)](https://docs.aws.amazon.com/dms/latest/userguide/CHAP中的创建任务 (_tasks.creating.html))。AWS

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"
```

```
-CdcStartTime "2019-08-08T12:12:12" `
-CdcStopPosition "server_time:2019-08-09T12:12:12" `
-MigrationType "full-load-and-cdc" `
-ReplicationTaskIdentifier "task1" `
-ReplicationTaskSetting "" `
-SourceEndpointArn "arn:aws:dms:us-east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE" `
-TableMapping "file:///home/testuser/table-mappings.json" `
-Tag @{"Key"="Stage";"Value"="Test"} `
-TargetEndpointArn "arn:aws:dms:us-east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateReplicationTask](#) 中的。

使用适用于 V4 的工具的 DynamoDB 示例 PowerShell

以下代码示例向您展示了如何使用带有 DynamoDB 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-DDBIndexSchema

以下代码示例演示了如何使用 Add-DDBIndexSchema。

适用于 PowerShell V4 的工具

示例 1：在将 TableSchema 对象写入管道之前，创建一个空 TableSchema 对象并向其添加新的本地二级索引定义。

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

输出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{LastPostDateTime}	{}
{LastPostIndex}	

示例 2：在将对象写回管道之前，向提供的 TableSchema 对象添加新的本地二级索引定义。TableSchema 也可以使用 -Schema 参数提供该 TableSchema 对象。

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

输出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{LastPostDateTime}	{}
{LastPostIndex}	

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的[添加DDBIndex架构](#)。

Add-DDBKeySchema

以下代码示例演示了如何使用 Add-DDBKeySchema。

适用于 PowerShell V4 的工具

示例 1：创建一个空 TableSchema 对象，并在将 TableSchema 对象写入管道之前，使用指定的密钥数据向其添加键和属性定义条目。默认情况下，键类型声明为“HASH”；使用值为“RANGE”的 -KeyType 参数来声明范围键。

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

输出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{ForumName}	{ForumName}
{}	

示例 2：在将对象写入管道之前，向提供的 TableSchema 对象添加新的键和属性定义条目。TableSchema 默认情况下，键类型声明为“HASH”；使用值为“RANGE”的-KeyType 参数来声明范围键。也可以使用-Schema 参数提供该 TableSchema 对象。

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

输出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{ForumName}	{ForumName}
{}	

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的[添加DDBKey架构](#)。

ConvertFrom-DDBItem

以下代码示例演示了如何使用 ConvertFrom-DDBItem。

适用于 PowerShell V4 的工具

示例 1：ConvertFrom-DDBItem 用于将结果 Get-DDBItem 从 D AttributeValues ynamoDB 的哈希表转换为字符串和双精度等常见类型的哈希表。

```
@{
```

```

    SongTitle = 'Somewhere Down The Road'
    Artist     = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem

```

输出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DDBItem 中的 ConvertFrom](#)。

ConvertTo-DDBItem

以下代码示例演示了如何使用 ConvertTo-DDBItem。

适用于 PowerShell V4 的工具

示例 1：将哈希表转换为 DynamoDB 属性值字典的示例。

```

@{
    SongTitle = 'Somewhere Down The Road'
    Artist     = 'No One You Know'
} | ConvertTo-DDBItem

Key      Value
---      -
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue

```

示例 2：将哈希表转换为 DynamoDB 属性值字典的示例。

```

@{
    MyMap      = @{

```

```

        MyString = 'my string'
    }
    MyStringSet = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string'))))
    )
    MyList1      = @('my', 'string')
    MyList2      = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3      = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem

```

输出：

Key	Value
---	----
MyStringSet	Amazon.DynamoDBv2.Model.AttributeValue
MyList1	Amazon.DynamoDBv2.Model.AttributeValue
MyNumericSet	Amazon.DynamoDBv2.Model.AttributeValue
MyList2	Amazon.DynamoDBv2.Model.AttributeValue
MyBinarySet	Amazon.DynamoDBv2.Model.AttributeValue
MyMap	Amazon.DynamoDBv2.Model.AttributeValue
MyList3	Amazon.DynamoDBv2.Model.AttributeValue

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DDBItem 中的 ConvertTo-](#)。

Get-DDBBatchItem

以下代码示例演示了如何使用 Get-DDBBatchItem。

适用于 PowerShell V4 的工具

示例 1：从 DynamoDB 表格“音乐”SongTitle 和“歌曲”中获取带有“Somewhere Down The Road”的物品。

```

$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

```

```
$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-DDBItem
```

输出：

Name	Value
----	-----
Artist	No One You Know
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
CriticRating	10
Genre	Country
Price	1.94
Artist	No One You Know
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
CriticRating	10
Genre	Country
Price	1.94

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [BatchGetItem](#) 中的。

Get-DDBItem

以下代码示例演示了如何使用 Get-DDBItem。

适用于 PowerShell V4 的工具

示例 1：返回带有分区 SongTitle 键和排序键 Artist 的 DynamoDB 项目。

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

输出：

Name	Value
----	----
Genre	Country
SongTitle	Somewhere Down The Road
Price	1.94
Artist	No One You Know
CriticRating	9
AlbumTitle	Somewhat Famous

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetItem](#) 中的。

Get-DDBTable

以下代码示例演示了如何使用 Get-DDBTable。

适用于 PowerShell V4 的工具

示例 1：返回指定表的详细信息。

```
Get-DDBTable -TableName "myTable"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTable](#) 中的。

Get-DDBTableList

以下代码示例演示了如何使用 Get-DDBTableList。

适用于 PowerShell V4 的工具

示例 1：返回所有表的详细信息，自动迭代，直到服务指示不存在其它表。

Get-DDBTableList

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListTables](#) 中的。

Invoke-DDBQuery

以下代码示例演示了如何使用 Invoke-DDBQuery。

适用于 PowerShell V4 的工具

示例 1：调用查询返回指定和艺术家的 DynamoDB 项目。 SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

输出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Query](#)。

Invoke-DDBScan

以下代码示例演示了如何使用 Invoke-DDBScan。

适用于 PowerShell V4 的工具

示例 1：返回 Music 表中的所有项目。

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

输出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
Genre	Country
Artist	No One You Know
Price	1.98
CriticRating	8.4
SongTitle	My Dog Spot
AlbumTitle	Hey Now

示例 2：返回 Music 表中 CriticRating 大于或等于 9 的项目。

```
$scanFilter = @{
    CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
        AttributeValueList = @( @{N = '9'} )
        ComparisonOperator = 'GE'
    }
}
Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

输出：

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9

SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Scan](#)。

New-DDBTable

以下代码示例演示了如何使用 New-DDBTable。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为 Thread 的表，该表的主键由 'ForumName' (键类型哈希) 和 'Subject' (键类型范围) 组成。用于构造表的架构可以通过管道传输到所示的每个 cmdlet 中，也可以使用 -Schema 参数指定。

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

输出：

```
AttributeDefinitions : {ForumName, Subject}
TableName             : Thread
KeySchema              : {ForumName, Subject}
TableStatus           : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes        : 0
ItemCount              : 0
LocalSecondaryIndexes : {}
```

示例 2：此示例创建了一个名为 Thread 的表，该表的主键由 'ForumName' (键类型哈希) 和 'Subject' (键类型范围) 组成。还定义了本地二级索引。本地二级索引的键将根据表上的主哈希键自动设置 (ForumName)。用于构造表的架构可以通过管道传输到所示的每个 cmdlet 中，也可以使用 -Schema 参数指定。

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
```

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

输出：

```
AttributeDefinitions : {ForumName, LastPostDateTime, Subject}
TableName             : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes : {LastPostIndex}
```

示例 3：此示例说明如何使用单个管道创建名为 Thread 的表，该表的主键由 'ForumName' (键类型哈希) 和 'Subject' (键类型范围) 组成，还有一个本地二级索引。如果管道或-DDBKey S DDBIndex chema参数未提供新 TableSchema 对象，则添加架构和添加架构将为您创建一个新对象。

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
    -RangeKeyName "LastPostDateTime" `
    -RangeKeyDataType "S" `
    -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

输出：

```
AttributeDefinitions : {ForumName, LastPostDateTime, Subject}
TableName             : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes : {LastPostIndex}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTable](#) 中的。

New-DDBTableSchema

以下代码示例演示了如何使用 New-DDBTableSchema。

适用于 PowerShell V4 的工具

示例 1：创建一个准备接受密钥和索引定义的空 TableSchema 对象，用于创建新的 Amazon DynamoDB 表。返回的对象可以通过管道传输到 Add-DDBKey Schema、Add-Schema 和 New-DDBIndex cmdlet 中，也可以使用每个 DDBTable cmdlet 上的 -Schema 参数传递给它们。

```
New-DDBTableSchema
```

输出：

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

{}	{}
{}	

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) 中的 [新DDBTable架构](#)。

Remove-DDBItem

以下代码示例演示了如何使用 Remove-DDBItem。

适用于 PowerShell V4 的工具

示例 1：移除与提供的键匹配的 DynamoDB 项目。

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteItem](#) 中的。

Remove-DDBTable

以下代码示例演示了如何使用 Remove-DDBTable。

适用于 PowerShell V4 的工具

示例 1：删除指定的表。在操作继续之前，系统会提示您进行确认。

```
Remove-DDBTable -TableName "myTable"
```

示例 2：删除指定的表。在操作继续之前，系统不会提示您进行确认。

```
Remove-DDBTable -TableName "myTable" -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTable](#) 中的。

Set-DDBBatchItem

以下代码示例演示了如何使用 Set-DDBBatchItem。

适用于 PowerShell V4 的工具

示例 1：创建一个新项目，或将现有项目替换为 DynamoDB 表 Music 和 Songs 中的新项目。

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
```

```
}  
  
Set-DDBBatchItem -RequestItem $requestItem
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [BatchWriteItem](#) 中的。

Set-DDBItem

以下代码示例演示了如何使用 Set-DDBItem。

适用于 PowerShell V4 的工具

示例 1：创建一个新项目，或将现有项目替换为新项目。

```
$item = @{  
    SongTitle = 'Somewhere Down The Road'  
    Artist = 'No One You Know'  
    AlbumTitle = 'Somewhat Famous'  
    Price = 1.94  
    Genre = 'Country'  
    CriticRating = 9.0  
} | ConvertTo-DDBItem  
Set-DDBItem -TableName 'Music' -Item $item
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutItem](#) 中的。

Update-DDBItem

以下代码示例演示了如何使用 Update-DDBItem。

适用于 PowerShell V4 的工具

示例 1：使用分区 SongTitle 键和排序键 Artist 将 DynamoDB 项目上的流派属性设置为“说唱”。

```
$key = @{  
    SongTitle = 'Somewhere Down The Road'  
    Artist = 'No One You Know'  
} | ConvertTo-DDBItem  
  
$updateDdbItem = @{
```

```

    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem

```

输出：

Name	Value
----	-----
Genre	Rap

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateItem](#) 中的。

Update-DDBTable

以下代码示例演示了如何使用 Update-DDBTable。

适用于 PowerShell V4 的工具

示例 1：更新给定表的预置吞吐量。

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateTable](#) 中的。

使用适用于 PowerShell V4 的工具的亚马逊 EC2 示例

以下代码示例向您展示了如何使用带有 Amazon EC2 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-EC2CapacityReservation

以下代码示例演示了如何使用 Add-EC2CapacityReservation。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的属性创建新的容量预留。

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

输出：

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate            : 3/28/2019 9:29:41 AM
EbsOptimized          : True
EndDate               : 1/1/0001 12:00:00 AM
EndDateType           : unlimited
EphemeralStorage      : False
InstanceMatchCriteria : open
InstancePlatform      : Windows
InstanceType          : m4.xlarge
State                 : active
Tags                  : {}
Tenancy                : default
TotalInstanceCount    : 2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateCapacityReservation](#) 中的。

Add-EC2InternetGateway

以下代码示例演示了如何使用 Add-EC2InternetGateway。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的互联网网关连接到指定的 VPC。

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

示例 2：此示例创建一个 VPC 和一个互联网网关，然后将互联网网关连接到 VPC。

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16  
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `AttachInternetGateway`](#) 中的。

Add-EC2NetworkInterface

以下代码示例演示了如何使用 `Add-EC2NetworkInterface`。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的网络接口连接到指定的实例。

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -  
DeviceIndex 1
```

输出：

```
eni-attach-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `AttachNetworkInterface`](#) 中的。

Add-EC2Volume

以下代码示例演示了如何使用 `Add-EC2Volume`。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的卷连接到指定的实例，并使用指定的设备名称公开它。

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

输出：

```
AttachTime      : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device          : /dev/sdh
InstanceId      : i-1a2b3c4d
State          : attaching
VolumeId       : vol-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachVolume](#) 中的。

Add-EC2VpnGateway

以下代码示例演示了如何使用 Add-EC2VpnGateway。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的虚拟专用网关连接到指定的 VPC。

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

输出：

State	VpcId
-----	-----
attaching	vpc-12345678

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachVpnGateway](#) 中的。

Approve-EC2VpcPeeringConnection

以下代码示例演示了如何使用 Approve-EC2VpcPeeringConnection。

适用于 PowerShell V4 的工具

示例 1：此示例批准了请求的 VpcPeeringConnectionId pcx-1dfad234b56ff78be

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

输出：

```
AcceptorVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime       : 1/1/0001 12:00:00 AM
RequesterVpcInfo     : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status               : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                 : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [AcceptVpcPeeringConnection](#) 中的。

Confirm-EC2ProductInstance

以下代码示例演示了如何使用 Confirm-EC2ProductInstance。

适用于 PowerShell V4 的工具

示例 1：此示例确定指定的产品代码是否与指定的实例相关联。

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ConfirmProductInstance](#) 中的。

Copy-EC2Image

以下代码示例演示了如何使用 Copy-EC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例将“欧洲地区（爱尔兰）”区域中的指定 AMI 复制到“美国西部（俄勒冈州）”区域。如果未指定 -Region，则使用当前默认区域作为目标区域。

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2  
-Name "Copy of ami-12345678"
```

输出：

```
ami-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CopyImage](#) 中的。

Copy-EC2Snapshot

以下代码示例演示了如何使用 Copy-EC2Snapshot。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的快照从欧洲地区（爱尔兰）区域复制到美国西部（俄勒冈州）区域。

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-west-2
```

示例 2：如果您设置了默认区域并省略了 Region 参数，则默认目标区域为默认区域。

```
Set-DefaultAWSRegion us-west-2  
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CopySnapshot](#) 中的。

Deny-EC2VpcPeeringConnection

以下代码示例演示了如何使用 Deny-EC2VpcPeeringConnection。

适用于 PowerShell V4 的工具

示例 1：上面的示例拒绝了请求编号为 pcx-01a2b3ce45fe67eb8 的 VpcPeering 请求

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RejectVpcPeeringConnection](#) 中的。

Disable-EC2VgwRoutePropagation

以下代码示例演示了如何使用 Disable-EC2VgwRoutePropagation。

适用于 PowerShell V4 的工具

示例 1：此示例禁止 VGW 自动将路由传播到指定的路由表。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DisableVgwRoutePropagation](#) 中的。

Disable-EC2VpcClassicLink

以下代码示例演示了如何使用 Disable-EC2VpcClassicLink。

适用于 PowerShell V4 的工具

示例 1：此示例禁用 vpc-01e23c4a5d6d EC2 VpcClassicLink b78e9。它返回 True 或 False

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DisableVpcClassicLink](#) 中的。

Disable-EC2VpcClassicLinkDnsSupport

以下代码示例演示了如何使用 Disable-EC2VpcClassicLinkDnsSupport。

适用于 PowerShell V4 的工具

示例 1：此示例禁用了对 vpc-0b12d3456a7e ClassicLink 8910d 的 DNS 支持

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DisableVpcClassicLinkDnsSupport](#) 中的。

Dismount-EC2InternetGateway

以下代码示例演示了如何使用 Dismount-EC2InternetGateway。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的互联网网关与指定的 VPC 分离。

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DetachInternetGateway](#) 中的。

Dismount-EC2NetworkInterface

以下代码示例演示了如何使用 Dismount-EC2NetworkInterface。

适用于 PowerShell V4 的工具

示例 1：此示例移除网络接口和实例之间的指定连接。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DetachNetworkInterface](#) 中的。

Dismount-EC2Volume

以下代码示例演示了如何使用 Dismount-EC2Volume。

适用于 PowerShell V4 的工具

示例 1：此示例分离指定的卷。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

输出：

```
AttachTime       : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device           : /dev/sdh
InstanceId       : i-1a2b3c4d
State            : detaching
```

```
VolumeId : vol-12345678
```

示例 2：您还可以指定实例 ID 和设备名称，以确保分离的卷正确无误。

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachVolume](#) 中的。

Dismount-EC2VpnGateway

以下代码示例演示了如何使用 Dismount-EC2VpnGateway。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的虚拟专用网关与指定的 VPC 分离。

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachVpnGateway](#) 中的。

Edit-EC2CapacityReservation

以下代码示例演示了如何使用 Edit-EC2CapacityReservation。

适用于 PowerShell V4 的工具

示例 1：此示例通过将实例计数更改为 1 来修改 CapacityReservationId cr-0c1f2345db6f7cdba

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyCapacityReservation](#) 中的。

Edit-EC2Host

以下代码示例演示了如何使用 Edit-EC2Host。

适用于 PowerShell V4 的工具

示例 1：此示例将专用主机的 AutoPlacement 设置修改为关闭 h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

输出：

```
Successful          Unsuccessful
-----
{h-01e23f4cd567890f3} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyHosts](#) 中的。

Edit-EC2IdFormat

以下代码示例演示了如何使用 Edit-EC2IdFormat。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的资源类型启用较长的 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

示例 2：此示例为指定的资源类型禁用较长的 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyIdFormat](#) 中的。

Edit-EC2ImageAttribute

以下代码示例演示了如何使用 Edit-EC2ImageAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例更新指定 AMI 的描述。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

示例 2：此示例将 AMI 设为公开（例如，任何人 AWS 账户 都可以使用）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

示例 3：此示例将 AMI 设为私有（例如，只有作为所有者的您才能使用它）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

示例 4：此示例向指定的授予启动权限 AWS 账户。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

示例 5：此示例从指定的中删除启动权限 AWS 账户。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyImageAttribute`](#) 中的。

Edit-EC2InstanceAttribute

以下代码示例演示了如何使用 Edit-EC2InstanceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定实例的实例类型。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

示例 2：此示例通过将“simple”指定为单根虚拟化 (SR-IOV) 网络支持参数的值，为指定实例启用增强联网，方法是将“simple”指定为单根 I/O 虚拟化 (SR-IOV) 网络支持参数的值，...

SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

示例 3：此示例修改指定实例的安全组。该实例必须在 VPC 中。您必须指定每个安全组的 ID，而不是名称。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

示例 4：此示例为指定实例启用 EBS I/O 优化。此功能并非适用于所有实例类型。使用 EBS 优化的实例时，可能需要支付额外的使用费。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

示例 5：此示例启用了对指定实例的 source/destination 检查。对于要执行 NAT 的 NAT 实例，该值必须为“false”。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

示例 6：此示例对指定的实例禁用终止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

示例 7：此示例更改指定的实例，使其在从实例启动关闭时终止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior  
terminate
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyInstanceAttribute`](#) 中的。

Edit-EC2InstanceCreditSpecification

以下代码示例演示了如何使用 `Edit-EC2InstanceCreditSpecification`。

适用于 PowerShell V4 的工具

示例 1：这会为实例 i-01234567890abcdef 启用 T2 无限积分。

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyInstanceCreditSpecification`](#) 中的。

Edit-EC2NetworkInterfaceAttribute

以下代码示例演示了如何使用 Edit-EC2NetworkInterfaceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定的网络接口，以便在终止时删除指定的连接。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

示例 2：此示例修改指定网络接口的描述。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
description"
```

示例 3：此示例修改指定网络接口的安全组。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
sg-1a2b3c4d
```

示例 4：此示例禁用对指定网络接口的 source/destination 检查。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
$false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyNetworkInterfaceAttribute`](#) 中的。

Edit-EC2ReservedInstance

以下代码示例演示了如何使用 Edit-EC2ReservedInstance。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定预留实例的可用区、实例计数和平台。

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @"FE32132D-70D5-4795-B400-AE435EXAMPLE", "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE" `
-TargetConfiguration $config
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyReservedInstances](#) 中的。

Edit-EC2SnapshotAttribute

以下代码示例演示了如何使用 Edit-EC2SnapshotAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例通过设置指定快照的 CreateVolumePermission 属性将其公开。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission -OperationType Add -GroupName all
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifySnapshotAttribute](#) 中的。

Edit-EC2SpotFleetRequest

以下代码示例演示了如何使用 Edit-EC2SpotFleetRequest。

适用于 PowerShell V4 的工具

示例 1：此示例更新指定竞价型实例集请求的目标容量。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ModifySpotFleetRequest](#) 中的。

Edit-EC2SubnetAttribute

以下代码示例演示了如何使用 Edit-EC2SubnetAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的子网启用公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

示例 2：此示例为指定的子网禁用公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ModifySubnetAttribute](#) 中的。

Edit-EC2VolumeAttribute

以下代码示例演示了如何使用 Edit-EC2VolumeAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例修改了指定卷的指定属性。I/O 由于数据可能不一致，该卷的操作将在暂停后自动恢复。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyVolumeAttribute](#) 中的。

Edit-EC2VpcAttribute

以下代码示例演示了如何使用 Edit-EC2VpcAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例启用对指定 VPC 的 DNS 主机名的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

示例 2：此示例禁用对指定 VPC 的 DNS 主机名的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

示例 3：此示例启用对指定 VPC 的 DNS 解析的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

示例 4：此示例禁用对指定 VPC 的 DNS 解析的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyVpcAttribute](#) 中的。

Enable-EC2VgwRoutePropagation

以下代码示例演示了如何使用 Enable-EC2VgwRoutePropagation。

适用于 PowerShell V4 的工具

示例 1：此示例支持指定的 VGW 自动将路由传播到指定的路由表。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [EnableVgwRoutePropagation](#) 中的。

Enable-EC2VolumeIO

以下代码示例演示了如何使用 Enable-EC2VolumeIO。

适用于 PowerShell V4 的工具

示例 1：如果禁用了 I/O 操作，则此示例将启用对指定卷的 I/O 操作。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableVolumeIO](#) 中的。

Enable-EC2VpcClassicLink

以下代码示例演示了如何使用 Enable-EC2VpcClassicLink。

适用于 PowerShell V4 的工具

示例 1：此示例启用 VPC vpc-0123456b789b0d12f ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableVpcClassicLink](#) 中的。

Enable-EC2VpcClassicLinkDnsSupport

以下代码示例演示了如何使用 Enable-EC2VpcClassicLinkDnsSupport。

适用于 PowerShell V4 的工具

示例 1：此示例启用 vpc-0b12d3456a7e8910d 支持 DNS 主机名解析 ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```


- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) EnableVpcClassicLinkDnsSupport](#) 中的。

Get-EC2AccountAttribute

以下代码示例演示了如何使用 Get-EC2AccountAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述了您是在该区域的 EC2-Classic 和 EC2-VPC 中启动实例，还是只能启动到-VPC 中 EC2。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

输出：

```
AttributeValue
-----
EC2
VPC
```

示例 2：此示例描述您的默认 VPC，或者如果您在该区域没有默认 VPC，则为“无”。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

输出：

```
AttributeValue
-----
vpc-12345678
```

示例 3：此示例描述您可以运行的最大按需型实例数。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

输出：

```
AttributeValue
-----
20
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAccountAttributes](#) 中的。

Get-EC2Address

以下代码示例演示了如何使用 Get-EC2Address。

适用于 PowerShell V4 的工具

示例 1：此示例描述了 EC2-Classic 中实例的指定弹性 IP 地址。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

输出：

```
AllocationId      : eipalloc-12345678
AssociationId     : eipassoc-12345678
Domain           : vpc
InstanceId       : i-87654321
NetworkInterfaceId : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress  : 10.0.2.172
PublicIp         : 198.51.100.2
```

示例 2：此示例描述 VPC 中实例的弹性 IP 地址。此语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

示例 3：此示例描述了 EC2-Classic 中实例的指定弹性 IP 地址。

```
Get-EC2Address -PublicIp 203.0.113.17
```

输出：

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId       : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
PrivateIpAddress  :
```

```
PublicIp                : 203.0.113.17
```

示例 4：此示例描述了 EC2-Classic 中实例的弹性 IP 地址。此语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

示例 5：此示例描述所有弹性 IP 地址。

```
Get-EC2Address
```

示例 6：此示例返回筛选条件中提供的实例 ID 的公有和私有 IP

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

输出：

```
PrivateIpAddress PublicIp
-----
10.0.0.99          63.36.5.227
```

示例 7：此示例检索所有 Elast IPs ic 及其分配 ID、关联 ID 和实例 ID

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
AllocationId, PublicIp
```

输出：

InstanceId	AssociationId	AllocationId	PublicIp
-----	-----	-----	-----
		eipalloc-012e3b456789e1fad	
17.212.120.178			
i-0c123dfd3415bac67	eipassoc-0e123456bb7890bdb	eipalloc-01cd23ebf45f7890c	
17.212.124.77			
		eipalloc-012345678eeabcfad	
17.212.225.7			
i-0123d405c67e89a0c	eipassoc-0c123b456783966ba	eipalloc-0123cdd456a8f7892	
37.216.52.173			
i-0f1bf2f34c5678d09	eipassoc-0e12934568a952d96	eipalloc-0e1c23e4d5e6789e4	
37.218.222.278			

```
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
37.215.222.270
```

示例 8：此示例获取与标签键“类别”匹配且值为“Prod”的 EC2 IP 地址列表

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

输出：

```
AllocationId      : eipalloc-0123f456f81a01b58
AssociationId     : eipassoc-0d1b23a456d103810
CustomerOwnedIp   :
CustomerOwnedIpv4Pool :
Domain           : vpc
InstanceId        : i-012e3cb4df567e1aa
NetworkBorderGroup : eu-west-1
NetworkInterfaceId : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress  : 192.168.1.84
PublicIp          : 34.250.81.29
PublicIpv4Pool    : amazon
Tags              : {Category, Name}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeAddresses](#) 中的。

Get-EC2AvailabilityZone

以下代码示例演示了如何使用 Get-EC2AvailabilityZone。

适用于 PowerShell V4 的工具

示例 1：此示例描述当前区域可供您使用的可用区。

```
Get-EC2AvailabilityZone
```

输出：

Messages	RegionName	State	ZoneName

-----	-----	-----	-----
{}	us-west-2	available	us-west-2a
{}	us-west-2	available	us-west-2b
{}	us-west-2	available	us-west-2c

示例 2：此示例描述所有处于受损状态的可用区。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

示例 3：在 PowerShell 版本 2 中，必须使用 New-Object 来创建过滤器。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"
```

```
Get-EC2AvailabilityZone -Filter $filter
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeAvailabilityZones](#) 中的。

Get-EC2BundleTask

以下代码示例演示了如何使用 Get-EC2BundleTask。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的捆绑包任务。

```
Get-EC2BundleTask -BundleId bun-12345678
```

示例 2：此示例描述状态为“完成”或“失败”的捆绑包任务。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )
```

```
Get-EC2BundleTask -Filter $filter
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeBundleTasks](#) 中的。

Get-EC2CapacityReservation

以下代码示例演示了如何使用 Get-EC2CapacityReservation。

适用于 PowerShell V4 的工具

示例 1：此示例描述该区域的一个或多个容量预留

```
Get-EC2CapacityReservation -Region eu-west-1
```

输出：

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate            : 3/28/2019 9:29:41 AM
EbsOptimized          : True
EndDate               : 1/1/0001 12:00:00 AM
EndDateType           : unlimited
EphemeralStorage      : False
InstanceMatchCriteria : open
InstancePlatform      : Windows
InstanceType          : m4.xlarge
State                 : active
Tags                  : {}
Tenancy                : default
TotalInstanceCount    : 2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeCapacityReservations](#) 中的。

Get-EC2ConsoleOutput

以下代码示例演示了如何使用 Get-EC2ConsoleOutput。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定 Linux 实例的控制台输出。控制台输出经过编码。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

输出：

InstanceId	Output
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

示例 2：此示例将经过编码的控制台输出存储在变量中，然后对其进行解码。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetConsoleOutput](#) 中的。

Get-EC2CustomerGateway

以下代码示例演示了如何使用 Get-EC2CustomerGateway。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的客户网关。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

输出：

BgpAsn	: 65534
CustomerGatewayId	: cgw-1a2b3c4d
IpAddress	: 203.0.113.12
State	: available
Tags	: {}
Type	: ipsec.1

示例 2：此示例描述状态为待定或可用的所有客户网关。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )
```

```
Get-EC2CustomerGateway -Filter $filter
```

示例 3：此示例描述所有客户网关。

```
Get-EC2CustomerGateway
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeCustomerGateways](#) 中的。

Get-EC2DhcpOption

以下代码示例演示了如何使用 Get-EC2DhcpOption。

适用于 PowerShell V4 的工具

示例 1：此示例列出 DHCP 选项集。

```
Get-EC2DhcpOption
```

输出：

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

示例 2：此示例获取指定 DHCP 选项集的配置详细信息。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

输出：

Key	Values
---	-----
domain-name	{abc.local}
domain-name-servers	{10.0.0.101, 10.0.0.102}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeDhcpOptions](#) 中的。

Get-EC2FlowLog

以下代码示例演示了如何使用 Get-EC2FlowLog。

适用于 PowerShell V4 的工具

示例 1：此示例描述日志目标类型为“s3”的一个或多个流日志

```
Get-EC2FlowLog -Filter @{"Name="log-destination-type";Values="s3"}
```

输出：

```
CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : fl-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination         : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType     : s3
LogGroupName           :
ResourceId             : eni-01d2dda3456b7e890
TrafficType            : ALL
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeFlowLogs](#) 中的。

Get-EC2Host

以下代码示例演示了如何使用 Get-EC2Host。

适用于 PowerShell V4 的工具

示例 1：此示例返回 EC2 主机详细信息

```
Get-EC2Host
```

输出：

```
AllocationTime      : 3/23/2019 4:55:22 PM
AutoPlacement       : off
```

```

AvailabilityZone : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken      :
HostId           : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances        : {}
ReleaseTime      : 1/1/0001 12:00:00 AM
State            : available
Tags             : {}

```

示例 2：此示例查询主机 h-01e23f4cd AvailableInstanceCapacity 567899f1 的主机

```

Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity

```

输出：

```

AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeHosts](#) 中的。

Get-EC2HostReservationOffering

以下代码示例演示了如何使用 Get-EC2HostReservationOffering。

适用于 PowerShell V4 的工具

示例 1：此示例描述了可为给定过滤器“实例系列”购买的专用主机预留 PaymentOption，其中“”NoUpfront

```

Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
Where-Object PaymentOption -eq NoUpfront

```

输出：

```

CurrencyCode :

```

```

Duration      : 94608000
HourlyPrice   : 1.307
InstanceFamily : m4
OfferingId    : hro-0c1f234567890d9ab
PaymentOption : NoUpfront
UpfrontPrice  : 0.000

```

```

CurrencyCode  :
Duration      : 31536000
HourlyPrice   : 1.830
InstanceFamily : m4
OfferingId    : hro-04ad12aaaf34b5a67
PaymentOption : NoUpfront
UpfrontPrice  : 0.000

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeHostReservationOfferings](#) 中的。

Get-EC2HostReservationPurchasePreview

以下代码示例演示了如何使用 Get-EC2HostReservationPurchasePreview。

适用于 PowerShell V4 的工具

示例 1：此示例预览配置与专属主机 h-01e23f4cd567890f1 的配置匹配的预留购买。

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

输出：

```

CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
-----
{}          1.307          0.000

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetHostReservationPurchasePreview](#) 中的。

Get-EC2IdFormat

以下代码示例演示了如何使用 Get-EC2IdFormat。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定资源类型的 ID 格式。

```
Get-EC2IdFormat -Resource instance
```

输出：

Resource	UseLongIds
-----	-----
instance	False

示例 2：此示例描述了所有支持更长时间的资源类型的 ID 格式 IDs。

```
Get-EC2IdFormat
```

输出：

Resource	UseLongIds
-----	-----
reservation	False
instance	False

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeIdFormat](#) 中的。

Get-EC2IdentityIdFormat

以下代码示例演示了如何使用 Get-EC2IdentityIdFormat。

适用于 PowerShell V4 的工具

示例 1：此示例返回给定角色的资源“image”的 ID 格式

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource image
```

输出：

Deadline	Resource	UseLongIds

```
-----
8/2/2018 11:30:00 PM image      True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribelyIdFormat](#) 中的。

Get-EC2Image

以下代码示例演示了如何使用 Get-EC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 AMI。

```
Get-EC2Image -ImageId ami-12345678
```

输出：

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate       : 2014-10-20T00:56:28.000Z
Description        : My image
Hypervisor         : xen
ImageId            : ami-12345678
ImageLocation      : 123456789012/my-image
ImageOwnerAlias     :
ImageType          : machine
KernelId           :
Name               : my-image
OwnerId            : 123456789012
Platform           :
ProductCodes       : {}
Public             : False
RamdiskId          :
RootDeviceName     : /dev/xvda
RootDeviceType     : ebs
SriovNetSupport    : simple
State              : available
StateReason        :
Tags               : {Name}
VirtualizationType : hvm
```

示例 2：此示例描述 AMIs 了您拥有的。

```
Get-EC2Image -owner self
```

示例 3：此示例描述了运行微软 Windows Server 的公众 AMIs。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

示例 4：此示例描述了“us-west-2”区域的所有公众 AMIs。

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeImages](#) 中的。

Get-EC2ImageAttribute

以下代码示例演示了如何使用 Get-EC2ImageAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定 AMI 的描述。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

输出：

```
BlockDeviceMappings : {}
Description           : My image description
ImageId               : ami-12345678
KernelId              :
LaunchPermissions     : {}
ProductCodes          : {}
RamdiskId             :
SriovNetSupport       :
```

示例 2：此示例描述指定 AMI 的启动权限。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

输出：

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {all}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      :
```

示例 3：此示例测试是否启用增强联网。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

输出：

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      : simple
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeImageAttribute](#) 中的。

Get-EC2ImageByName

以下代码示例演示了如何使用 Get-EC2ImageByName。

适用于 PowerShell V4 的工具

示例 1：此示例描述当前支持的完整筛选条件名称集。

```
Get-EC2ImageByName
```

输出：

```
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
```

示例 2：此示例描述指定的 AMI。使用此命令查找 AMI 很有用，因为每个月都会 AWS 发布 AMIs 包含最新更新的新 Windows。您可以将 “ImageId” 指定 New-EC2Instance 为使用指定筛选器的当前 AMI 来启动实例。


```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

输出：

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc...}
CreationDate       : yyyy.mm.ddThh:mm:ss.000Z
Description        : Microsoft Windows Server 2016 with Desktop Experience Locale
                    English AMI provided by Amazon
Hypervisor         : xen
ImageId            : ami-xxxxxxxxx
ImageLocation       : amazon/Windows_Server-2016-English-Full-Base-yyyy.mm.dd
ImageOwnerAlias     : amazon
ImageType          : machine
KernelId           :
Name               : Windows_Server-2016-English-Full-Base-yyyy.mm.dd
OwnerId            : 801119661308
Platform           : Windows
ProductCodes       : {}
Public             : True
RamdiskId          :
RootDeviceName     : /dev/sda1
RootDeviceType     : ebs
SriovNetSupport     : simple
State              : available
StateReason        :
Tags               : {}
VirtualizationType : hvm
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [Get-EC2ImageByName](#) 中的。

Get-EC2ImportImageTask

以下代码示例演示了如何使用 Get-EC2ImportImageTask。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的映像导入任务。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

输出：

```
Architecture      : x86_64
Description        : Windows Image 2
Hypervisor         :
ImageId            : ami-1a2b3c4d
ImportTaskId       : import-ami-hgfedcba
LicenseType        : AWS
Platform           : Windows
Progress           :
SnapshotDetails    : {/dev/sda1}
Status             : completed
StatusMessage      :
```

示例 2：此示例描述所有映像导入任务。

```
Get-EC2ImportImageTask
```

输出：

```
Architecture      :
Description        : Windows Image 1
Hypervisor         :
ImageId            :
ImportTaskId       : import-ami-abcdefgh
LicenseType        : AWS
Platform           : Windows
Progress           :
SnapshotDetails    : {}
Status             : deleted
StatusMessage      : User initiated task cancelation

Architecture      : x86_64
Description        : Windows Image 2
Hypervisor         :
ImageId            : ami-1a2b3c4d
ImportTaskId       : import-ami-hgfedcba
LicenseType        : AWS
Platform           : Windows
Progress           :
SnapshotDetails    : {/dev/sda1}
Status             : completed
```

StatusMessage :

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeImportImageTasks](#) 中的。

Get-EC2ImportSnapshotTask

以下代码示例演示了如何使用 Get-EC2ImportSnapshotTask。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的快照导入任务。

Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh

输出：

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

示例 2：此示例描述所有快照导入任务。

Get-EC2ImportSnapshotTask

输出：

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2	import-snap-hgfedcba	Amazon.EC2.Model.SnapshotTaskDetail

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeImportSnapshotTasks](#) 中的。

Get-EC2Instance

以下代码示例演示了如何使用 Get-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的实例。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

输出：

```
AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken          : TleEy1448154045270
EbsOptimized         : False
Hypervisor           : xen
IamInstanceProfile   : Amazon.EC2.Model.IamInstanceProfile
ImageId              : ami-12345678
InstanceId           : i-12345678
InstanceLifecycle    :
InstanceType         : t2.micro
KernelId             :
KeyName              : my-key-pair
LaunchTime           : 12/4/2015 4:44:40 PM
Monitoring           : Amazon.EC2.Model.Monitoring
NetworkInterfaces    : {ip-10-0-2-172.us-west-2.compute.internal}
Placement            : Amazon.EC2.Model.Placement
Platform             : Windows
PrivateDnsName       : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress     : 10.0.2.172
ProductCodes         : {}
PublicDnsName        :
PublicIpAddress      :
RamdiskId            :
RootDeviceName       : /dev/sda1
RootDeviceType       : ebs
SecurityGroups       : {default}
SourceDestCheck      : True
SpotInstanceRequestId :
SriovNetSupport      :
State                : Amazon.EC2.Model.InstanceState
```

```
StateReason      :
StateTransitionReason :
SubnetId         : subnet-12345678
Tags             : {Name}
VirtualizationType : hvm
VpcId           : vpc-12345678
```

示例 2：此示例描述您在当前区域中的所有实例，按预留分组。要查看实例详细信息，请在每个预留对象中展开实例集合。

```
Get-EC2Instance
```

输出：

```
GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 226008221399
ReservationId   : r-c5df370c

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 854251627541
ReservationId   : r-63e65bab
...
```

示例 3：此示例说明如何使用筛选器查询 VPC 的特定子网中的 EC2 实例。

```
(Get-EC2Instance -Filter @{{Name="vpc-id";Values="vpc-1a2bc34d"}},{Name="subnet-id";Values="subnet-1a2b3c4d"}}).Instances
```

输出：

```
InstanceId      InstanceType Platform PrivateIpAddress PublicIpAddress
SecurityGroups SubnetId      VpcId
-----
-----
i-01af...82cf180e19 t2.medium   Windows  10.0.0.98
subnet-1a2b3c4d vpc-1a2b3c4d
```

```
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53    ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

示例 4：此示例说明如何使用具有多个值的筛选器来查询同时处于运行状态和已停止状态的 EC2 实例

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	PublicIpAddress
SecurityGroups	SubnetId	VpcId		
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	...
subnet-1a2b3c4d	vpc-1a2b3c4d			
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	...
subnet-1a2b3c4d	vpc-1a2b3c4d			

示例 5：此示例说明如何使用具有多个值的过滤器来查询同时运行和已停止的 EC2 实例，以及使用 Select-Object cmdlet 来选择要输出的特定值。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

Select-Object -InputObject ($InstanceParams | Get-EC2Instance) -Property InstanceId, InstanceType, Platform, PrivateIpAddress, PublicIpAddress
```

```
$SelectParams = @{
    Property = @(
        "InstanceId", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_].Tags.Key.IndexOf("Name")}.Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	Name	State
-----	-----	-----	-----	----	----
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	ec2-name-01	running
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	ec2-name-02	stopped
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	ec2-name-03	running
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	ec2-name-04	stopped
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	ec2-name-05	running
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	ec2-name-06	stopped

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeInstances](#) 中的。

Get-EC2InstanceAttribute

以下代码示例演示了如何使用 Get-EC2InstanceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定实例的实例类型。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

输出：

```
InstanceType           : t2.micro
```

示例 2：此示例描述是否为指定实例启用增强联网。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

输出：

```
SriovNetSupport          : simple
```

示例 3：此示例描述指定实例的安全组。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

输出：

```
GroupId
-----
sg-12345678
sg-45678901
```

示例 4：此示例描述是否为指定的实例启用 EBS 优化。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

输出：

```
EbsOptimized            : False
```

示例 5：此示例描述了指定实例disableApiTermination的“”属性。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

输出：

```
DisableApiTermination    : False
```

示例 6：此示例描述了指定实例instanceInitiatedShutdown的“行为”属性。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
instanceInitiatedShutdownBehavior
```


输出：

```
InstanceInitiatedShutdownBehavior : stop
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeInstanceAttribute](#) 中的。

Get-EC2InstanceMetadata

以下代码示例演示了如何使用 Get-EC2InstanceMetadata。

适用于 PowerShell V4 的工具

示例 1：列出可查询的可用实例元数据类别。

```
Get-EC2InstanceMetadata -ListCategory
```

输出：

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
```

```
IdentityPkcs7
```

示例 2：返回用于启动实例的亚马逊机器映像 (AMI) 的 ID。

```
Get-EC2InstanceMetadata -Category AmiId
```

输出：

```
ami-b2e756ca
```

示例 3：此示例查询实例的 JSON 格式的身份文档。

```
Get-EC2InstanceMetadata -Category IdentityDocument
{
  "availabilityZone" : "us-west-2a",
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : null,
  "version" : "2017-09-30",
  "instanceId" : "i-01ed50f7e2607f09e",
  "billingProducts" : [ "bp-6ba54002" ],
  "instanceType" : "t2.small",
  "pendingTime" : "2018-03-07T16:26:04Z",
  "imageId" : "ami-b2e756ca",
  "privateIp" : "10.0.0.171",
  "accountId" : "111122223333",
  "architecture" : "x86_64",
  "kernelId" : null,
  "ramdiskId" : null,
  "region" : "us-west-2"
}
```

示例 4：此示例使用路径查询来获取实例的网络接口 mac。

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

输出：

```
02:80:7f:ef:4c:e0/
```

示例 5：如果存在与该实例关联的 IAM 角色，则返回有关上次更新实例配置文件的时间的信息，包括实例的 LastUpdated 日期 InstanceProfileArn、和 InstanceProfileId。

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

输出：

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [Get-EC2InstanceMetadata](#) 中的。

Get-EC2InstanceStatus

以下代码示例演示了如何使用 Get-EC2InstanceStatus。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定实例的状态。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

输出：

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

输出：

Code	Name

```

----      ----
16        running

```

```
$status.Status
```

输出：

```

Details      Status
-----
{reachability}  ok

```

```
$status.SystemStatus
```

输出：

```

Details      Status
-----
{reachability}  ok

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeInstanceStatus](#) 中的。

Get-EC2InternetGateway

以下代码示例演示了如何使用 Get-EC2InternetGateway。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的互联网网关。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

输出：

```

Attachments      InternetGatewayId      Tags
-----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}

```

示例 2：此示例描述所有互联网网关。

```
Get-EC2InternetGateway
```

输出：

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeInternetGateways](#) 中的。

Get-EC2KeyPair

以下代码示例演示了如何使用 Get-EC2KeyPair。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的密钥对。

```
Get-EC2KeyPair -KeyName my-key-pair
```

输出：

KeyFingerprint	KeyName
-----	-----
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f	my-key-pair

示例 2：此示例描述所有密钥对。

```
Get-EC2KeyPair
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeKeyPairs](#) 中的。

Get-EC2NetworkAcl

以下代码示例演示了如何使用 Get-EC2NetworkAcl。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络 ACL。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

输出：

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

示例 2：此示例描述指定网络 ACL 的规则。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

输出：

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode  :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode  :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

示例 3：此示例描述了您的所有网络 ACLs。

```
Get-EC2NetworkAcl
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeNetworkAcls](#) 中的。

Get-EC2NetworkInterface

以下代码示例演示了如何使用 Get-EC2NetworkInterface。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络接口。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

输出：

```
Association      :
Attachment       : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone  : us-west-2c
Description      :
Groups           : {my-security-group}
MacAddress        : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId          : 123456789012
PrivateDnsName    : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress  : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId      :
RequesterManaged : False
SourceDestCheck   : True
Status           : in-use
SubnetId         : subnet-1a2b3c4d
TagSet           : {}
VpcId            : vpc-12345678
```

示例 2：此示例描述所有网络接口。

```
Get-EC2NetworkInterface
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeNetworkInterfaces](#) 中的。

Get-EC2NetworkInterfaceAttribute

以下代码示例演示了如何使用 Get-EC2NetworkInterfaceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

输出：

```
Attachment          : Amazon.EC2.Model.NetworkInterfaceAttachment
```

示例 2：此示例描述指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

输出：

```
Description         : My description
```

示例 3：此示例描述指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

输出：

```
Groups              : {my-security-group}
```

示例 4：此示例描述指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

输出：


```
SourceDestCheck      : True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeNetworkInterfaceAttribute](#) 中的。

Get-EC2PasswordData

以下代码示例演示了如何使用 Get-EC2PasswordData。

适用于 PowerShell V4 的工具

示例 1：此示例解密亚马逊为指定 Windows 实例的管理员账户 EC2 分配的密码。当指定了 pem 文件时，将自动假定 -Decrypt 开关的设置。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

输出：

```
mYZ(PA9?C)Q
```

示例 2：(PowerShell 仅限 Windows) 检查实例以确定用于启动实例的密钥对的名称，然后尝试在 Visual Studio T AWS Toolkit for Visual Studio 的配置存储中查找相应的密钥对数据。如果找到了密钥对数据，则对密码进行解密。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

输出：

```
mYZ(PA9?C)Q
```

示例 3：返回实例的已加密密码数据。

```
Get-EC2PasswordData -InstanceId i-12345678
```

输出：

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetPasswordData](#) 中的。

Get-EC2PlacementGroup

以下代码示例演示了如何使用 Get-EC2PlacementGroup。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的置放群组。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

输出：

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribePlacementGroups](#) 中的。

Get-EC2PrefixList

以下代码示例演示了如何使用 Get-EC2PrefixList。

适用于 PowerShell V4 的工具

示例 1：此示例以前缀列表格式获取该区域的可用 AWS 服务 内容

```
Get-EC2PrefixList
```

输出：

Cidrs	PrefixListId	PrefixListName
-----	-----	-----
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23}	pl-6fa54006	com.amazonaws.eu-west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}	pl-6da54004	com.amazonaws.eu-west-1.s3

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribePrefixLists](#) 中的。

Get-EC2Region

以下代码示例演示了如何使用 Get-EC2Region。

适用于 PowerShell V4 的工具

示例 1：此示例描述可供您使用的区域。

```
Get-EC2Region
```

输出：

Endpoint	RegionName
-----	-----
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeRegions](#) 中的。

Get-EC2RouteTable

以下代码示例演示了如何使用 Get-EC2RouteTable。

适用于 PowerShell V4 的工具

示例 1：此示例描述所有路由表。

```
Get-EC2RouteTable
```

输出：

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId :
```

示例 2：此示例返回指定路由表的详细信息。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

示例 3：此示例描述指定 VPC 的路由表。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

输出：

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeRouteTables](#) 中的。

Get-EC2ScheduledInstance

以下代码示例演示了如何使用 Get-EC2ScheduledInstance。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的计划实例。

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-123456789012
```

输出：

```
AvailabilityZone      : us-west-2b
CreateDate            : 1/25/2016 1:43:38 PM
HourlyPrice           : 0.095
InstanceCount        : 1
InstanceType          : c4.large
NetworkPlatform      : EC2-VPC
NextSlotStartTime     : 1/31/2016 1:00:00 AM
Platform             : Linux/UNIX
PreviousSlotEndTime   :
Recurrence            : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId   : sci-1234-1234-1234-123456789012
SlotDurationInHours   : 32
TermEndDate           : 1/31/2017 1:00:00 AM
TermStartDate        : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

示例 2：此示例描述所有计划实例。

```
Get-EC2ScheduledInstance
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeScheduledInstances](#) 中的。

Get-EC2ScheduledInstanceAvailability

以下代码示例演示了如何使用 Get-EC2ScheduledInstanceAvailability。

适用于 PowerShell V4 的工具

示例 1：此示例描述从指定日期开始每周星期日发生的计划。

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

输出：

```
AvailabilityZone           : us-west-2b
AvailableInstanceCount     : 20
FirstSlotStartTime         : 1/31/2016 8:00:00 AM
HourlyPrice                : 0.095
InstanceType               : c4.large
MaxTermDurationInDays     : 366
MinTermDurationInDays     : 366
NetworkPlatform            : EC2-VPC
Platform                   : Linux/UNIX
PurchaseToken              : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                 : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours        : 23
TotalScheduledInstanceHours : 1219
...
```

示例 2：要缩小结果范围，您可以为操作系统、网络 and 实例类型等条件添加筛选条件。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeScheduledInstanceAvailability](#) 中的。

Get-EC2SecurityGroup

以下代码示例演示了如何使用 Get-EC2SecurityGroup。

适用于 PowerShell V4 的工具

示例 1：此示例描述 VPC 的指定安全组。使用属于 VPC 的安全组时，必须使用安全组 ID (-GroupId 参数)，而不是名称 (-GroupName 参数) 来引用该组。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

输出：

```
Description      : default VPC security group
GroupId          : sg-12345678
GroupName        : default
IpPermissions    : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId         : 123456789012
Tags            : {}
VpcId           : vpc-12345678
```

示例 2：此示例描述了 EC2-Classic 的指定安全组。在 EC2-Classic 中使用安全组时，您可以使用组名称 (`-GroupName` 参数) 或组 ID (`-GroupId` 参数) 来引用安全组。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

输出：

```
Description      : my security group
GroupId          : sg-45678901
GroupName        : my-security-group
IpPermissions    : {Amazon.EC2.Model.IpPermission, Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId         : 123456789012
Tags            : {}
VpcId           :
```

示例 3：此示例检索 vpc-0fc1ff23456b789eb 的所有安全组

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeSecurityGroups](#) 中的。

Get-EC2Snapshot

以下代码示例演示了如何使用 `Get-EC2Snapshot`。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的快照。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

输出：

```
DataEncryptionKeyId :
Description          : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
                      vol-12345678
Encrypted            : False
KmsKeyId             :
OwnerAlias           :
OwnerId              : 123456789012
Progress             : 100%
SnapshotId           : snap-12345678
StartTime            : 10/23/2014 6:01:28 AM
State                : completed
StateMessage         :
Tags                 : {}
VolumeId             : vol-12345678
VolumeSize           : 8
```

示例 2：此示例描述具有“Name”标签的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

示例 3：此示例描述了带有“名称”标签且值为“TestValue”的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
                      $_.Tags.Value -eq "TestValue" }
```

示例 4：此示例描述所有快照。

```
Get-EC2Snapshot -Owner self
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeSnapshots](#) 中的。

Get-EC2SnapshotAttribute

以下代码示例演示了如何使用 Get-EC2SnapshotAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定快照的指定属性。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

输出：

CreateVolumePermissions	ProductCodes	SnapshotId
-----	-----	-----
{}	{}	snap-12345678

示例 2：此示例描述指定快照的指定属性。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute CreateVolumePermission).CreateVolumePermissions
```

输出：

Group	UserId
-----	-----
all	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSnapshotAttribute](#) 中的。

Get-EC2SpotDatafeedSubscription

以下代码示例演示了如何使用 Get-EC2SpotDatafeedSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例描述竞价型实例数据源。

```
Get-EC2SpotDatafeedSubscription
```

输出：

Bucket	: amzn-s3-demo-bucket

```
Fault      :  
OwnerId    : 123456789012  
Prefix     : spotdata  
State      : Active
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotDatafeedSubscription](#) 中的。

Get-EC2SpotFleetInstance

以下代码示例演示了如何使用 Get-EC2SpotFleetInstance。

适用于 PowerShell V4 的工具

示例 1：此示例描述与指定竞价型实例集请求关联的实例。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-  
aa30-494c-8788-1cee4EXAMPLE
```

输出：

InstanceId	InstanceType	SpotInstanceRequestId
-----	-----	-----
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotFleetInstances](#) 中的。

Get-EC2SpotFleetRequest

以下代码示例演示了如何使用 Get-EC2SpotFleetRequest。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的竞价型实例集请求。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE  
| format-list
```

输出：

```
ConfigData          : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime          : 12/26/2015 8:23:33 AM
SpotFleetRequestId  : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

示例 2：此示例描述所有竞价型实例集请求。

```
Get-EC2SpotFleetRequest
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotFleetRequests](#) 中的。

Get-EC2SpotFleetRequestHistory

以下代码示例演示了如何使用 Get-EC2SpotFleetRequestHistory。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的竞价型实例集请求的历史记录。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

输出：

```
HistoryRecords      : {Amazon.EC2.Model.HistoryRecord,
  Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime   : 12/26/2015 8:29:11 AM
NextToken           :
SpotFleetRequestId  : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime           : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

输出：

EventInformation	EventType	Timestamp
-----	-----	-----
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:05 AM

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeSpotFleetRequestHistory](#) 中的。

Get-EC2SpotInstanceRequest

以下代码示例演示了如何使用 Get-EC2SpotInstanceRequest。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的竞价型实例请求。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

输出：

```
ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 4/8/2015 2:51:33 PM
Fault                       :
InstanceId                  : i-12345678
LaunchedAvailabilityZone    : us-west-2b
LaunchGroup                 :
LaunchSpecification         : Amazon.EC2.Model.LaunchSpecification
ProductDescription          : Linux/UNIX
SpotInstanceRequestId       : sir-12345678
SpotPrice                   : 0.020000
State                       : active
Status                      : Amazon.EC2.Model.SpotInstanceStatus
Tags                        : {Name}
Type                        : one-time
```

示例 2：此示例描述所有竞价型实例请求。

Get-EC2SpotInstanceRequest

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotInstanceRequests](#) 中的。

Get-EC2SpotPriceHistory

以下代码示例演示了如何使用 Get-EC2SpotPriceHistory。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定实例类型和可用区的竞价价格历史记录中的最后 10 个条目。请注意，为-AvailabilityZone 参数指定的值必须对提供给 cmdlet 的-Region 参数（示例中未显示）的区域值有效，或者在 shell 中设置为默认值。此示例命令假设环境中已设置了默认区域“us-west-2”。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

输出：

```
AvailabilityZone : us-west-2a
InstanceType     : c3.large
Price            : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp        : 12/25/2015 7:39:49 AM

AvailabilityZone : us-west-2a
InstanceType     : c3.large
Price            : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp        : 12/25/2015 7:38:29 AM

AvailabilityZone : us-west-2a
InstanceType     : c3.large
Price            : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp        : 12/25/2015 6:57:13 AM
...
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotPriceHistory](#) 中的。

Get-EC2Subnet

以下代码示例演示了如何使用 Get-EC2Subnet。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的子网。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

输出：

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz          : False
MapPublicIpOnLaunch   : False
State                 : available
SubnetId              : subnet-1a2b3c4d
Tags                  : {}
VpcId                 : vpc-12345678
```

示例 2：此示例描述所有子网。

```
Get-EC2Subnet
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeSubnets](#) 中的。

Get-EC2Tag

以下代码示例演示了如何使用 Get-EC2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例获取资源类型“image”的标签

```
Get-EC2Tag -Filter @{"Name"="resource-type";Values="image"}
```

输出：

Key	ResourceId	ResourceType	Value
---	-----	-----	----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

示例 2：此示例获取所有资源的所有标签，并按资源类型对它们进行分组

```
Get-EC2Tag | Group-Object resourcetype
```

输出：

Count	Name	Group
-----	-----	-----
9	subnet	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
53	instance	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
3	route-table	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
5	security-group	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
30	volume	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
1	internet-gateway	{Amazon.EC2.Model.TagDescription}
3	network-interface	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
4	elastic-ip	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
1	dhcp-options	{Amazon.EC2.Model.TagDescription}
2	image	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
3	vpc	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

示例 3：此示例显示给定区域中所有带有“auto-delete”标签且值为“no”的资源

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name":"tag:auto-delete";Values="no"}
```

输出：

Key	ResourceId	ResourceType	Value
---	-----	-----	----
auto-delete	i-0f1bce234d5dd678b	instance	no
auto-delete	vol-01d234aa5678901a2	volume	no
auto-delete	vol-01234bfb5def6f7b8	volume	no
auto-delete	vol-01ccb23f4c5e67890	volume	no

示例 4：此示例获取所有带有“auto-delete”标签且值为“无”的资源，并在下一个管道中进行进一步筛选以仅解析“实例”资源类型，最终为每个实例资源创建 ThisInstance “” 标签，其值为实例 ID 本身

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name":"tag:auto-delete";Values="no"} |
Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -ResourceId
$_.ResourceId -Tag @{"Key":"ThisInstance";Value=$_.ResourceId}}
```

示例 5：此示例获取所有实例资源的标签以及“Name”键并以表格式显示它们

```
Get-EC2Tag -Filter @{"Name":"resource-
type";Values="instance"},@{"Name":"key";Values="Name"} | Select-Object ResourceId,
@{"Name":"Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

输出：

ResourceId	Name-Tag
-----	-----
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

Get-EC2Volume

以下代码示例演示了如何使用 Get-EC2Volume。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 EBS 卷。

```
Get-EC2Volume -VolumeId vol-12345678
```

输出：

```
Attachments      : {}
AvailabilityZone  : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

示例 2：此示例描述状态为“可用”的 EBS 卷。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

输出：

```
Attachments      : {}
AvailabilityZone  : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

示例 3：此示例描述所有 EBS 卷。

```
Get-EC2Volume
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeVolumes](#) 中的。

Get-EC2VolumeAttribute

以下代码示例演示了如何使用 Get-EC2VolumeAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定卷的指定属性。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

输出：

AutoEnableIO	ProductCodes	VolumeId
-----	-----	-----
False	{}	vol-12345678

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeVolumeAttribute](#) 中的。

Get-EC2VolumeStatus

以下代码示例演示了如何使用 Get-EC2VolumeStatus。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定卷的状态。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

输出：

```
Actions           : {}
AvailabilityZone   : us-west-2a
Events            : {}
```

```
VolumeId      : vol-12345678
VolumeStatus  : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

输出：

```
Details              Status
-----
{io-enabled, io-performance}  ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

输出：

```
Name              Status
----
io-enabled         passed
io-performance     not-applicable
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVolumeStatus](#) 中的。

Get-EC2Vpc

以下代码示例演示了如何使用 Get-EC2Vpc。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 VPC。

```
Get-EC2Vpc -VpcId vpc-12345678
```

输出：

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
```

```
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

示例 2：此示例描述默认 VPC（每个区域只能有一个）。如果您的账户在此区域支持 EC2-Classic，则没有默认 VPC。

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

输出：

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

示例 3：此示例描述了 VPCs 与指定过滤器匹配的（即，具有与值“10.0.0.0/16”匹配且处于“可用”状态的 CIDR）。

```
Get-EC2Vpc -Filter @{Name="cidr";
  Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

示例 4：此示例描述了您的所有内容 VPCs。

```
Get-EC2Vpc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeVpcs](#) 中的。

Get-EC2VpcAttribute

以下代码示例演示了如何使用 Get-EC2VpcAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述了“enableDnsSupport”属性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

输出：

```
EnableDnsSupport
-----
True
```

示例 2：此示例描述了“enableDnsHostnames”属性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

输出：

```
EnableDnsHostnames
-----
True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVpcAttribute](#) 中的。

Get-EC2VpcClassicLink

以下代码示例演示了如何使用 Get-EC2VpcClassicLink。

适用于 PowerShell V4 的工具

示例 1：上面的示例返回 VPCs 了该区域的所有及其 ClassicLinkEnabled 状态

```
Get-EC2VpcClassicLink -Region eu-west-1
```

输出：

ClassicLinkEnabled	Tags	VpcId
-----	----	-----
False	{Name}	vpc-0fc1ff23f45b678eb
False	{}	vpc-01e23c4a5d6db78e9
False	{Name}	vpc-0123456b078b9d01f
False	{}	vpc-12cf3b4f

```
False {Name} vpc-0b12d3456a7e8901d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVpcClassicLink](#) 中的。

Get-EC2VpcClassicLinkDnsSupport

以下代码示例演示了如何使用 Get-EC2VpcClassicLinkDnsSupport。

适用于 PowerShell V4 的工具

示例 1：此示例描述了 eu-west-1 区域 VPCs 的 ClassicLink DNS 支持状态

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

输出：

```
ClassicLinkDnsSupported VpcId
-----
False                  vpc-0b12d3456a7e8910d
False                  vpc-12cf3b4f
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVpcClassicLinkDnsSupport](#) 中的。

Get-EC2VpcEndpoint

以下代码示例演示了如何使用 Get-EC2VpcEndpoint。

适用于 PowerShell V4 的工具

示例 1：此示例描述区域 eu-west-1 的一个或多个 VPC 端点。然后，它将输出通过管道传递给下一个命令，该命令选择 VpcEndpointId 属性并以字符串数组的形式返回数组 VPC ID

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

输出：

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
```

```
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

示例 2：此示例描述了 eu-west-1 区域的所有 vpc 终端节点，并 VpcEndpointId选择 VpcId、ServiceName、PrivateDnsEnabled 和属性以表格格式呈现该终端节点

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

输出：

VpcEndpointId	VpcId	ServiceName
PrivateDnsEnabled		
-----	-----	-----

vpce-02a2ab2f2f2cc2f2d	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssm
True		
vpce-01d1b111a1114561b	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2
True		
vpce-0011e23d45167e838	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2messages
True		
vpce-0c123db4567890123	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssmmessages
True		

示例 3：此示例将 VPC 端点 vpce-01a2ab3f4f5cc6f7d 的策略文档导出到 json 文件中

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeVpcEndpoints](#)中的。

Get-EC2VpcEndpointService

以下代码示例演示了如何使用 Get-EC2VpcEndpointService。

适用于 PowerShell V4 的工具

示例 1：此示例描述了使用给定筛选条件的 EC2 VPC 终端节点服务，在本例中为 com.amazonaws.eu-west-1.ecs。此外，它还会扩展 ServiceDetails 属性并显示细节

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{"Name"="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

输出：

```
AcceptanceRequired      : False
AvailabilityZones       : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames    : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                   : amazon
PrivateDnsName          : ecs.eu-west-1.amazonaws.com
ServiceName             : com.amazonaws.eu-west-1.ecs
ServiceType             : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

示例 2：此示例检索所有 EC2 VPC 终端节点服务并返回 ServiceNames 匹配的“ssm”

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty Servicenames | Where-Object { -match "ssm"}
```

输出：

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVpcEndpointServices](#) 中的。

Get-EC2VpnConnection

以下代码示例演示了如何使用 Get-EC2VpnConnection。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 VPN 连接。

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

输出：


```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
Type                         : ipsec.1
VgwTelemetry                 : {Amazon.EC2.Model.VgwTelemetry,
    Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId             : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

示例 2：此示例描述状态为待定或可用的任何 VPN 连接。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

示例 3：此示例描述所有 VPN 连接。

```
Get-EC2VpnConnection
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeVpnConnections](#) 中的。

Get-EC2VpnGateway

以下代码示例演示了如何使用 Get-EC2VpnGateway。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的虚拟专用网关。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

输出：

```
AvailabilityZone :
```

```
State           : available
Tags            : {}
Type            : ipsec.1
VpcAttachments  : {vpc-12345678}
VpnGatewayId    : vgw-1a2b3c4d
```

示例 2：此示例描述状态为待定或可用的所有虚拟专用网关。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

示例 3：此示例描述所有虚拟专用网关。

```
Get-EC2VpnGateway
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVpnGateways](#) 中的。

Grant-EC2SecurityGroupEgress

以下代码示例演示了如何使用 Grant-EC2SecurityGroupEgress。

适用于 PowerShell V4 的工具

示例 1：此示例为 EC2-VPC 的指定安全组定义了出口规则。此规则授予对 TCP 端口 80 上指定 IP 地址范围的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
```

```
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 3：此示例授予对 TCP 端口 80 上指定源安全组的访问权限。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [AuthorizeSecurityGroupEgress](#) 中的。

Grant-EC2SecurityGroupIngress

以下代码示例演示了如何使用 Grant-EC2SecurityGroupIngress。

适用于 PowerShell V4 的工具

示例 1：此示例定义了 EC2-VPC 安全组的入口规则。这些规则为 SSH (端口 22) 和 RDC (端口 3389) 授予对特定 IP 地址的访问权限。请注意，您必须使用安全组 ID 而不是安全组名称来识别 EC2-VPC 的安全组。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$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 sg-12345678 -IpPermission @( $ip1, $ip2 )
```

示例 3：此示例定义了 EC2-Classic 安全组的入口规则。这些规则为 SSH (端口 22) 和 RDC (端口 3389) 授予对特定 IP 地址的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
  $ip2 )
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$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 -GroupName "my-security-group" -IpPermission @( $ip1,
  $ip2 )
```

示例 5：此示例授予 TCP 端口 8081 从指定的源安全组 (sg-1a2b3c4d) 到指定的安全组 (sg-12345678) 的访问权限。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
```

```
Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

示例 6：此示例将 CIDR 5.5.5.5/32 添加到 TCP 端口 22 流量的安全组 sg-1234abcd 的入口规则中，并附有描述。

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [AuthorizeSecurityGroupIngress](#) 中的。

Import-EC2Image

以下代码示例演示了如何使用 Import-EC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例使用等性令牌将单磁盘虚拟机映像从指定的 Amazon S3 存储桶导入到 EC2 Amazon。该示例要求存在默认名称为“vmimport”的虚拟机导入服务角色，其策略允许亚马逊 EC2 访问指定的存储桶，如虚拟机导入先决条件主题中所述。要使用自定义角色，请使用 **-RoleName** 参数指定角色名称。

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
```

```
"LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

输出：

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ImportImage](#) 中的。

Import-EC2KeyPair

以下代码示例演示了如何使用 Import-EC2KeyPair。

适用于 PowerShell V4 的工具

示例 1：此示例将公钥导入到 EC2。第一行将公有密钥文件 (*.pub) 的内容存储在变量 **\$publickey** 中。接下来，该示例将公钥文件的 UTF8 格式转换为 Base64 编码的字符串，并将转换后的字符串存储在变量中。**\$pkbase64** 在最后一行中，转换后的公钥被导入到 EC2。cmdlet 将密钥指纹和名称作为结果返回。

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

输出：

KeyFingerprint	KeyName

```
-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
-----
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ImportKeyPair](#) 中的。

Import-EC2Snapshot

以下代码示例演示了如何使用 Import-EC2Snapshot。

适用于 PowerShell V4 的工具

示例 1：此示例将格式为“VMDK”的虚拟机磁盘映像导入到 Amazon EBS 快照中。该示例需要一个默认名称为“vmimport”的虚拟机导入服务角色，其策略允许亚马逊 EC2 访问指定的存储桶，如 <http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html>。要使用自定义角色，请使用 **-RoleName** 参数指定角色名称。

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

输出：

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import	import-snap-abcdefgh	
Amazon.EC2.Model.SnapshotTaskDetail		

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ImportSnapshot](#) 中的。

Move-EC2AddressToVpc

以下代码示例演示了如何使用 Move-EC2AddressToVpc。

适用于 PowerShell V4 的工具

示例 1：此示例将公有 IP 地址为 12.345.67.89 的 EC2 实例移动到美国东部（弗吉尼亚北部）区域的 EC2-VPC 平台。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

示例 2：此示例将 Get-EC2Instance 命令的结果传送到 Move-EC2AddressToVpc cmdlet。该 Get-EC2Instance 命令获取由实例 ID 指定的实例，然后返回该实例的公有 IP 地址属性。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) MoveAddressToVpc](#) 中的。

New-EC2Address

以下代码示例演示了如何使用 New-EC2Address。

适用于 PowerShell V4 的工具

示例 1：此示例分配一个弹性 IP 地址，供 VPC 中的实例使用。

```
New-EC2Address -Domain Vpc
```

输出：

AllocationId	Domain	PublicIp
-----	-----	-----
eipalloc-12345678	vpc	198.51.100.2

示例 2：此示例分配弹性 IP 地址以用于 EC2-Classic 中的实例。

```
New-EC2Address
```

输出：

AllocationId	Domain	PublicIp
-----	-----	-----
	standard	203.0.113.17

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AllocateAddress](#) 中的。

New-EC2CustomerGateway

以下代码示例演示了如何使用 New-EC2CustomerGateway。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的客户网关。

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

输出：

```
BgpAsn           : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress        : 203.0.113.12
State            : available
Tags             : {}
Type             : ipsec.1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateCustomerGateway](#) 中的。

New-EC2DhcpOption

以下代码示例演示了如何使用 New-EC2DhcpOption。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的 DHCP 选项集。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{"domain-name-servers";Values=@("10.0.0.101","10.0.0.102")})
```

```
New-EC2DhcpOption -DhcpConfiguration $options
```

输出：

DhcpConfigurations	DhcpOptionsId	Tags
-----	-----	----
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个 DHCP 选项。

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @"10.0.0.101","10.0.0.102"@

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

输出：

DhcpConfigurations	DhcpOptionsId	Tags
-----	-----	----
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateDhcpOptions](#) 中的。

New-EC2FlowLog

以下代码示例演示了如何使用 New-EC2FlowLog。

适用于 PowerShell V4 的工具

示例 1：此示例使用“管理员”角色的权限为所有“拒绝”EC2 流量创建子网 subnet-1d234567 到 cloud-watch-log 名为“subnet1-log”的流日志

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

输出：

```
ClientToken                                FlowLogIds                                Unsuccessful
-----
m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw= {f1-012fc34eed5678c9d} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateFlowLogs](#) 中的。

New-EC2Host

以下代码示例演示了如何使用 New-EC2Host。

适用于 PowerShell V4 的工具

示例 1：此示例向您的账户分配给定实例类型和可用区的专属主机

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

输出：

```
h-01e23f4cd567890f3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AllocateHosts](#) 中的。

New-EC2HostReservation

以下代码示例演示了如何使用 New-EC2HostReservation。

适用于 PowerShell V4 的工具

示例 1：此示例购买可提供 0c1f23456789d0ab 的预留，其配置与专属主机 h-01e23f4cd567890f1 的配置匹配

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
h-01e23f4cd567890f1
```

输出：

```
ClientToken      :  
CurrencyCode     :  
Purchase         : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PurchaseHostReservation](#) 中的。

New-EC2Image

以下代码示例演示了如何使用 New-EC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的实例创建具有指定名称和描述的 AMI。Amazon EC2 尝试在创建映像之前彻底关闭实例，并在完成后重新启动实例。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI"
```

示例 2：此示例从指定的实例创建具有指定名称和描述的 AMI。Amazon 在不关闭和重启实例的情况下 EC2 创建映像；因此，无法保证所创建映像的文件系统的完整性。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI" -NoReboot $true
```

示例 3：此示例创建一个包含三个卷的 AMI。第一个卷基于 Amazon EBS 快照。第二个卷是一个空的 100 GiB Amazon EBS 卷。第三卷是一个实例存储卷。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ebsBlock1 = @{{SnapshotId="snap-1a2b3c4d"}}  
$ebsBlock2 = @{{VolumeSize=100}}  
  
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description  
  "My web server AMI" -BlockDeviceMapping @( @{{DeviceName="/dev/sdf";Ebs=  
$ebsBlock1}}, @{{DeviceName="/dev/sdg";Ebs=$ebsBlock2}}, @{{DeviceName="/dev/  
sdc";VirtualName="ephemeral0"}})
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateImage](#) 中的。

New-EC2Instance

以下代码示例演示了如何使用 New-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例在 EC2-Classic 或默认 VPC 中启动指定 AMI 的单个实例。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

示例 2：此示例在 VPC 中启动指定 AMI 的单个实例。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

示例 3：要添加 EBS 卷或实例存储卷，请定义块设备映射并将其添加到命令中。此示例添加实例存储卷。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

示例 4：要指定当前 Windows 中的一个 AMIs，请使用获取其 AMI ID Get-EC2ImageByName。此示例从适用于 Windows Server 2016 的当前基础 AMI 启动一个实例。

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

示例 5：将实例启动到指定的专属主机环境中。

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID
h-1a2b3c4d5e6f1a2b3
```

示例 6：此请求启动两个实例，并将键为 `webserver` 和值为 `production` 的标签应用于实例。该请求还向创建的卷（此示例中为每个实例的根卷）应用键为 `cost-center` 且值为 `cc123` 的标签。

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RunInstances](#) 中的。

New-EC2InstanceExportTask

以下代码示例演示了如何使用 `New-EC2InstanceExportTask`。

适用于 PowerShell V4 的工具

示例 1：此示例将已停止的实例 **i-0800b00a00EXAMPLE** 作为虚拟硬盘（VHD）导出到 S3 存储桶 **testbucket-export-instances-2019**。目标环境是 **Microsoft**，添加区域参数是因为实例位于该区域，而用户的默认 **us-east-1** AWS 区域不是 **us-east-1**。要获取导出任务的状态，请复制此命令的结果中的 **ExportTaskId** 值，然后运行 **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket" -
TargetEnvironment Microsoft -Region us-east-1
```

输出：

```
Description          :
ExportTaskId         : export-i-077c73108aEXAMPLE
```

```
ExportToS3Task      : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State               : active
StatusMessage       :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateInstanceExportTask](#) 中的。

New-EC2InternetGateway

以下代码示例演示了如何使用 New-EC2InternetGateway。

适用于 PowerShell V4 的工具

示例 1：此示例创建互联网网关。

```
New-EC2InternetGateway
```

输出：

Attachments	InternetGatewayId	Tags
-----	-----	----
{}	igw-1a2b3c4d	{}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateInternetGateway](#) 中的。

New-EC2KeyPair

以下代码示例演示了如何使用 New-EC2KeyPair。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个密钥对，并在具有指定名称的文件中捕获经过 PEM 编码的 RSA 私钥。使用时 PowerShell，必须将编码设置为 ascii 才能生成有效的密钥。有关更多信息，请参阅《AWS 命令行界面用户指南》中的创建、显示和删除亚马逊 EC2 密钥对 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html>)。

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateKeyPair](#) 中的。

New-EC2NetworkAcl

以下代码示例演示了如何使用 New-EC2NetworkAcl。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 VPC 创建网络 ACL。

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

输出：

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateNetworkAcl](#) 中的。

New-EC2NetworkAclEntry

以下代码示例演示了如何使用 New-EC2NetworkAclEntry。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的网络 ACL 创建一个条目。该规则可让来自 UDP 端口 53 (DNS) 上任何地址 (0.0.0.0/0) 的入站流量进入任何关联的子网。

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -RuleAction
allow
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateNetworkAclEntry](#) 中的。

New-EC2NetworkInterface

以下代码示例演示了如何使用 New-EC2NetworkInterface。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的网络接口。

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

输出：

```
Association      :
Attachment       :
AvailabilityZone  : us-west-2c
Description      : my network interface
Groups           : {my-security-group}
MacAddress       : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId          : 123456789012
PrivateDnsName    : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress  : 10.0.0.17
PrivateIpAddresses : {}
RequesterId      :
RequesterManaged : False
SourceDestCheck   : True
Status           : pending
SubnetId          : subnet-1a2b3c4d
TagSet           : {}
VpcId            : vpc-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateNetworkInterface](#) 中的。

New-EC2PlacementGroup

以下代码示例演示了如何使用 New-EC2PlacementGroup。

适用于 PowerShell V4 的工具

示例 1：此示例命令使用指定的名称创建置放群组。

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreatePlacementGroup](#) 中的。

New-EC2Route

以下代码示例演示了如何使用 New-EC2Route。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的路由表创建指定的路由。该路由匹配所有流量，并将其路由到指定的互联网网关。

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId igw-1a2b3c4d
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateRoute](#) 中的。

New-EC2RouteTable

以下代码示例演示了如何使用 New-EC2RouteTable。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 VPC 创建路由表。

```
New-EC2RouteTable -VpcId vpc-12345678
```

输出：

```
Associations      : {}  
PropagatingVgws   : {}  
Routes            : {}  
RouteTableId      : rtb-1a2b3c4d
```

```
Tags          : {}  
VpcId         : vpc-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateRouteTable](#) 中的。

New-EC2ScheduledInstance

以下代码示例演示了如何使用 New-EC2ScheduledInstance。

适用于 PowerShell V4 的工具

示例 1：此示例启动指定的计划实例。

```
New-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012 -  
InstanceCount 1 `  
-IamInstanceProfile_Name my-iam-role `  
-LaunchSpecification_ImageId ami-12345678 `  
-LaunchSpecification_InstanceType c4.large `  
-LaunchSpecification_SubnetId subnet-12345678 `  
-LaunchSpecification_SecurityGroupId sg-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RunScheduledInstances](#) 中的。

New-EC2ScheduledInstancePurchase

以下代码示例演示了如何使用 New-EC2ScheduledInstancePurchase。

适用于 PowerShell V4 的工具

示例 1：此示例购买计划实例。

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest  
$request.InstanceCount = 1  
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiEsImMiOi..."  
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

输出：

```
AvailabilityZone      : us-west-2b
```

```

CreateDate           : 1/25/2016 1:43:38 PM
HourlyPrice          : 0.095
InstanceCount        : 1
InstanceType         : c4.large
NetworkPlatform      : EC2-VPC
NextSlotStartTime    : 1/31/2016 1:00:00 AM
Platform             : Linux/UNIX
PreviousSlotEndTime   :
Recurrence            : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId   : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours   : 32
TermEndDate          : 1/31/2017 1:00:00 AM
TermStartDate         : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PurchaseScheduledInstances](#) 中的。

New-EC2SecurityGroup

以下代码示例演示了如何使用 New-EC2SecurityGroup。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 VPC 创建一个安全组。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

输出：

```
sg-12345678
```

示例 2：此示例为 EC2-Classic 创建了一个安全组。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

输出：

```
sg-45678901
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateSecurityGroup](#) 中的。

New-EC2Snapshot

以下代码示例演示了如何使用 New-EC2Snapshot。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定卷的快照。

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

输出：

```
DataEncryptionKeyId :  
Description          : This is a test  
Encrypted            : False  
KmsKeyId             :  
OwnerAlias           :  
OwnerId              : 123456789012  
Progress             :  
SnapshotId           : snap-12345678  
StartTime            : 12/22/2015 1:28:42 AM  
State                : pending  
StateMessage         :  
Tags                 : {}  
VolumeId             : vol-12345678  
VolumeSize           : 20
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateSnapshot](#) 中的。

New-EC2SpotDatafeedSubscription

以下代码示例演示了如何使用 New-EC2SpotDatafeedSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例创建竞价型实例数据源。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

输出：

```
Bucket   : amzn-s3-demo-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateSpotDatafeedSubscription](#) 中的。

New-EC2Subnet

以下代码示例演示了如何使用 New-EC2Subnet。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的 CIDR 创建子网。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

输出：

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock              : 10.0.0.0/24
DefaultForAz           : False
MapPublicIpOnLaunch    : False
State                  : pending
SubnetId               : subnet-1a2b3c4d
Tag                    : {}
VpcId                  : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateSubnet](#) 中的。

New-EC2Tag

以下代码示例演示了如何使用 New-EC2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的资源添加单个标签。标签键为“myTag”，标签值为“myTagValue”。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

示例 2：此示例更新指定的标签，或将其添加到指定的资源。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
    @{ Key="test"; Value="anotherTagValue" } )
```

示例 3：在 PowerShell 版本 2 中，必须使用 New-Object 为标签参数创建标签。

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTags](#) 中的。

New-EC2Volume

以下代码示例演示了如何使用 New-EC2Volume。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的卷。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

输出：

```
Attachments      : {}  
AvailabilityZone  : us-west-2a
```

```

CreateTime      : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size            : 50
SnapshotId       :
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2

```

示例 2：此示例请求创建一个卷，并应用键为 stack 且值为 production 的标签。

```

$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateVolume](#) 中的。

New-EC2Vpc

以下代码示例演示了如何使用 New-EC2Vpc。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的 CIDR 创建 VPC。Amazon VPC 还会为 VPC 创建以下各项：默认 DHCP 选项集、主路由表和默认网络 ACL。

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

输出：

```

CidrBlock        : 10.0.0.0/16
DhcpOptionsId     : dopt-1a2b3c4d
InstanceTenancy   : default
IsDefault         : False
State            : pending

```



```
Tags          : {}  
VpcId         : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateVpc](#) 中的。

New-EC2VpcEndpoint

以下代码示例演示了如何使用 New-EC2VpcEndpoint。

适用于 PowerShell V4 的工具

示例 1：此示例在 VPC vpc-0fc1ff23f45b678eb 中为服务 com.amazonaws.eu-west-1.s3 创建新的 VPC 端点

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId  
vpc-0fc1ff23f45b678eb
```

输出：

```
ClientToken VpcEndpoint  
-----  
Amazon.EC2.Model.VpcEndpoint
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateVpcEndpoint](#) 中的。

New-EC2VpnConnection

以下代码示例演示了如何使用 New-EC2VpnConnection。

适用于 PowerShell V4 的工具

示例 1：此示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接。输出包括网络管理员需要的配置信息（采用 XML 格式）。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId  
vgw-1a2b3c4d
```

输出：

```
CustomerGatewayConfiguration : [XML document]
```

```
CustomerGatewayId      : cgw-1a2b3c4d
Options                 :
Routes                  : {}
State                   : pending
Tags                    : {}
Type                    :
VgwTelemetry            : {}
VpnConnectionId        : vpn-12345678
VpnGatewayId           : vgw-1a2b3c4d
```

示例 2：此示例创建 VPN 连接，并在具有指定名称的文件中捕获配置。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

示例 3：此示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接以及静态路由。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateVpnConnection](#) 中的。

New-EC2VpnConnectionRoute

以下代码示例演示了如何使用 New-EC2VpnConnectionRoute。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 VPN 连接创建指定的静态路由。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
11.12.0.0/16
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateVpnConnectionRoute](#) 中的。

New-EC2VpnGateway

以下代码示例演示了如何使用 New-EC2VpnGateway。

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的虚拟专用网关。

```
New-EC2VpnGateway -Type ipsec.1
```

输出：

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateVpnGateway](#) 中的。

Register-EC2Address

以下代码示例演示了如何使用 Register-EC2Address。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的弹性 IP 地址与 VPC 中的指定实例相关联。

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

输出：

```
eipassoc-12345678
```

示例 2：此示例将指定的弹性 IP 地址与 EC2-Classic 中的指定实例相关联。

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AssociateAddress](#) 中的。

Register-EC2DhcpOption

以下代码示例演示了如何使用 Register-EC2DhcpOption。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的 DHCP 选项集与指定的 VPC 相关联。

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

示例 2：此示例将默认 DHCP 选项集与指定的 VPC 相关联。

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AssociateDhcpOptions](#) 中的。

Register-EC2Image

以下代码示例演示了如何使用 Register-EC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例在 Amazon S3 中使用指定的清单文件注册 AMI。

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterImage](#) 中的。

Register-EC2PrivateIpAddress

以下代码示例演示了如何使用 Register-EC2PrivateIpAddress。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的辅助私有 IP 地址分配给指定的网络接口。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress  
10.0.0.82
```

示例 2：此示例创建两个辅助私有 IP 地址并将它们分配给指定的网络接口。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -  
SecondaryPrivateIpAddressCount 2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AssignPrivateIpAddresses](#) 中的。

Register-EC2RouteTable

以下代码示例演示了如何使用 Register-EC2RouteTable。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的路由表与指定的子网相关联。

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

输出：

```
rtbassoc-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AssociateRouteTable](#) 中的。

Remove-EC2Address

以下代码示例演示了如何使用 Remove-EC2Address。

适用于 PowerShell V4 的工具

示例 1：此示例为 VPC 中的实例释放指定的弹性 IP 地址。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

示例 2：此示例为 EC2-Classic 中的实例释放指定的弹性 IP 地址。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReleaseAddress](#) 中的。

Remove-EC2CapacityReservation

以下代码示例演示了如何使用 Remove-EC2CapacityReservation。

适用于 PowerShell V4 的工具

示例 1：此示例取消容量预留 cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation (CancelCapacityReservation)"
on target "cr-0c1f2345db6f7cdba".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CancelCapacityReservation](#) 中的。

Remove-EC2CustomerGateway

以下代码示例演示了如何使用 Remove-EC2CustomerGateway。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的客户网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

输出：

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on Target  
"cgw-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteCustomerGateway](#) 中的。

Remove-EC2DhcpOption

以下代码示例演示了如何使用 Remove-EC2DhcpOption。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的 DHCP 选项集。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target  
"dopt-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteDhcpOptions](#) 中的。

Remove-EC2FlowLog

以下代码示例演示了如何使用 Remove-EC2FlowLog。

适用于 PowerShell V4 的工具

示例 1：此示例删除了给定的 FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"fl-01a2b3456a789c01".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFlowLogs](#) 中的。

Remove-EC2Host

以下代码示例演示了如何使用 Remove-EC2Host。

适用于 PowerShell V4 的工具

示例 1：此示例释放给定的主机 ID 0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
"h-0badafd1dcb2f3456".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

Successful                Unsuccessful
-----
{h-0badafd1dcb2f3456} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReleaseHosts](#) 中的。

Remove-EC2Instance

以下代码示例演示了如何使用 Remove-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例终止指定的实例（该实例可能正在运行或处于“已停止”状态）。在继续操作之前，cmdlet 会提示您进行确认；使用 -Force 开关可禁止提示。

```
Remove-EC2Instance -InstanceId i-12345678
```

输出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) TerminateInstances](#) 中的。

Remove-EC2InternetGateway

以下代码示例演示了如何使用 Remove-EC2InternetGateway。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的互联网网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on Target
"igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteInternetGateway](#) 中的。

Remove-EC2KeyPair

以下代码示例演示了如何使用 Remove-EC2KeyPair。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的密钥对。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteKeyPair](#) 中的。

Remove-EC2NetworkAcl

以下代码示例演示了如何使用 Remove-EC2NetworkAcl。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的网络 ACL。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

输出：

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target  
"acl-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteNetworkAcl](#) 中的。

Remove-EC2NetworkAclEntry

以下代码示例演示了如何使用 Remove-EC2NetworkAclEntry。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的网络 ACL 移除指定的规则。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on Target  
"acl-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteNetworkAclEntry](#) 中的。

Remove-EC2NetworkInterface

以下代码示例演示了如何使用 Remove-EC2NetworkInterface。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的网络接口。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on Target
"eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteNetworkInterface](#) 中的。

Remove-EC2PlacementGroup

以下代码示例演示了如何使用 Remove-EC2PlacementGroup。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的置放群组。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
"my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeletePlacementGroup](#) 中的。

Remove-EC2Route

以下代码示例演示了如何使用 Remove-EC2Route。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的路由表中删除指定的路由。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRoute](#) 中的。

Remove-EC2RouteTable

以下代码示例演示了如何使用 Remove-EC2RouteTable。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的路由表。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
"rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRouteTable](#) 中的。

Remove-EC2SecurityGroup

以下代码示例演示了如何使用 Remove-EC2SecurityGroup。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 EC2-VPC 的指定安全组。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
"sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

示例 2：此示例删除了 EC2-Classic 的指定安全组。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteSecurityGroup](#) 中的。

Remove-EC2Snapshot

以下代码示例演示了如何使用 Remove-EC2Snapshot。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的快照。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

输出：

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target  
"snap-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteSnapshot](#) 中的。

Remove-EC2SpotDatafeedSubscription

以下代码示例演示了如何使用 Remove-EC2SpotDatafeedSubscription。

适用于 PowerShell V4 的工具

示例 1：此示例删除竞价型实例数据源。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2SpotDatafeedSubscription
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2SpotDatafeedSubscription  
(DeleteSpotDatafeedSubscription)" on Target "".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteSpotDatafeedSubscription](#) 中的。

Remove-EC2Subnet

以下代码示例演示了如何使用 Remove-EC2Subnet。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的子网。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target "subnet-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteSubnet](#) 中的。

Remove-EC2Tag

以下代码示例演示了如何使用 Remove-EC2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的资源删除指定的标签，无论标签值如何。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

示例 2：此示例从指定的资源删除指定的标签，但前提是标签值匹配。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

示例 3：此示例从指定的资源删除指定的标签，无论标签值如何。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

示例 4：此示例从指定的资源删除指定的标签，但前提是标签值匹配。

```
$tag = New-Object Amazon.EC2.Model.Tag
```



```
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTags](#) 中的。

Remove-EC2Volume

以下代码示例演示了如何使用 Remove-EC2Volume。

适用于 PowerShell V4 的工具

示例 1：此示例分离指定的卷。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Volume -VolumeId vol-12345678
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target "vol-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteVolume](#) 中的。

Remove-EC2Vpc

以下代码示例演示了如何使用 Remove-EC2Vpc。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的 VPC。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteVpc](#) 中的。

Remove-EC2VpnConnection

以下代码示例演示了如何使用 Remove-EC2VpnConnection。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的 VPN 连接。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target "vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteVpnConnection](#) 中的。

Remove-EC2VpnConnectionRoute

以下代码示例演示了如何使用 Remove-EC2VpnConnectionRoute。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的 VPN 连接中移除指定的静态路由。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
Target "vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteVpnConnectionRoute](#) 中的。

Remove-EC2VpnGateway

以下代码示例演示了如何使用 Remove-EC2VpnGateway。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的虚拟专用网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
"vgw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteVpnGateway](#) 中的。

Request-EC2SpotFleet

以下代码示例演示了如何使用 Request-EC2SpotFleet。

适用于 PowerShell V4 的工具

示例 1：此示例在价格最低的可用区中为指定的实例类型创建竞价型实例集请求。如果您的账户仅支持 EC2-VPC，则竞价型队列会在具有默认子网的价格最低的可用区启动实例。如果您的账户支持 EC2-Classic，则竞价型队列会在价格最低的可用区中启动 EC2-Classic 中的实例。请注意，您支付的价格不会超过请求的指定竞价价格。

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RequestSpotFleet](#) 中的。

Request-EC2SpotInstance

以下代码示例演示了如何使用 Request-EC2SpotInstance。

适用于 PowerShell V4 的工具

示例 1：此示例在指定的子网中请求一次性竞价型实例。请注意，必须为包含指定子网的 VPC 创建安全组，并且必须使用网络接口通过 ID 指定该安全组。指定网络接口时，您必须包括使用网络接口的子网 ID。

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
```

```
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

输出：

```
ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 12/26/2015 7:44:10 AM
Fault                       :
InstanceId                  :
LaunchedAvailabilityZone    :
LaunchGroup                 :
LaunchSpecification         : Amazon.EC2.Model.LaunchSpecification
ProductDescription          : Linux/UNIX
SpotInstanceRequestId       : sir-12345678
SpotPrice                   : 0.050000
State                       : open
Status                      : Amazon.EC2.Model.SpotInstanceStatus
Tags                        : {}
Type                        : one-time
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RequestSpotInstances](#) 中的。

Reset-EC2ImageAttribute

以下代码示例演示了如何使用 Reset-EC2ImageAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例将“launchPermission”属性重置为其默认值。默认情况下，AMIs 是私有的。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ResetImageAttribute](#) 中的。

Reset-EC2InstanceAttribute

以下代码示例演示了如何使用 Reset-EC2InstanceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例重置指定实例sriovNetSupport的“”属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

示例 2：此示例重置指定实例的“ebsOptimized”属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

示例 3：此示例重置指定实例sourceDestCheck的“”属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

示例 4：此示例重置指定实例disableApiTermination的“”属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

示例 5：此示例重置指定实例的 instanceInitiatedShutdown “行为” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResetInstanceAttribute](#) 中的。

Reset-EC2NetworkInterfaceAttribute

以下代码示例演示了如何使用 Reset-EC2NetworkInterfaceAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例重置对指定网络接口的 source/destination 检查。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ResetNetworkInterfaceAttribute](#) 中的。

Reset-EC2SnapshotAttribute

以下代码示例演示了如何使用 Reset-EC2SnapshotAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例重置指定快照的指定属性。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ResetSnapshotAttribute](#) 中的。

Restart-EC2Instance

以下代码示例演示了如何使用 Restart-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例重启指定的实例。

```
Restart-EC2Instance -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RebootInstances](#) 中的。

Revoke-EC2SecurityGroupEgress

以下代码示例演示了如何使用 Revoke-EC2SecurityGroupEgress。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定安全组的 EC2-VPC 规则。这将撤销对 TCP 端口 80 上指定 IP 地址范围的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
```

```
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 3：此示例撤销对 TCP 端口 80 上指定源安全组的访问权限。

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug }
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RevokeSecurityGroupEgress](#) 中的。

Revoke-EC2SecurityGroupIngress

以下代码示例演示了如何使用 Revoke-EC2SecurityGroupIngress。

适用于 PowerShell V4 的工具

示例 1：此示例撤消了 EC2-VPC 的指定安全组的指定地址范围内对 TCP 端口 22 的访问权限。请注意，您必须使用安全组 ID 而不是安全组名称来识别 EC2-VPC 的安全组。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
```



```
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

示例 3：此示例撤消了 EC2-Classic 的指定安全组的指定地址范围内对 TCP 端口 22 的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RevokeSecurityGroupIngress](#) 中的。

Send-EC2InstanceStatus

以下代码示例演示了如何使用 Send-EC2InstanceStatus。

适用于 PowerShell V4 的工具

示例 1：此示例报告指定实例的状态反馈。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
unresponsive
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ReportInstanceStatus](#) 中的。

Set-EC2NetworkAclAssociation

以下代码示例演示了如何使用 Set-EC2NetworkAclAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的网络 ACL 与指定网络 ACL 关联的子网相关联。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId  
aclassoc-1a2b3c4d
```

输出：

```
aclassoc-87654321
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ReplaceNetworkAclAssociation](#) 中的。

Set-EC2NetworkAclEntry

以下代码示例演示了如何使用 Set-EC2NetworkAclEntry。

适用于 PowerShell V4 的工具

示例 1：此示例替换指定网络 ACL 的指定条目。新规则支持从指定地址到任何关联子网的入站流量。

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100  
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -  
RuleAction allow
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ReplaceNetworkAclEntry](#) 中的。

Set-EC2Route

以下代码示例演示了如何使用 Set-EC2Route。

适用于 PowerShell V4 的工具

示例 1：此示例替换指定路由表的指定路由。新路由将指定的流量发送到指定的虚拟专用网关。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId  
vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReplaceRoute](#) 中的。

Set-EC2RouteTableAssociation

以下代码示例演示了如何使用 Set-EC2RouteTableAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的路由表与指定路由表关联的子网相关联。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
rtbassoc-12345678
```

输出：

```
rtbassoc-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReplaceRouteTableAssociation](#) 中的。

Start-EC2Instance

以下代码示例演示了如何使用 Start-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例启动指定的实例。

```
Start-EC2Instance -InstanceId i-12345678
```

输出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

示例 2：此示例启动指定的实例。

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

示例 3：此示例启动当前已停止的一组实例。返回的实例对象通过 Get-EC2Instance 管道传送到。Start-EC2Instance 此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
| Start-EC2Instance
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 为 Filter 参数创建过滤器。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartInstances](#) 中的。

Start-EC2InstanceMonitoring

以下代码示例演示了如何使用 Start-EC2InstanceMonitoring。

适用于 PowerShell V4 的工具

示例 1：此示例启用对指定实例的详细监控。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

输出：

InstanceId	Monitoring
i-12345678	Amazon.EC2.Model.Monitoring

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [MonitorInstances](#) 中的。

Stop-EC2ImportTask

以下代码示例演示了如何使用 Stop-EC2ImportTask。

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的导入任务（快照或映像导入）。如果需要，可以使用 **-CancelReason** 参数提供原因。

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CancelImportTask](#) 中的。

Stop-EC2Instance

以下代码示例演示了如何使用 Stop-EC2Instance。

适用于 PowerShell V4 的工具

示例 1：此示例停止指定的实例。

```
Stop-EC2Instance -InstanceId i-12345678
```

输出：

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopInstances](#) 中的。

Stop-EC2InstanceMonitoring

以下代码示例演示了如何使用 Stop-EC2InstanceMonitoring。

适用于 PowerShell V4 的工具

示例 1：此示例禁用对指定实例的详细监控。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

输出：

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UnmonitorInstances](#) 中的。

Stop-EC2SpotFleetRequest

以下代码示例演示了如何使用 Stop-EC2SpotFleetRequest。

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的竞价型实例集请求并终止关联的竞价型实例。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

示例 2：此示例取消指定的竞价型实例集请求，但不终止关联的竞价型实例。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CancelSpotFleetRequests](#) 中的。

Stop-EC2SpotInstanceRequest

以下代码示例演示了如何使用 Stop-EC2SpotInstanceRequest。

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的竞价型实例请求。

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

输出：

SpotInstanceRequestId	State

```
-----      -----  
sir-12345678      cancelled
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `CancelSpotInstanceRequests`](#) 中的。

Unregister-EC2Address

以下代码示例演示了如何使用 `Unregister-EC2Address`。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的弹性 IP 地址与 VPC 中的指定实例取消关联。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

示例 2：此示例在 EC2-Classic 中取消指定弹性 IP 地址与指定实例的关联。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `DisassociateAddress`](#) 中的。

Unregister-EC2Image

以下代码示例演示了如何使用 `Unregister-EC2Image`。

适用于 PowerShell V4 的工具

示例 1：此示例取消注册指定的 AMI。

```
Unregister-EC2Image -ImageId ami-12345678
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `DeregisterImage`](#) 中的。

Unregister-EC2PrivateIpAddress

以下代码示例演示了如何使用 `Unregister-EC2PrivateIpAddress`。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的网络接口取消分配指定的私有 IP 地址。

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress 10.0.0.82
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UnassignPrivateIpAddresses](#) 中的。

Unregister-EC2RouteTable

以下代码示例演示了如何使用 Unregister-EC2RouteTable。

适用于 PowerShell V4 的工具

示例 1：此示例移除路由表和子网之间的指定关联。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DisassociateRouteTable](#) 中的。

Update-EC2SecurityGroupRuleIngressDescription

以下代码示例演示了如何使用 Update-EC2SecurityGroupRuleIngressDescription。

适用于 PowerShell V4 的工具

示例 1：更新现有入口（入站）安全组规则的描述。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupId" = $existingInboundRule.SecurityGroupId
    "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```


示例 2：移除现有入口（入站）安全组规则的描述（通过在请求中省略该参数）。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
-SecurityGroupRuleDescription $ruleWithoutDescription
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateSecurityGroupRuleDescriptionsIngress](#) 中的。

使用适用 PowerShell 于 V4 的工具的 Amazon ECR 示例

以下代码示例向您展示如何使用带有 Amazon ECR 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-ECRLoginCommand

以下代码示例演示了如何使用 Get-ECRLoginCommand。

适用于 PowerShell V4 的工具

示例 1：返回一个 PObject 包含登录信息的登录信息，该信息可用于对您的 IAM 委托人有权访问的任何 Amazon ECR 注册表进行身份验证。旨在获取授权令牌的调用所需的凭证和区域端点

是从 Shell 默认值 (由 **Set-AWSCredential/Set-DefaultAWSRegion** 或 **Initialize-AWSDefaultConfiguration** cmdlet 设置) 中获取的。您可以使用带有 Invoke-Expression 的 Command 属性登录到指定的注册表, 或者在其它需要登录的工具中使用返回的凭证。

```
Get-ECRLoginCommand
```

输出:

```
Username      : AWS
Password      : eyJwYXlsb2Fk...kRBVEffS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
eyJwYXlsb2Fk...kRBVEffS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

示例 2: 检索 PObject 包含您用作 docker 登录命令输入的登录信息。只要 IAM 主体有权访问 Amazon ECR 注册表, 您就可以指定任何 Amazon ECR 注册表 URI 来进行身份验证。

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- 有关 API 的详细信息, 请参阅 AWS Tools for PowerShell Cmdlet 参考 (V4) 中的 [Get-ECRLogin 命令](#)。

使用适用于 PowerShell V4 的工具的 Amazon ECS 示例

以下代码示例向您展示了如何使用带有 Amazon ECS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录, 必须在上下文中运行。您可以通过操作了解如何调用单个服务函数, 还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接, 您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-ECSClusterDetail

以下代码示例演示了如何使用 Get-ECSClusterDetail。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 描述一个或多个 ECS 集群。

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

输出：

```
LoggedAt      : 12/27/2019 9:27:41 PM
Clusters      : {LAB-ECS-CL}
Failures      : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength  : 396
HttpStatusCode : OK
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeClusters](#) 中的。

Get-ECSClusterList

以下代码示例演示了如何使用 Get-ECSClusterList。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 返回现有 ECS 集群的列表。

```
Get-ECSClusterList
```

输出：

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListClusters](#) 中的。

Get-ECSClusterService

以下代码示例演示了如何使用 Get-ECSClusterService。

适用于 PowerShell V4 的工具

示例 1：此示例列出在默认集群中运行的所有服务。

```
Get-ECSClusterService
```

示例 2：此示例列出在指定集群中运行的所有服务。

```
Get-ECSClusterService -Cluster myCluster
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListServices](#) 中的。

Get-ECSService

以下代码示例演示了如何使用 Get-ECSService。

适用于 PowerShell V4 的工具

示例 1：此示例演示如何从默认集群中检索特定服务的详细信息。

```
Get-ECSService -Service my-http-service
```

示例 2：此示例演示如何检索在指定集群中运行的特定服务的详细信息。

```
Get-ECSService -Cluster myCluster -Service my-http-service
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeServices](#) 中的。

New-ECSCluster

以下代码示例演示了如何使用 New-ECSCluster。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 创建一个新的 Amazon ECS 集群。

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{"Name="containerInsights";
Value="enabled"}
```

输出：

```
ActiveServicesCount      : 0
Attachments              : {}
AttachmentsStatus        :
CapacityProviders        : {}
ClusterArn               : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName              : LAB-ECS-CL
DefaultCapacityProviderStrategy : {}
PendingTasksCount        : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount        : 0
Settings                 : {containerInsights}
Statistics               : {}
Status                   : ACTIVE
Tags                     : {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateCluster](#) 中的。

New-ECSService

以下代码示例演示了如何使用 New-ECSService。

适用于 PowerShell V4 的工具

示例 1：此示例命令在您的默认集群中创建了一个名为 ecs-simple-service 的服务。该服务使用“ecs-demo”任务定义，并且它维护该任务的 10 个实例化。

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

示例 2：此示例命令在默认集群中的负载均衡器后面创建了一个名为 ecs-simple-service 的服务。该服务使用“ecs-demo”任务定义，并且它维护该任务的 10 个实例化。

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
```

```
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateService](#) 中的。

Remove-ECSCluster

以下代码示例演示了如何使用 Remove-ECSCluster。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除指定的 ECS 集群。必须先从该集群中取消注册所有容器实例，然后才能删除此集群。

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteCluster](#) 中的。

Remove-ECSService

以下代码示例演示了如何使用 Remove-ECSService。

适用于 PowerShell V4 的工具

示例 1：删除默认集群中名为 my-http-service 的服务。在删除服务之前，必须将所需的计数和运行计数设置为 0。在命令继续执行之前，系统会提示您进行确认。要绕过确认提示，请添加 -Force 开关。

```
Remove-ECSService -Service my-http-service
```

示例 2：删除指定集群中名为 my-http-service 的服务。

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteService](#) 中的。

Update-ECSClusterSetting

以下代码示例演示了如何使用 Update-ECSClusterSetting。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 修改要用于 ECS 集群的设置。

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";  
Value="disabled"}
```

输出：

```
ActiveServicesCount      : 0
Attachments              : {}
AttachmentsStatus        :
CapacityProviders        : {}
ClusterArn               : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-  
ECS-CL
ClusterName              : LAB-ECS-CL
DefaultCapacityProviderStrategy : {}
PendingTasksCount        : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount        : 0
Settings                 : {containerInsights}
Statistics               : {}
Status                   : ACTIVE
Tags                    : {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateClusterSettings](#) 中的。

Update-ECSService

以下代码示例演示了如何使用 Update-ECSService。

适用于 PowerShell V4 的工具

示例 1：此示例命令更新 my-http-service 服务以使用 amazon-ecs-sample 任务定义。

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

示例 2：此示例命令将 my-http-service 服务的所需计数更新为 10。

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateService](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon EFS 示例

以下代码示例向您展示了如何使用带有 Amazon EFS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Edit-EFSMountTargetSecurityGroup

以下代码示例演示了如何使用 Edit-EFSMountTargetSecurityGroup。

适用于 PowerShell V4 的工具

示例 1：更新对指定挂载目标有效的安全组。最多可以指定 5 个，格式为“sg-xxxxxxx”。


```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ModifyMountTargetSecurityGroups](#) 中的。

Get-EFSFileSystem

以下代码示例演示了如何使用 Get-EFSFileSystem。

适用于 PowerShell V4 的工具

示例 1：返回调用方的账户在区域中拥有的所有文件系统的集合。

```
Get-EFSFileSystem
```

输出：

```
CreationTime      : 5/26/2015 4:02:38 PM
CreationToken     : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId      : fs-1a2b3c4d
LifecycleState    : available
Name              :
NumberOfMountTargets : 0
OwnerId           : 123456789012
SizeInBytes       : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime      : 5/26/2015 4:06:23 PM
CreationToken     : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId      : fs-4d3c2b1a
...
```

示例 2：返回指定文件系统的详细信息。

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

示例 3：使用在创建文件系统时指定的幂等性创建令牌返回文件的详细信息。

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeFileSystems](#) 中的。

Get-EFSMountTarget

以下代码示例演示了如何使用 Get-EFSMountTarget。

适用于 PowerShell V4 的工具

示例 1：返回与指定文件系统关联的挂载目标集合。

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

输出：

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifecycleState    : available
MountTargetId     : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId          : 123456789012
SubnetId         : subnet-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeMountTargets](#) 中的。

Get-EFSMountTargetSecurityGroup

以下代码示例演示了如何使用 Get-EFSMountTargetSecurityGroup。

适用于 PowerShell V4 的工具

示例 1：返回安全组的 ID，这些安全组当前分配给与挂载目标关联的网络接口。

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

输出：

```
sg-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeMountTargetSecurityGroups](#) 中的。

Get-EFSTag

以下代码示例演示了如何使用 Get-EFSTag。

适用于 PowerShell V4 的工具

示例 1：返回当前与指定文件系统关联的标签集合。

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

输出：

Key	Value
---	----
Name	My File System
tagkey1	tagvalue1
tagkey2	tagvalue2

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

New-EFSFileSystem

以下代码示例演示了如何使用 New-EFSFileSystem。

适用于 PowerShell V4 的工具

示例 1：创建新的空文件系统。将自动生成用于确保幂等性创建的令牌，并且可以从所返回对象的 **CreationToken** 成员中访问此令牌。

```
New-EFSFileSystem
```

输出：

CreationTime	: 5/26/2015 4:02:38 PM
CreationToken	: 1a2bff54-85e0-4747-bd95-7bc172c4f555

```

FileSystemId      : fs-1a2b3c4d
LifeCycleState    : creating
Name              :
NumberOfMountTargets : 0
OwnerId           : 123456789012
SizeInBytes       : Amazon.ElasticFileSystem.Model.FileSystemSize

```

示例 2：使用自定义令牌创建一个新的空文件系统，以确保幂等性创建。

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateFileSystem](#) 中的。

New-EFSMountTarget

以下代码示例演示了如何使用 New-EFSMountTarget。

适用于 PowerShell V4 的工具

示例 1：为文件系统创建新的挂载目标。将使用指定的子网来确定将在哪个虚拟私有云 (VPC) 中创建挂载目标，以及将自动分配哪个 IP 地址 (来自子网的地址范围)。然后，可以使用分配的 IP 地址将此文件系统挂载到 Amazon EC2 实例上。由于未指定安全组，因此，为目标创建的网络接口与子网的 VPC 的默认安全组相关联。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

输出：

```

FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifeCycleState    : creating
MountTargetId     : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId           : 123456789012
SubnetId          : subnet-1a2b3c4d

```

示例 2：使用自动分配的 IP 地址为指定的文件系统创建新的挂载目标。为挂载目标创建的网络接口与指定的安全组相关联 (最多可以指定 5 个安全组，格式为“sg-xxxxxxx”)。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -  
SecurityGroup sg-group1,sg-group2,sg-group3
```

示例 3：使用指定的 IP 地址为指定的文件系统创建新的挂载目标。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress  
10.0.0.131
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateMountTarget](#) 中的。

New-EFSTag

以下代码示例演示了如何使用 New-EFSTag。

适用于 PowerShell V4 的工具

示例 1：将标签集合应用于指定的文件系统。如果文件系统上已经存在指定了键的标签，则该标签的值将更新。

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag  
@{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

示例 2：为指定的文件系统设置名称标签。使用 Get-EFSFileSystem cmdlet 时，此值会与其他文件系统详细信息一起返回。

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTags](#) 中的。

Remove-EFSFileSystem

以下代码示例演示了如何使用 Remove-EFSFileSystem。

适用于 PowerShell V4 的工具

示例 1：删除不再使用的指定文件系统（如果文件系统有挂载目标，则必须先将其移除）。在 cmdlet 继续运行之前，系统将提示您进行确认 - 要禁止确认，请使用 **-Force** 开关。

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFileSystem](#) 中的。

Remove-EFSMountTarget

以下代码示例演示了如何使用 Remove-EFSMountTarget。

适用于 PowerShell V4 的工具

示例 1：删除指定的挂载目标。在操作继续之前，系统会提示您进行确认。要禁止提示，请使用 **-Force** 开关。请注意，此操作通过目标来强制中断文件系统的所有挂载 - 如果可行，您可能需要考虑在运行此命令之前卸载文件系统。

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteMountTarget](#) 中的。

Remove-EFSTag

以下代码示例演示了如何使用 Remove-EFSTag。

适用于 PowerShell V4 的工具

示例 1：从文件系统中删除包含一个或多个标签的集合。在 cmdlet 继续运行之前，系统将提示您进行确认 - 要禁止确认，请使用 **-Force** 开关。

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTags](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon EKS 示例

以下代码示例向您展示了如何使用带有 Amazon EKS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-EKSResourceTag

以下代码示例演示了如何使用 Add-EKSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 将指定的标签与具有指定 resourceArn 的资源相关联。

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -  
Tag @{Name = "EKSPRODCLUSTER"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResource](#) 中的。

Get-EKSCluster

以下代码示例演示了如何使用 Get-EKSCluster。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 返回有关 Amazon EKS 集群的描述性信息。

```
Get-EKSCluster -Name "PROD"
```

输出：

```
Arn           : arn:aws:eks:us-west-2:012345678912:cluster/PROD  
CertificateAuthority : Amazon.EKS.Model.Certificate
```

```
ClientRequestToken :  
CreatedAt          : 12/25/2019 6:46:17 AM  
Endpoint           : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-  
west-2.eks.amazonaws.com  
Identity           : Amazon.EKS.Model.Identity  
Logging            : Amazon.EKS.Model.Logging  
Name               : PROD  
PlatformVersion    : eks.7  
ResourcesVpcConfig : Amazon.EKS.Model.VpcConfigResponse  
RoleArn            : arn:aws:iam::012345678912:role/eks-iam-role  
Status             : ACTIVE  
Tags               : {}  
Version            : 1.14
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeCluster](#) 中的。

Get-EKSClusterList

以下代码示例演示了如何使用 Get-EKSClusterList。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了您在 AWS 账户 指定区域内的 Amazon EKS 集群。

```
Get-EKSClusterList
```

输出：

```
PROD
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListClusters](#) 中的。

Get-EKSFargateProfile

以下代码示例演示了如何使用 Get-EKSFargateProfile。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 返回有关 Far AWS gate 配置文件的描述性信息。


```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

输出：

```
ClusterName      : TEST
CreatedAt        : 12/26/2019 12:34:47 PM
FargateProfileArn : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors        : {Amazon.EKS.Model.FargateProfileSelector}
Status           : ACTIVE
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeFargateProfile](#) 中的。

Get-EKSFargateProfileList

以下代码示例演示了如何使用 Get-EKSFargateProfileList。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了与指定区域中指定集群关联的 AWS Fargate 配置文件。AWS 账户

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

输出：

```
EKSFargate
EKSFargateProfile
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListFargateProfiles](#) 中的。

Get-EKSNodegroup

以下代码示例演示了如何使用 Get-EKSNodegroup。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 返回有关 Amazon EKS 节点组的描述性信息。

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

输出：

```
AmiType       : AL2_x86_64
ClusterName   : PROD
CreatedAt     : 12/25/2019 10:16:45 AM
DiskSize      : 40
Health        : Amazon.EKS.Model.NodegroupHealth
InstanceTypes : {t3.large}
Labels        : {}
ModifiedAt    : 12/25/2019 10:16:45 AM
NodegroupArn  : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status        : CREATING
Subnets       : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags          : {}
Version        : 1.14
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeNodegroup](#) 中的。

Get-EKSNodegroupList

以下代码示例演示了如何使用 Get-EKSNodegroupList。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了与您在 AWS 账户 指定区域中的指定集群关联的 Amazon EKS 节点组。

```
Get-EKSNodegroupList -ClusterName PROD
```

输出：

```
ProdEKSNodeGroup
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListNodegroups](#) 中的。

Get-EKSResourceTag

以下代码示例演示了如何使用 Get-EKSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出 Amazon EKS 资源的标签。

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

输出：

```
Key  Value
---  -
Name EKSPRODCLUSTER
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListTagsForResource](#) 中的。

Get-EKSUpdate

以下代码示例演示了如何使用 Get-EKSUpdate。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 返回有关针对 Amazon EKS 集群或关联的托管式节点组进行的更新的描述性信息。

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

输出：

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeUpdate](#) 中的。

Get-EKSUpdateList

以下代码示例演示了如何使用 Get-EKSUpdateList。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了在指定区域中与您的 AWS 账户的 Amazon EKS 集群或托管节点组相关的更新。

```
Get-EKSUpdateList -Name "PROD"
```

输出：

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListUpdates](#) 中的。

New-EKSCluster

以下代码示例演示了如何使用 New-EKSCluster。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为“prod”的新集群。

```
New-EKSCluster -Name prod -ResourcesVpcConfig
@{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
-RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

输出：

```

Arn                : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/10/2018 9:25:31 PM
Endpoint            :
Name                : prod
PlatformVersion     : eks.3
ResourcesVpcConfig  : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678901:role/eks-service-role
Status              : CREATING
Version             : 1.10

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateCluster](#) 中的。

New-EKSFargateProfile

以下代码示例演示了如何使用 New-EKSFargateProfile。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为您的 Amazon EKS 集群创建 AWS Fargate 配置文件。您必须在集群中至少有一个 Fargate 配置文件，才能在 Fargate 基础设施上安排容器组 (pod)。

```

New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
@{Namespace="default"}

```

输出：

```

ClusterName        : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}

```

```
Status          : CREATING
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateFargateProfile](#) 中的。

New-EKSNodeGroup

以下代码示例演示了如何使用 New-EKSNodeGroup。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为 Amazon EKS 集群创建托管式 Worker 节点组。您只能为集群创建等于集群的当前 Kubernetes 版本的节点组。创建的所有节点组对于集群的相应 Kubernetes 次要版本均具有最新的 AMI 发行版。

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
-DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
-NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
"subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

输出：

```
AmiType          : AL2_x86_64
ClusterName      : PROD
CreatedAt        : 12/25/2019 10:16:45 AM
DiskSize         : 40
Health           : Amazon.EKS.Model.NodegroupHealth
InstanceTypes    : {t3.large}
Labels           : {}
ModifiedAt       : 12/25/2019 10:16:45 AM
NodegroupArn     : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName    : ProdEKSNodeGroup
NodeRole         : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion   : 1.14.7-20190927
RemoteAccess     :
Resources        :
ScalingConfig    : Amazon.EKS.Model.NodegroupScalingConfig
Status           : CREATING
```

```
Subnets      : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags          : {}
Version       : 1.14
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateNodegroup](#) 中的。

Remove-EKSCluster

以下代码示例演示了如何使用 Remove-EKSCluster。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除 Amazon EKS 集群控制面板。

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

输出：

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

Arn                : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/25/2019 9:33:25 AM
Endpoint            : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-west-2.eks.amazonaws.com
Identity            : Amazon.EKS.Model.Identity
Logging             : Amazon.EKS.Model.Logging
Name                : DEV-KUBE-CL
PlatformVersion     : eks.7
ResourcesVpcConfig  : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678912:role/eks-iam-role
Status              : DELETING
Tags                : {}
Version             : 1.14
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteCluster](#) 中的。

Remove-EKSFargateProfile

以下代码示例演示了如何使用 Remove-EKSFargateProfile。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除了 Fargate 配置文件 AWS。删除 Fargate 配置文件时，将删除使用该配置文件创建的在 Fargate 上运行的任何容器组 (pod)。

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

输出：

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

ClusterName      : TEST
CreatedAt        : 12/26/2019 12:34:47 PM
FargateProfileArn : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors        : {Amazon.EKS.Model.FargateProfileSelector}
Status           : DELETING
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFargateProfile](#) 中的。

Remove-EKSNodegroup

以下代码示例演示了如何使用 Remove-EKSNodegroup。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除集群的 Amazon EKS 节点组。

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

输出：

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

AmiType           : AL2_x86_64
ClusterName       : PROD
CreatedAt          : 12/25/2019 10:16:45 AM
DiskSize           : 40
Health             : Amazon.EKS.Model.NodegroupHealth
InstanceTypes     : {t3.large}
Labels            : {}
ModifiedAt         : 12/25/2019 11:01:16 AM
NodegroupArn       : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName     : ProdEKSNodeGroup
NodeRole           : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion     : 1.14.7-20190927
RemoteAccess       :
Resources          : Amazon.EKS.Model.NodegroupResources
ScalingConfig      : Amazon.EKS.Model.NodegroupScalingConfig
Status             : DELETING
Subnets           : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags               : {}
Version            : 1.14
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteNodegroup](#) 中的。

Remove-EKSResourceTag

以下代码示例演示了如何使用 Remove-EKSResourceTag。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 从 EKS 资源中删除指定的标签。

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
-TagKey "Name"
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
"arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagResource](#) 中的。

Update-EKSClusterConfig

以下代码示例演示了如何使用 Update-EKSClusterConfig。

适用于 PowerShell V4 的工具

示例 1：更新 Amazon EKS 集群配置。在更新期间，集群将继续运行。

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
@{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

输出：

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateClusterConfig](#) 中的。

Update-EKSClusterVersion

以下代码示例演示了如何使用 Update-EKSClusterVersion。

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 将 Amazon EKS 集群更新到指定的 Kubernetes 版本。在更新期间，集群将继续运行。

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

输出：

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateClusterVersion](#) 中的。

Elastic Load Balancing-使用适用于 PowerShell V4 的工具的版本 1 示例

以下代码示例向您展示了如何使用带有 Elastic Load Balancing 的 AWS Tools for PowerShell V4-版本 1 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-ELBLoadBalancerToSubnet

以下代码示例演示了如何使用 Add-ELBLoadBalancerToSubnet。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的子网添加到为指定的负载均衡器配置的子网集。输出包括子网的完整列表。

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

输出：

```
subnet-12345678
subnet-87654321
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AttachLoadBalancerToSubnets](#) 中的。

Add-ELBResourceTag

以下代码示例演示了如何使用 Add-ELBResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的标签添加到指定的负载均衡器。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
@{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 为标签参数创建标签。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
```

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddTags](#) 中的。

Disable-ELBAvailabilityZoneForLoadBalancer

以下代码示例演示了如何使用 Disable-ELBAvailabilityZoneForLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的负载均衡器移除指定的可用区。输出包括剩余的可用区。

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -  
AvailabilityZone us-west-2a
```

输出：

```
us-west-2b
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DisableAvailabilityZonesForLoadBalancer](#) 中的。

Dismount-ELBLoadBalancerFromSubnet

以下代码示例演示了如何使用 Dismount-ELBLoadBalancerFromSubnet。

适用于 PowerShell V4 的工具

示例 1：此示例从为指定负载均衡器配置的子网集中删除指定的子网。输出包括其余子网。

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet  
subnet-12345678
```

输出：

```
subnet-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachLoadBalancerFromSubnets](#) 中的。

Edit-ELBLoadBalancerAttribute

以下代码示例演示了如何使用 Edit-ELBLoadBalancerAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的负载均衡器启用跨可用区负载均衡。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
CrossZoneLoadBalancing_Enabled $true
```

示例 2：此示例为指定的负载均衡器禁用连接耗尽。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
ConnectionDraining_Enabled $false
```

示例 3：此示例为指定的负载均衡器启用访问日志记录。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ModifyLoadBalancerAttributes](#) 中的。

Enable-ELBAvailabilityZoneForLoadBalancer

以下代码示例演示了如何使用 Enable-ELBAvailabilityZoneForLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的可用区添加到指定的负载均衡器。输出包括可用区的完整列表。

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -  
AvailabilityZone us-west-2a
```

输出：

```
us-west-2a
```

```
us-west-2b
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) EnableAvailabilityZonesForLoadBalancer](#) 中的。

Get-ELBInstanceHealth

以下代码示例演示了如何使用 Get-ELBInstanceHealth。

适用于 PowerShell V4 的工具

示例 1：此示例描述向指定负载均衡器注册的实例的状态。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

输出：

Description State	InstanceId	ReasonCode
-----	-----	-----

N/A	i-87654321	N/A
InService		
Instance has failed at lea... i-12345678		Instance
OutOfService		

示例 2：此示例描述向指定负载均衡器注册的指定实例的状态。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

示例 3：此示例显示指定实例的状态的完整描述。

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678).Description
```

输出：

```
Instance has failed at least the UnhealthyThreshold number of health checks consecutively.
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeInstanceHealth](#) 中的。

Get-ELBLoadBalancer

以下代码示例演示了如何使用 Get-ELBLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例列出负载均衡器的名称。

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

输出：

```
LoadBalancerName
-----
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

示例 2：此示例描述指定的负载均衡器。

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

输出：

```
AvailabilityZones      : {us-west-2a, us-west-2b}
BackendServerDescriptions :
  {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime            : 4/11/2015 12:12:45 PM
DNSName                 : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
HealthCheck             : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances               : {i-207d9717, i-afefb49b}
ListenerDescriptions    : {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName        : my-load-balancer
Policies                : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                  : internet-facing
SecurityGroups          : {sg-a61988c3}
```



```
SourceSecurityGroup      : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                 : {subnet-15aaab61}
VPCId                    : vpc-a01106c2
```

示例 3：此示例描述了您在当前 AWS 区域中的所有负载均衡器。

```
Get-ELBLoadBalancer
```

示例 4：此示例描述所有可用 AWS 区域中的所有负载均衡器。

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeLoadBalancers](#) 中的。

Get-ELBLoadBalancerAttribute

以下代码示例演示了如何使用 Get-ELBLoadBalancerAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定负载均衡器的属性。

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

输出：

```
AccessLog                : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes      : {}
ConnectionDraining       : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings       : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing    : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeLoadBalancerAttributes](#) 中的。

Get-ELBLoadBalancerPolicy

以下代码示例演示了如何使用 Get-ELBLoadBalancerPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例描述与指定负载均衡器关联的策略。

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

输出：

PolicyAttributeDescriptions PolicyTypeName	PolicyName
-----	-----

{ProxyProtocol}	my-ProxyProtocol-policy
ProxyProtocolPolicyType	
{CookieName}	my-app-cookie-policy
AppCookieStickinessPolicyType	

示例 2：此示例描述指定策略的属性。

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-ProxyProtocol-policy).PolicyAttributeDescriptions
```

输出：

AttributeName	AttributeValue
-----	-----
ProxyProtocol	true

示例 3：此示例描述预定义的策略，包括示例策略。示例策略的名称带有 ELBSample-前缀。

```
Get-ELBLoadBalancerPolicy
```

输出：

PolicyAttributeDescriptions PolicyTypeName	PolicyName
-----	-----

{Protocol-SSLv2, Protocol-TLSv1, Pro...	ELBSecurityPolicy-2015-05
SSLNegotiationPolicyType	

```
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
  SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
  SSLNegotiationPolicyType
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeLoadBalancerPolicies](#) 中的。

Get-ELBLoadBalancerPolicyType

以下代码示例演示了如何使用 Get-ELBLoadBalancerPolicyType。

适用于 PowerShell V4 的工具

示例 1：此示例获取弹性负载均衡支持的策略类型。

```
Get-ELBLoadBalancerPolicyType
```

输出：

Description	PolicyAttributeTypeDescriptions
PolicyTypeName	
-----	-----

Stickiness policy with session lifet...	{CookieExpirationPeriod}
LBCookieStickinessPolicyType	
Policy that controls authentication ...	{PublicKeyPolicyName}
BackendServerAuthenticationPolicyType	
Listener policy that defines the cip...	{Protocol-SSLv2, Protocol-TLSv1, Pro...
SSLNegotiationPolicyType	
Policy containing a list of public k...	{PublicKey}
PublicKeyPolicyType	

```
Stickiness policy with session lifet... {CookieName}
AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
ProxyProtocolPolicyType
```

示例 2：此示例描述指定的策略类型。

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

输出：

Description	PolicyAttributeTypeDescriptions
PolicyTypeName	
-----	-----

Policy that controls whether to incl... {ProxyProtocol}	
ProxyProtocolPolicyType	

示例 3：此示例显示指定策略类型的完整描述。

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

输出：

```
Policy that controls whether to include the IP address and port of the originating
request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeLoadBalancerPolicyTypes](#) 中的。

Get-ELBResourceTag

以下代码示例演示了如何使用 Get-ELBResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例列出指定负载均衡器的标签。

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-balancer")
```

输出：

LoadBalancerName	Tags
-----	----
my-load-balancer	{project, department}
my-internal-load-balancer	{project, department}

示例 2：此示例描述指定负载均衡器的标签。

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

输出：

Key	Value
---	-----
project	lima
department	digital-media

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

Join-ELBSecurityGroupToLoadBalancer

以下代码示例演示了如何使用 Join-ELBSecurityGroupToLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定安全组替换指定负载均衡器的当前安全组。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

输出：

```
sg-87654321
```

示例 2：要保留当前安全组并指定其它安全组，请同时指定现有安全组和新安全组。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup @("sg-12345678", "sg-87654321")
```

输出：

```
sg-12345678  
sg-87654321
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ApplySecurityGroupsToLoadBalancer](#) 中的。

New-ELBAppCookieStickinessPolicy

以下代码示例演示了如何使用 New-ELBAppCookieStickinessPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个粘性策略，该策略遵循指定的应用程序生成的 Cookie 的粘性会话生命周期。

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-  
app-cookie-policy -CookieName my-app-cookie
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateAppCookieStickinessPolicy](#) 中的。

New-ELBLBCookieStickinessPolicy

以下代码示例演示了如何使用 New-ELBLBCookieStickinessPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个粘性策略，其粘性会话生命周期由指定的有效期限控制（以秒为单位）。

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-  
duration-cookie-policy -CookieExpirationPeriod 60
```

示例 2：此示例创建一个粘性策略，其粘性会话生命周期由浏览器（用户代理）的生命周期控制。

```
New-ELBLCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateLbCookieStickinessPolicy](#) 中的。

New-ELBLoadBalancer

以下代码示例演示了如何使用 New-ELBLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例在 VPC 中创建带有 HTTP 侦听器的负载均衡器。

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

示例 2：此示例在 EC2-Classic 中创建了一个带有 HTTP 侦听器的负载均衡器。

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-west-2a -Listener $httpListener
```

输出：

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

示例 3：此示例创建带有 HTTPS 侦听器的负载均衡器。

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
```

```
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
-Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateLoadBalancer](#) 中的。

New-ELBLoadBalancerListener

以下代码示例演示了如何使用 New-ELBLoadBalancerListener。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的负载均衡器添加 HTTPS 侦听器。

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
$httpsListener
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateLoadBalancerListeners](#) 中的。

New-ELBLoadBalancerPolicy

以下代码示例演示了如何使用 New-ELBLoadBalancerPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的负载均衡器创建一个新的代理协议策略。


```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -Property
@{
    AttributeName="ProxyProtocol"
    AttributeValue="True"
}
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
$attribute
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateLoadBalancerPolicy](#) 中的。

Register-ELBInstanceWithLoadBalancer

以下代码示例演示了如何使用 Register-ELBInstanceWithLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的负载均衡器注册指定 EC2 实例。

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
i-12345678
```

输出：

```
InstanceId
-----
i-12345678
i-87654321
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterInstancesWithLoadBalancer](#) 中的。

Remove-ELBInstanceFromLoadBalancer

以下代码示例演示了如何使用 Remove-ELBInstanceFromLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的负载均衡器中移除指定 EC2 实例。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance  
i-12345678
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ELBInstanceFromLoadBalancer  
(DeregisterInstancesFromLoadBalancer)" on Target  
"Amazon.ElasticLoadBalancing.Model.Instance".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):  
  
InstanceId  
-----  
i-87654321
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeregisterInstancesFromLoadBalancer](#) 中的。

Remove-ELBLoadBalancer

以下代码示例演示了如何使用 Remove-ELBLoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的负载均衡器。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

输出：

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-  
load-balancer".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteLoadBalancer](#) 中的。

Remove-ELBLoadBalancerListener

以下代码示例演示了如何使用 Remove-ELBLoadBalancerListener。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定负载均衡器的端口 80 上的侦听器。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort 80
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener (DeleteLoadBalancerListeners)"
on Target "80".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteLoadBalancerListeners](#) 中的。

Remove-ELBLoadBalancerPolicy

以下代码示例演示了如何使用 Remove-ELBLoadBalancerPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的负载均衡器中删除指定的策略。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
Target "my-duration-cookie-policy".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteLoadBalancerPolicy](#) 中的。

Remove-ELBResourceTag

以下代码示例演示了如何使用 Remove-ELBResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的负载均衡器移除指定的标签。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
"Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

示例 2：对于 PowerShell 版本 2，必须使用 New-Object 为 Tag 参数创建标签。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RemoveTags](#) 中的。

Set-ELBHealthCheck

以下代码示例演示了如何使用 Set-ELBHealthCheck。

适用于 PowerShell V4 的工具

示例 1：此示例配置指定负载均衡器的运行状况检查设置。

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

输出：

```
HealthyThreshold    : 2
Interval            : 30
Target              : HTTP:80/ping
Timeout             : 3
UnhealthyThreshold  : 2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ConfigureHealthCheck](#) 中的。

Set-ELBLoadBalancerListenerSSLCertificate

以下代码示例演示了如何使用 Set-ELBLoadBalancerListenerSSLCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例替换用于终止指定侦听器的 SSL 连接的证书。

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) SetLoadBalancerListenerSslCertificate](#) 中的。

Set-ELBLoadBalancerPolicyForBackendServer

以下代码示例演示了如何使用 Set-ELBLoadBalancerPolicyForBackendServer。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的策略替换指定端口的策略。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

示例 2：此示例移除与指定端口关联的所有策略。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetLoadBalancerPoliciesForBackendServer](#) 中的。

Set-ELBLoadBalancerPolicyOfListener

以下代码示例演示了如何使用 Set-ELBLoadBalancerPolicyOfListener。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定策略替换指定侦听器的策略。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

示例 2：此示例移除与指定侦听器关联的所有策略。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetLoadBalancerPoliciesOfListener](#) 中的。

Elastic Load Balancing-使用适用于 PowerShell V4 的工具的版本 2 示例

以下代码示例向您展示了如何使用带有 Elastic Load Balancing 的 AWS Tools for PowerShell V4-版本 2 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-ELB2ListenerCertificate

以下代码示例演示了如何使用 Add-ELB2ListenerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例向指定侦听器添加附加证书。

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618' -
Certificate @{CertificateArn = 'arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

输出：

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
False
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddListenerCertificates](#) 中的。

Add-ELB2Tag

以下代码示例演示了如何使用 Add-ELB2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的 **AWS.Tools.ElasticLoadBalancingV2** 资源添加新标签。

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key = 'productVersion'; Value = '1.0.0'}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddTags](#) 中的。

Edit-ELB2Listener

以下代码示例演示了如何使用 Edit-ELB2Listener。

适用于 PowerShell V4 的工具

示例 1：此示例将指定侦听器的默认操作修改为固定响应。

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
        "ContentType" = "text/plain"
        "MessageBody" = "Hello World"
        "StatusCode" = "200"
    }
    "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port 8080 -DefaultAction $newDefaultAction
```

输出：

```
Certificates      : {}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
```



```

ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port             : 8080
Protocol         : HTTP
SslPolicy        :

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyListener](#) 中的。

Edit-ELB2LoadBalancerAttribute

以下代码示例演示了如何使用 Edit-ELB2LoadBalancerAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定负载均衡器的属性。

```

Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
'deletion_protection.enabled'; Value = 'true'}

```

输出：

Key	Value
---	----
deletion_protection.enabled	true
access_logs.s3.enabled	false
access_logs.s3.bucket	
access_logs.s3.prefix	
idle_timeout.timeout_seconds	60
routing.http2.enabled	true
routing.http.drop_invalid_header_fields.enabled	false

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyLoadBalancerAttributes](#) 中的。

Edit-ELB2Rule

以下代码示例演示了如何使用 Edit-ELB2Rule。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定的侦听器规则配置。

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
        "Values" = "/login1","/login2","/login3"
    }
    "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc' -Condition $newRuleCondition
```

输出：

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 10
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyRule](#) 中的。

Edit-ELB2TargetGroup

以下代码示例演示了如何使用 Edit-ELB2TargetGroup。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定目标组的属性。

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

输出：

```
HealthCheckEnabled      : True
```

```

HealthCheckIntervalSeconds : 60
HealthCheckPath             : /index.html
HealthCheckPort             : 8080
HealthCheckProtocol         : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount       : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                     : vpc-2cfd7000

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyTargetGroup](#) 中的。

Edit-ELB2TargetGroupAttribute

以下代码示例演示了如何使用 Edit-ELB2TargetGroupAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定目标组的 deregistration_delay 属性。

```

Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
'deregistration_delay.timeout_seconds'; Value = 600}

```

输出：

Key	Value
---	----
stickiness.enabled	false
deregistration_delay.timeout_seconds	600
stickiness.type	lb_cookie
stickiness.lb_cookie.duration_seconds	86400
slow_start.duration_seconds	0
load_balancing.algorithm.type	round_robin

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyTargetGroupAttributes](#) 中的。

Get-ELB2AccountLimit

以下代码示例演示了如何使用 Get-ELB2AccountLimit。

适用于 PowerShell V4 的工具

示例 1：此命令列出给定区域的 ELB2 账户限制。

```
Get-ELB2AccountLimit
```

输出：

```
Max  Name
---  ---
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeAccountLimits](#) 中的。

Get-ELB2Listener

以下代码示例演示了如何使用 Get-ELB2Listener。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定 ALB/NLB 的侦听器。

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

输出：

```
Certificates      : {}
DefaultActions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn       : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn   : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port              : 80
Protocol          : HTTP
SslPolicy         :

Certificates      : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn       : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn   : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port              : 443
Protocol          : HTTPS
SslPolicy         : ELBSecurityPolicy-2016-08
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeListeners](#) 中的。

Get-ELB2ListenerCertificate

以下代码示例演示了如何使用 Get-ELB2ListenerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定侦听器的证书。

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

输出：

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
True
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeListenerCertificates](#) 中的。

Get-ELB2LoadBalancer

以下代码示例演示了如何使用 Get-ELB2LoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例显示给定区域的所有负载均衡器。

```
Get-ELB2LoadBalancer
```

输出：

```
AvailabilityZones      : {us-east-1c}
CanonicalHostedZoneId  : Z26RNL4JYFT0TI
CreatedTime            : 6/22/18 11:21:50 AM
DNSName                : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType          : ipv4
LoadBalancerArn        : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName       : test-elb1234567890
Scheme                : internet-facing
SecurityGroups         : {}
State                  : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                   : network
VpcId                  : vpc-2cf00000
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeLoadBalancers](#) 中的。

Get-ELB2LoadBalancerAttribute

以下代码示例演示了如何使用 Get-ELB2LoadBalancerAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述给定负载均衡器的属性。

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

输出：

Key	Value
---	----
access_logs.s3.enabled	false
load_balancing.cross_zone.enabled	true
access_logs.s3.prefix	
deletion_protection.enabled	false
access_logs.s3.bucket	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeLoadBalancerAttributes](#) 中的。

Get-ELB2Rule

以下代码示例演示了如何使用 Get-ELB2Rule。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定侦听器 ARN 的侦听器规则。

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

输出：

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
```

```

Priority      : 1
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 2
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {}
IsDefault    : True
Priority      : default
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeRules](#) 中的。

Get-ELB2SSLPolicy

以下代码示例演示了如何使用 Get-ELB2SSLPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例列出了 ElasticLoadBalancing V2 的所有可用侦听器策略。

```
Get-ELB2SSLPolicy
```

输出：

Ciphers	Name	SslProtocols
-----	----	-----
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-SHA256, ECDHE-RSA-AES128-SHA256}	ELBSecurityPolicy-2016-08	{TLSv1, TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-SHA256, ECDHE-RSA-AES128-SHA256}	ELBSecurityPolicy-TLS-1-2-2017-01	{TLSv1.2}


```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01      {TLSv1.1,
  TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06          {TLSv1,
  TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05             {TLSv1,
  TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
  TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08     {TLSv1.1,
  TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08     {TLSv1.2}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeSslPolicies](#) 中的。

Get-ELB2Tag

以下代码示例演示了如何使用 Get-ELB2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例列出指定资源的标签。

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

输出：

```
ResourceArn
          Tags
-----
-----
          ----
```

```
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

Get-ELB2TargetGroup

以下代码示例演示了如何使用 Get-ELB2TargetGroup。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的目标组。

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

输出：

```
HealthCheckEnabled      : True
HealthCheckIntervalSeconds : 30
HealthCheckPath         : /
HealthCheckPort         : traffic-port
HealthCheckProtocol     : HTTP
HealthCheckTimeoutSeconds : 5
HealthyThresholdCount   : 5
LoadBalancerArns        : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                 : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                    : 80
Protocol                : HTTP
TargetGroupArn          : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName         : test-tg
TargetType              : instance
UnhealthyThresholdCount : 2
VpcId                   : vpc-2cfd7000
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTargetGroups](#) 中的。

Get-ELB2TargetGroupAttribute

以下代码示例演示了如何使用 Get-ELB2TargetGroupAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例描述指定目标组的属性。

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

输出：

Key	Value
---	-----
stickiness.enabled	false
deregistration_delay.timeout_seconds	300
stickiness.type	lb_cookie
stickiness.lb_cookie.duration_seconds	86400
slow_start.duration_seconds	0
load_balancing.algorithm.type	round_robin

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeTargetGroupAttributes](#) 中的。

Get-ELB2TargetHealth

以下代码示例演示了如何使用 Get-ELB2TargetHealth。

适用于 PowerShell V4 的工具

示例 1：此示例返回指定目标组中存在的目标的运行状况。

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

输出：

HealthCheckPort	Target	TargetHealth
-----	-----	-----

```
80          Amazon.ElasticLoadBalancingV2.Model.TargetDescription
          Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeTargetHealth](#) 中的。

New-ELB2Listener

以下代码示例演示了如何使用 New-ELB2Listener。

适用于 PowerShell V4 的工具

示例 1：此示例创建新的 ALB 侦听器，其默认操作为“转发”，以便将流量发送到指定目标组。

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    ForwardConfig = @{
        TargetGroups = @(
            @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
        )
        TargetGroupStickinessConfig = @{
            DurationSeconds = 900
            Enabled = $true
        }
    }
    Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -Protocol
"HTTP" -DefaultAction $defaultAction
```

输出：

```
Certificates      : {}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port             : 8001
Protocol         : HTTP
```

SslPolicy :

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateListener](#) 中的。

New-ELB2LoadBalancer

以下代码示例演示了如何使用 New-ELB2LoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例创建具有两个子网、面向互联网的新应用程序负载均衡器。

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
  ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
  c37a67a6', 'subnet-fc02eea0'
```

输出：

```
AvailabilityZones      : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXD0TRQ7X7K
CreatedTime            : 12/28/19 2:58:03 PM
DNSName                : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType          : ipv4
LoadBalancerArn        : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName       : New-Test-ALB
Scheme                 : internet-facing
SecurityGroups         : {sg-07c3414abb8811cbd}
State                  : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                   : application
VpcId                  : vpc-2cfd7000
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateLoadBalancer](#) 中的。

New-ELB2Rule

以下代码示例演示了如何使用 New-ELB2Rule。

适用于 PowerShell V4 的工具

示例 1：此示例为指定侦听器创建基于客户标头值且具有固定响应操作的新侦听器规则。

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
        "ContentType" = "text/plain"
        "MessageBody" = "Hello World"
        "StatusCode" = "200"
    }
    "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "httpHeaderConfig" = @{
        "HttpHeaderName" = "customHeader"
        "Values" = "header2","header1"
    }
    "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -Action
$newRuleAction -Condition $newRuleCondition -Priority 10
```

输出：

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority     : 10
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateRule](#) 中的。

New-ELB2TargetGroup

以下代码示例演示了如何使用 New-ELB2TargetGroup。

适用于 PowerShell V4 的工具

示例 1：此示例使用提供的参数创建新的目标组。

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

输出：

```
HealthCheckEnabled      : True
HealthCheckIntervalSeconds : 30
HealthCheckPath         : /index.html
HealthCheckPort         : 80
HealthCheckProtocol     : HTTP
HealthCheckTimeoutSeconds : 5
HealthyThresholdCount   : 2
LoadBalancerArns       : {}
Matcher                 : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                    : 80
Protocol                : HTTP
TargetGroupArn          : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName         : NewTargetGroup
TargetType              : instance
UnhealthyThresholdCount : 5
VpcId                   : vpc-2cfd7000
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateTargetGroup](#) 中的。

Register-ELB2Target

以下代码示例演示了如何使用 Register-ELB2Target。

适用于 PowerShell V4 的工具

示例 1：此示例将实例“i-0672a4c4cdeae3111”注册到指定目标组。

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id = 'i-0672a4c4cdeae3111'}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterTargets](#) 中的。

Remove-ELB2Listener

以下代码示例演示了如何使用 Remove-ELB2Listener。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的侦听器。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

示例 2：此示例从指定的负载均衡器中移除指定的侦听器。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
```



```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteListener](#) 中的。

Remove-ELB2ListenerCertificate

以下代码示例演示了如何使用 Remove-ELB2ListenerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的目标组移除指定的证书。

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
(RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveListenerCertificates](#) 中的。

Remove-ELB2LoadBalancer

以下代码示例演示了如何使用 Remove-ELB2LoadBalancer。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的负载均衡器。

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteLoadBalancer](#) 中的。

Remove-ELB2Rule

以下代码示例演示了如何使用 Remove-ELB2Rule。

适用于 PowerShell V4 的工具

示例 1：此示例从侦听器中移除指定的规则

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRule](#) 中的。

Remove-ELB2Tag

以下代码示例演示了如何使用 Remove-ELB2Tag。

适用于 PowerShell V4 的工具

示例 1：此示例移除指定键的标签。

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey 'productVersion'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveTags](#) 中的。

Remove-ELB2TargetGroup

以下代码示例演示了如何使用 Remove-ELB2TargetGroup。

适用于 PowerShell V4 的工具

示例 1：此示例移除指定的目标组。

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

输出：

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteTargetGroup](#) 中的。

Set-ELB2IpAddressType

以下代码示例演示了如何使用 Set-ELB2IpAddressType。

适用于 PowerShell V4 的工具

示例 1：此示例将负载均衡器 IP 地址类型从“IPv4”更改为“DualStack”。

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
dualstack
```

输出：

```
Value
-----
dualstack
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetIpAddressType](#) 中的。

Set-ELB2RulePriority

以下代码示例演示了如何使用 Set-ELB2RulePriority。

适用于 PowerShell V4 的工具

示例 1：此示例更改指定侦听器规则的优先级。

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

输出：

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority     : 11
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SetRulePriorities](#) 中的。

Set-ELB2SecurityGroup

以下代码示例演示了如何使用 Set-ELB2SecurityGroup。

适用于 PowerShell V4 的工具

示例 1：此示例将安全组“sg-07c3414abb8811cbd”添加到指定负载均衡器。

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
'sg-07c3414abb8811cbd'
```

输出：

```
sg-07c3414abb8811cbd
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SetSecurityGroups](#) 中的。

Set-ELB2Subnet

以下代码示例演示了如何使用 Set-ELB2Subnet。

适用于 PowerShell V4 的工具

示例 1：此示例修改指定负载均衡器的子网。

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet 'subnet-7d8a0a51', 'subnet-c37a67a6'
```

输出：

LoadBalancerAddresses	SubnetId	ZoneName
-----	-----	-----
{}	subnet-7d8a0a51	us-east-1c
{}	subnet-c37a67a6	us-east-1b

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SetSubnets](#) 中的。

Unregister-ELB2Target

以下代码示例演示了如何使用 Unregister-ELB2Target。

适用于 PowerShell V4 的工具

示例 1：此示例从指定目标组中取消注册实例“i-0672a4c4cdeae3111”。

```
$targetDescription = New-Object
    Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
    'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeregisterTargets](#) 中的。

使用适用于 PowerShell V4 的工具的亚马逊 FSx 示例

以下代码示例向您展示了如何使用带有 Amazon FSx 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-FSXResourceTag

以下代码示例演示了如何使用 Add-FSXResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例向给定的资源添加标签。

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

输出：

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResource](#) 中的。

Get-FSXBackup

以下代码示例演示了如何使用 Get-FSXBackup。

适用于 PowerShell V4 的工具

示例 1：此示例获取自昨天以来为给定文件系统 ID 创建的备份。

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-Object CreationTime -gt (Get-Date).AddDays(-1)
```

输出：

```
BackupId      : backup-01dac234e56782bcc
```

```

CreationTime      : 6/14/2019 3:35:14 AM
FailureDetails    :
FileSystem        : Amazon.FSx.Model.FileSystem
KmsKeyId          : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f1-
e1234c5af123
Lifecycle        : AVAILABLE
ProgressPercent   : 100
ResourceARN       : arn:aws:fsx:eu-west-1:123456789012:backup/backup-01dac234e56782bcc
Tags              : {}
Type              : AUTOMATIC

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeBackups](#) 中的。

Get-FSXFileSystem

以下代码示例演示了如何使用 Get-FSXFileSystem。

适用于 PowerShell V4 的工具

示例 1：此示例返回给定 filesystemId 的描述。

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

输出：

```

CreationTime      : 1/17/2019 9:55:30 AM
DNSName           : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails    :
FileSystemId      : fs-01cd23bc4bdf5678a
FileSystemType    : WINDOWS
KmsKeyId          : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-8bde-
a9f0-e1234c5af678
Lifecycle        : AVAILABLE
LustreConfiguration :
NetworkInterfaceIds : {eni-07d1dda1322b7e209}
OwnerId          : 123456789012
ResourceARN       : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a
StorageCapacity   : 300
SubnetIds         : {subnet-7d123456}
Tags              : {FSx-Service}

```



```
VpcId           : vpc-41cf2b3f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeFileSystems](#) 中的。

Get-FSXResourceTagList

以下代码示例演示了如何使用 Get-FSXResourceTagList。

适用于 PowerShell V4 的工具

示例 1：此示例列出所提供的资源 arn 的标签。

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

输出：

Key	Value
---	----
FSx-Service	Windows
Users	Dev

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListTagsForResource](#) 中的。

New-FSXBackup

以下代码示例演示了如何使用 New-FSXBackup。

适用于 PowerShell V4 的工具

示例 1：此示例创建给定文件系统的备份。

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

输出：

```
BackupId       : backup-0b1fac2345623456ba
CreationTime    : 6/14/2019 5:37:17 PM
```

```

FailureDetails      :
FileSystem          : Amazon.FSx.Model.FileSystem
KmsKeyId            : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f3-
e1234c5af678
Lifecycle          : CREATING
ProgressPercent     : 0
ResourceARN         : arn:aws:fsx:eu-west-1:123456789012:backup/
backup-0b1fac2345623456ba
Tags                : {}
Type                : USER_INITIATED

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateBackup](#) 中的。

New-FSXFileSystem

以下代码示例演示了如何使用 New-FSXFileSystem。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个新的 300 GB Windows 文件系统，容许从指定的子网进行访问，支持高达每秒 8 兆字节的吞吐量。新的文件系统将自动加入指定的 Microsoft Active Directory 中。

```

New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}

```

输出：

```

CreationTime        : 12/10/2018 6:06:59 PM
DNSName             : fs-abcdef01234567890.example.com
FailureDetails      :
FileSystemId        : fs-abcdef01234567890
FileSystemType      : WINDOWS
KmsKeyId            : arn:aws:kms:us-west-2:123456789012:key/a1234567-252c-45e9-
afaa-123456789abc
Lifecycle          : CREATING
LustreConfiguration :
NetworkInterfaceIds : {}
OwnerId            : 123456789012
ResourceARN         : arn:aws:fsx:us-west-2:123456789012:file-system/fs-
abcdef01234567890

```

```
StorageCapacity      : 300
SubnetIds            : {subnet-1a2b3c4d5e6f}
Tags                 : {}
VpcId                : vpc-1a2b3c4d5e6f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateFileSystem](#) 中的。

New-FSXFileSystemFromBackup

以下代码示例演示了如何使用 New-FSXFileSystemFromBackup。

适用于 PowerShell V4 的工具

示例 1：此示例使用现有的 Amazon FSx for Windows 文件服务器备份创建新的亚马逊 FSx 文件系统。

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

输出：

```
CreationTime      : 8/8/2019 12:59:58 PM
DNSName           : fs-012ff34e56789120.ktmsad.local
FailureDetails    :
FileSystemId      : fs-012ff34e56789120
FileSystemType    : WINDOWS
KmsKeyId          : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-1bde-
a2f3-e4567c8a9321
Lifecycle        : CREATING
LustreConfiguration :
NetworkInterfaceIds : {}
OwnerId           : 933303704102
ResourceARN       : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-012ff34e56789120
StorageCapacity   : 300
SubnetIds         : {subnet-fa1ae23c}
Tags              : {tag:Name}
VpcId             : vpc-12cf3b4f
```

```
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateFileSystemFromBackup](#) 中的。

Remove-FSXBackup

以下代码示例演示了如何使用 Remove-FSXBackup。

适用于 PowerShell V4 的工具

示例 1：此示例移除给定的 backup-id。

```
Remove-FSXBackup -BackupId $backupID
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXBackup (DeleteBackup)" on target
"backup-0bbca1e2345678e12".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y

BackupId                Lifecycle
-----
backup-0bbca1e2345678e12 DELETED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteBackup](#) 中的。

Remove-FSXFileSystem

以下代码示例演示了如何使用 Remove-FSXFileSystem。

适用于 PowerShell V4 的工具

示例 1：此示例移除给定的 FSX 文件系统 ID。

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXFileSystem (DeleteFileSystem)" on target
"fs-012ff34e567890120".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y

FileSystemId          Lifecycle WindowsResponse
-----
fs-012ff34e567890120 DELETING  Amazon.FSx.Model.DeleteFileSystemWindowsResponse
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFileSystem](#) 中的。

Remove-FSXResourceTag

以下代码示例演示了如何使用 Remove-FSXResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例移除给定 FSX 文件系统资源 ARN 的资源标签。

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXResourceTag (UntagResource)" on target
"arn:aws:fsx:eu-west-1:933303704102:file-system/fs-07cd45bc6bdf2674a".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagResource](#) 中的。

Update-FSXFileSystem

以下代码示例演示了如何使用 Update-FSXFileSystem。

适用于 PowerShell V4 的工具

示例 1：此示例通过 UpdateFileSystemWindowsConfiguration 以下方式更新 FSX 文件系统的自动备份保留天数。

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

输出：

```
CreationTime      : 1/17/2019 9:55:30 AM
DNSName           : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails    :
FileSystemId      : fs-01cd23bc4bdf5678a
FileSystemType    : WINDOWS
KmsKeyId          : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-
a1f2-e1234c5af678
Lifecycle         : AVAILABLE
LustreConfiguration :
NetworkInterfaceIds : {eni-01cd23bc4bdf5678a}
OwnerId           : 933303704102
ResourceARN       : arn:aws:fsx:eu-west-1:933303704102:file-system/
fs-07cd45bc6bdf2674a
StorageCapacity   : 300
SubnetIds         : {subnet-1d234567}
Tags              : {FSx-Service}
VpcId             : vpc-23cf4b5f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateFileSystem](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon Glacier 示例

以下代码示例向您展示了如何使用带有 Amazon Glacier 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-GLCJob

以下代码示例演示了如何使用 Get-GLCJob。

适用于 PowerShell V4 的工具

示例 1：返回指定任务的详细信息。任务成功完成后，可以使用 Read-Object GCJob output cmdlet 将任务的内容（档案或清单列表）检索到本地文件系统。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

输出：

```
Action                : ArchiveRetrieval
ArchiveId              : o909j...X-TpIhQJw
ArchiveSHA256TreeHash  : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes     : 38034480
Completed              : False
CompletionDate         : 1/1/0001 12:00:00 AM
CreationDate           : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes    : 0
JobDescription         :
JobId                  : op1x...JSbthM
JobOutputPath          :
OutputLocation         :
RetrievalByteRange     : 0-38034479
SelectParameters       :
SHA256TreeHash         : 79f3ea754c02f58...dc57bf4395b
SNSTopic               :
StatusCode             : InProgress
StatusMessage          :
Tier                   : Standard
```

```
VaultARN : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeJob](#) 中的。

New-GLCVault

以下代码示例演示了如何使用 New-GLCVault。

适用于 PowerShell V4 的工具

示例 1：为用户账户创建新文件库。由于未向 -AccountId 参数提供任何值，因此 cmdlet 使用默认值“-”来表示当前账户。

```
New-GLCVault -VaultName myvault
```

输出：

```
/01234567812/vaults/myvault
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateVault](#) 中的。

Read-GLCJobOutput

以下代码示例演示了如何使用 Read-GLCJobOutput。

适用于 PowerShell V4 的工具

示例 1：下载计划在指定任务中检索的档案内容，并将这些内容存储到磁盘上的文件中。如果有校验和，则下载会为您验证校验和。如果需要，可以通过指定 **-Select '*'** 来返回包括校验和在内的整个响应。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp\blue.bin"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetJobOutput](#) 中的。

Start-GLCJob

以下代码示例演示了如何使用 Start-GLCJob。

适用于 PowerShell V4 的工具

示例 1：启动一项任务，从用户拥有的指定文件库中检索档案。可以使用 Get-GLCJob cmdlet 检查作业的状态。任务成功完成后，可以使用读取-GCJob 输出 cmdlet 将存档内容检索到本地文件系统。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
"archive retrieval" -ArchiveId "o909j...TX-TpIhQJw"
```

输出：

JobId	JobOutputPath	Location
-----	-----	-----
op1x...JSbthM		/012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [InitiateJob](#) 中的。

Write-GLCArchive

以下代码示例演示了如何使用 Write-GLCArchive。

适用于 PowerShell V4 的工具

示例 1：将单个文件上传到指定文件库，返回档案 ID 和计算出的校验和。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

输出：

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b

示例 2：将文件夹层次结构的内容上传到用户账户中的指定文件库。对于上传的每个文件，cmdlet 都会发出文件名、相应的档案 ID 和计算出的档案校验和。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

输出：

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b
C:\temp\green.bin	qXAf0dSG...czo729UHXrw	d50a1...9184b9
C:\temp\lum.bin	39aNifP3...q9nb8nZkFIg	28886...5c3e27
C:\temp\red.bin	vp7E6rU_...Ejk_HhjAxKA	e05f7...4e34f5
C:\temp\Folder1\file1.txt	_eRINlip...5Sxy7dD2BaA	d0d2a...c8a3ba
C:\temp\Folder2\file2.iso	-Ix3jlmU...iXiDh-Xf0PA	7469e...3e86f1

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UploadArchive](#) 中的。

AWS Glue 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS Glue 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

New-GLUEJob

以下代码示例演示了如何使用 New-GLUEJob。

适用于 PowerShell V4 的工具

示例 1：此示例在 AWS Glue 中创建了一个新作业。命令名称的值始终为 **glueet1**。AWS Glue 支持运行用 Python 或 Scala 编写的作业脚本。在此示例中，作业脚本 (MyTestGlueJob.py) 是用 Python 编写的。Python 参数在 **\$DefArgs** 变量中指定，然后传递给 **DefaultArguments** 参数中的 PowerShell 命令，参数接受哈希表。**\$JobParams** 变量中的参数来自 CreateJob API，记录在 Glue

API 参考的作业 (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html>) 主题中 AWS。

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
    '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
    '--job-bookmark-option' = 'job-bookmark-disable'
    '--job-language' = 'python'
}
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"     = $DefArgs
    "Description"          = "This is a test"
    "ExecutionProperty"    = $ExecutionProp
    "MaxRetries"           = "1"
    "Name"                 = "MyOregonTestGlueJob"
    "Role"                 = "Amazon-GlueServiceRoleForSSM"
    "Timeout"              = "20"
}

New-GlueJob @JobParams
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateJob](#) 中的。

AWS Health 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS Health 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-HLTHEvent

以下代码示例演示了如何使用 Get-HLTHEvent。

适用于 PowerShell V4 的工具

示例 1：此命令从 Person AWS al Health Dashboard 返回事件。用户添加 -Region 参数以查看美国东部（弗吉尼亚州北部）区域中可用于服务的事件，而 -Filter_Region 参数筛选在欧洲地区（伦敦）和美国西部（俄勒冈州）区域（eu-west-2 和 us-west-2）中记录的事件。-Filter_StartTime 参数筛选事件可以开始的时间范围，而 -Filter_EndTime 参数则筛选事件可能结束的时间范围。结果是 RDS 的计划维护事件，该事件在指定的 -Filter_StartTime 范围内开始，并在计划的 -Filter_ 范围内结束。EndTime

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -  
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -  
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

输出：

```
Arn                : arn:aws:health:us-west-2::event/RDS/  
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/  
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321  
AvailabilityZone   :  
EndTime            : 3/21/2019 2:00:00 PM
```

```
EventTypeCategory : scheduledChange
EventTypeCode      : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime    : 2/28/2019 2:26:07 PM
Region             : us-west-2
Service            : RDS
StartTime           : 3/14/2019 2:00:00 PM
StatusCode          : open
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeEvents](#) 中的。

使用适用于 PowerShell V4 的工具的 IAM 示例

以下代码示例向您展示了如何使用带有 IAM 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-IAMClientIDToOpenIDConnectProvider

以下代码示例演示了如何使用 Add-IAMClientIDToOpenIDConnectProvider。

适用于 PowerShell V4 的工具

示例 1：此命令将客户端 ID (或受众) **my-application-ID** 添加到名为 **server.example.com** 的现有 OIDC 提供者。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
-OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddClientIdToOpenIdConnectProvider](#) 中的。

Add-IAMRoleTag

以下代码示例演示了如何使用 Add-IAMRoleTag。

适用于 PowerShell V4 的工具

示例 1：此示例在身份管理服务中为角色添加标签

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagRole](#) 中的。

Add-IAMRoleToInstanceProfile

以下代码示例演示了如何使用 Add-IAMRoleToInstanceProfile。

适用于 PowerShell V4 的工具

示例 1：此命令将名为 **S3Access** 的角色添加到名为 **webserver** 的现有实例配置文件中。要创建实例配置文件，请使用 **New-IAMInstanceProfile** 命令。使用此命令创建实例配置文件并将其与角色关联后，您可以将其附加到 EC2 实例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile-Name** 参数的 **New-EC2Instance** cmdlet 来启动新实例。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddRoleToInstanceProfile](#) 中的。

Add-IAMUserTag

以下代码示例演示了如何使用 Add-IAMUserTag。

适用于 PowerShell V4 的工具

示例 1：此示例在身份管理服务中为用户添加标签

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagUser](#) 中的。

Add-IAMUserToGroup

以下代码示例演示了如何使用 Add-IAMUserToGroup。

适用于 PowerShell V4 的工具

示例 1：此命令将名为 **Bob** 的用户添加到名为 **Admins** 的组中。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddUserToGroup](#) 中的。

Disable-IAMMFADevice

以下代码示例演示了如何使用 Disable-IAMMFADevice。

适用于 PowerShell V4 的工具

示例 1：此命令禁用与具有序列号 **123456789012** 的用户 **Bob** 关联的硬件 MFA 设备。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

示例 2：此命令禁用与具有 ARN **arn:aws:iam::210987654321:mfa/David** 的用户 **David** 关联的虚拟 MFA 设备。请注意，虚拟 MFA 设备不会从账户中删除。虚拟设备仍然存在并出现在 **Get-IAMVirtualMFADevice** 命令的输出中。您必须先使用 **Remove-IAMVirtualMFADevice** 命令删除旧的虚拟 MFA 设备，然后才能为同一用户创建新的虚拟 MFA 设备。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/David"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeactivateMfaDevice](#) 中的。

Edit-IAMPassword

以下代码示例演示了如何使用 Edit-IAMPassword。

适用于 PowerShell V4 的工具

示例 1：此命令更改运行该命令的用户的密码。此命令只能由 IAM 用户调用。如果您在使用 AWS 账户（根）凭据登录时调用此命令，则该命令将返回错误。**InvalidUserType**

```
Edit-IAMPASSWORD -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ChangePassword](#) 中的。

Enable-IAMMFADevice

以下代码示例演示了如何使用 Enable-IAMMFADevice。

适用于 PowerShell V4 的工具

示例 1：此命令启用序列号为 **987654321098** 的硬件 MFA 设备，并将该设备与用户 **Bob** 关联。它包含设备中按顺序排列的前两个代码。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -  
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

示例 2：此示例创建并启用虚拟 MFA 设备。第一条命令创建虚拟设备并在变量 **\$MFADevice** 中返回设备的对象表示形式。您可以使用 **.Base32StringSeed** 或 **QRCodePng** 属性来配置用户的软件应用程序。最后一条命令将该设备分配给用户 **David**，通过其序列号识别设备。该命令还 AWS 通过按顺序包含虚拟 MFA 设备中的前两个代码来与设备同步。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"  
# see example for New-IAMVirtualMFADevice to see how to configure the software  
# program with PNG or base32 seed code  
Enable-IAMMFADevice -UserName "David" -SerialNumber $MFADevice.SerialNumber  
-AuthenticationCode1 "24681357" -AuthenticationCode2  
"13572468"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [EnableMfaDevice](#) 中的。

Get-IAMAccessKey

以下代码示例演示了如何使用 Get-IAMAccessKey。

适用于 PowerShell V4 的工具

示例 1：此命令列出名为 **Bob** 的 IAM 用户的访问密钥。请注意，您无法列出 IAM 用户的秘密访问密钥。如果秘密访问密钥丢失，则必须使用 **New-IAMAccessKey** cmdlet 创建新的访问密钥。

```
Get-IAMAccessKey -UserName "Bob"
```

输出：

AccessKeyId	CreateDate	Status	UserName
-----	-----	-----	-----
AKIAIOSFODNN7EXAMPLE	12/3/2014 10:53:41 AM	Active	Bob
AKIAI44QH8DHBEXAMPLE	6/6/2013 8:42:26 PM	Inactive	Bob

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListAccessKeys](#) 中的。

Get-IAMAccessKeyLastUsed

以下代码示例演示了如何使用 Get-IAMAccessKeyLastUsed。

适用于 PowerShell V4 的工具

示例 1：返回所提供访问密钥的拥有用户名和上次使用信息。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetAccessKeyLastUsed](#) 中的。

Get-IAMAccountAlias

以下代码示例演示了如何使用 Get-IAMAccountAlias。

适用于 PowerShell V4 的工具

示例 1：此命令返回 AWS 账户的账户别名。

```
Get-IAMAccountAlias
```

输出：

```
ExampleCo
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListAccountAliases](#) 中的。

Get-IAMAccountAuthorizationDetail

以下代码示例演示了如何使用 Get-IAMAccountAuthorizationDetail。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关 AWS 账户中身份的授权详细信息，并显示返回对象的元素列表，包括用户、群组 and 角色。例如，**UserDetailList** 属性显示有关用户的详细信息。**RoleDetailList** 和 **GroupDetailList** 属性中也提供类似的信息。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

输出：

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

输出：

```
Arn          : arn:aws:iam::123456789012:user/Administrator
CreateDate   : 10/16/2014 9:03:09 AM
GroupList    : {Administrators}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE1
UserName     : Administrator
UserPolicyList : {}
```

```

Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/6/2015 12:54:42 PM
GroupList    : {Developers}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE2
UserName     : bab
UserPolicyList : {}

Arn          : arn:aws:iam::123456789012:user/BackupToS3
CreateDate   : 1/27/2015 10:15:08 AM
GroupList    : {Backup}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE3
UserName     : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAccountAuthorizationDetails](#) 中的。

Get-IAMAccountPasswordPolicy

以下代码示例演示了如何使用 Get-IAMAccountPasswordPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关当前账户密码策略的详细信息。如果没有为账户定义密码策略，则命令将返回 **NoSuchEntity** 错误。

```
Get-IAMAccountPasswordPolicy
```

输出：

```

AllowUsersToChangePassword : True
ExpirePasswords             : True
HardExpiry                  : False
MaxPasswordAge              : 90
MinimumPasswordLength       : 8
PasswordReusePrevention     : 20
RequireLowercaseCharacters  : True
RequireNumbers              : True
RequireSymbols               : False
RequireUppercaseCharacters  : True

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAccountPasswordPolicy](#) 中的。

Get-IAMAccountSummary

以下代码示例演示了如何使用 Get-IAMAccountSummary。

适用于 PowerShell V4 的工具

示例 1：此示例将返回有关 AWS 账户中当前 IAM 实体使用情况和当前 IAM 实体配额的信息。

```
Get-IAMAccountSummary
```

输出：

Key	Value
Users	7
GroupPolicySizeQuota	5120
PolicyVersionsInUseQuota	10000
ServerCertificatesQuota	20
AccountSigningCertificatesPresent	0
AccountAccessKeysPresent	0
Groups	3
UsersQuota	5000
RolePolicySizeQuota	10240
UserPolicySizeQuota	2048
GroupsPerUserQuota	10
AssumeRolePolicySizeQuota	2048
AttachedPoliciesPerGroupQuota	2
Roles	9
VersionsPerPolicyQuota	5
GroupsQuota	100
PolicySizeQuota	5120
Policies	5
RolesQuota	250
ServerCertificates	0
AttachedPoliciesPerRoleQuota	2
MFADevicesInUse	2
PoliciesQuota	1000
AccountMFAEnabled	1
Providers	2

InstanceProfilesQuota	100
MFADevices	4
AccessKeysPerUserQuota	2
AttachedPoliciesPerUserQuota	2
SigningCertificatesPerUserQuota	2
PolicyVersionsInUse	4
InstanceProfiles	1
...	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAccountSummary](#) 中的。

Get-IAMAttachedGroupPolicyList

以下代码示例演示了如何使用 Get-IAMAttachedGroupPolicyList。

适用于 PowerShell V4 的工具

示例 1：此命令返回挂载到 AWS 账户中名为 IAM 组的托管策略 **Admins** 的名称和 ARNs 托管策略。要查看组中嵌入的内联策略列表，请使用 **Get-IAMGroupPolicyList** 命令。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

输出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess	AdministratorAccess

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListAttachedGroupPolicies](#) 中的。

Get-IAMAttachedRolePolicyList

以下代码示例演示了如何使用 Get-IAMAttachedRolePolicyList。

适用于 PowerShell V4 的工具

示例 1：此命令返回挂载到 AWS 账户中名为 IAM 角色的托管策略 **SecurityAuditRole** 的名称和 ARNs 托管策略。要查看嵌入在角色中的内联策略的列表，请使用 **Get-IAMRolePolicyList** 命令。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

输出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListAttachedRolePolicies](#) 中的。

Get-IAMAttachedUserPolicyList

以下代码示例演示了如何使用 Get-IAMAttachedUserPolicyList。

适用于 PowerShell V4 的工具

示例 1：此命令返回 AWS 账户 **Bob** 中名为 IAM 用户的名称和 ARNs 托管策略。要查看嵌入在 IAM 用户中的内联策略的列表，请使用 **Get-IAMUserPolicyList** 命令。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

输出：

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/TesterPolicy	TesterPolicy

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListAttachedUserPolicies](#) 中的。

Get-IAMContextKeysForCustomPolicy

以下代码示例演示了如何使用 Get-IAMContextKeysForCustomPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例获取所提供策略 json 中存在的所有上下文键。为提供多个策略，您可以用逗号分隔值列表提供。

```
$policy1 = '{"Version":"2012-10-17",          "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",          "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"},"}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetContextKeysForCustomPolicy`](#) 中的。

Get-IAMContextKeysForPrincipalPolicy

以下代码示例演示了如何使用 `Get-IAMContextKeysForPrincipalPolicy`。

适用于 PowerShell V4 的工具

示例 1：此示例获取所提供的策略 json 中和附加到 IAM 实体（用户/角色等）的策略中存在的所有上下文键。For-PolicyInputList 您可以将多个值列表作为逗号分隔的值提供。

```
$policy1 = '{"Version":"2012-10-17",          "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",          "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"},"}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetContextKeysForPrincipalPolicy`](#) 中的。

Get-IAMCredentialReport

以下代码示例演示了如何使用 Get-IAMCredentialReport。

适用于 PowerShell V4 的工具

示例 1：此示例打开返回的报告，并将其作为文本行数组输出到管道。第一行是标题，含有用逗号分隔的列名。接下来的每一行都是一个用户的详细信息行，其中每个字段用逗号分隔。必须先使用 **Request-IAMCredentialReport** cmdlet 生成报告，然后才能查看报告。要将报告作为单个字符串检索，请使用 **-Raw** 而不是 **-AsTextArray**。**-AsTextArray** 开关也接受别名 **-SplitLines**。有关输出中列的完整列表，请参阅服务 API 参考。请注意，如果不使用 **-AsTextArray** 或 **-SplitLines**，则必须使用 .NET **StreamReader** 类从 **.Content** 属性中提取文本。

```
Request-IAMCredentialReport
```

输出：

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

```
Get-IAMCredentialReport -AsTextArray
```

输出：

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,password_last_changed,password_last_used,password_last_changed,password_last_used,root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04-15T16:31:25+00:00,A,false,N/A,false,N/A,false,N/AAdministrator,arn:aws:iam::123456789012:user/Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+00:00,A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/A,A,false,N/ABill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/A,A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetCredentialReport](#) 中的。

Get-IAMEntitiesForPolicy

以下代码示例演示了如何使用 Get-IAMEntitiesForPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例返回附加了策略 **arn:aws:iam::123456789012:policy/TestPolicy** 的 IAM 组、角色和用户的列表。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

输出：

```
IsTruncated   : False
Marker        :
PolicyGroups  : {}
PolicyRoles   : {testRole}
PolicyUsers   : {Bob, Theresa}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListEntitiesForPolicy](#) 中的。

Get-IAMGroup

以下代码示例演示了如何使用 Get-IAMGroup。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关 IAM 组 **Testers** 的详细信息，包括属于该组的所有 IAM 用户的集合。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

输出：

Group	IsTruncated	Marker
Users		
-----	-----	-----

Amazon.IdentityManagement.Model.Group	False	
{Theresa, David}		

```
$results.Group
```

输出：

```
Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName : Testers
Path      : /
```

```
$results.Users
```

输出：

```
Arn      : arn:aws:iam::123456789012:user/Theresa
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path      : /
UserId    : 40SVDDJJTF4XEEXAMPLE2
UserName  : Theresa

Arn      : arn:aws:iam::123456789012:user/David
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path      : /
UserId    : Y4FKWQCXTA52QEXAMPLE3
UserName  : David
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetGroup](#) 中的。

Get-IAMGroupForUser

以下代码示例演示了如何使用 Get-IAMGroupForUser。

适用于 PowerShell V4 的工具

示例 1：此示例返回 IAM 用户 **David** 所属的 IAM 组列表。

```
Get-IAMGroupForUser -UserName David
```

输出：

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
GroupName : Administrators
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE2
GroupName : Testers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZ0EXAMPLE3
GroupName : Developers
Path      : /
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListGroupForUser](#) 中的。

Get-IAMGroupList

以下代码示例演示了如何使用 Get-IAMGroupList。

适用于 PowerShell V4 的工具

示例 1：此示例返回当前中定义的所有 IAM 组的集合 AWS 账户。

```
Get-IAMGroupList
```

输出：

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEEXAMPLE1
GroupName : Administrators
Path      : /
```

```

Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZ0EXAMPLE2
GroupName : Developers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE3
GroupName : Testers
Path      : /

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListGroupsWithPolicy](#) 中的。

Get-IAMGroupPolicy

以下代码示例演示了如何使用 Get-IAMGroupPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关组 **Testers** 的名为 **PowerUserAccess-Testers** 的嵌入式内联策略的详细信息。**PolicyDocument** 属性采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

输出：

```

GroupName      PolicyDocument                                     PolicyName
-----
Testers        %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [

```

```
        "ec2:DescribeInstances"
    ],
    "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
    ]
}
]
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetGroupPolicy](#) 中的。

Get-IAMGroupPolicyList

以下代码示例演示了如何使用 Get-IAMGroupPolicyList。

适用于 PowerShell V4 的工具

示例 1：返回嵌入组 **Testers** 中的内联策略的列表。要获取附加到该组的托管策略，请使用命令 **Get-IAMAttachedGroupPolicyList**。

```
Get-IAMGroupPolicyList -GroupName Testers
```

输出：

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListGroupPolicies](#) 中的。

Get-IAMInstanceProfile

以下代码示例演示了如何使用 Get-IAMInstanceProfile。

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 AWS 账户中定义 **ec2instancerole** 的名为的实例配置文件的详细信息。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

输出：

```
Arn           : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles         : {ec2instancerole}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetInstanceProfile](#) 中的。

Get-IAMInstanceProfileForRole

以下代码示例演示了如何使用 `Get-IAMInstanceProfileForRole`。

适用于 PowerShell V4 的工具

示例 1：此示例返回与角色 **ec2instancerole** 关联的实例配置文件的详细信息。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

输出：

```
Arn           : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles         : {ec2instancerole}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListInstanceProfilesForRole](#) 中的。

Get-IAMInstanceProfileList

以下代码示例演示了如何使用 `Get-IAMInstanceProfileList`。

适用于 PowerShell V4 的工具

示例 1：此示例返回当前中定义的实例配置文件的集合 AWS 账户。

```
Get-IAMInstanceProfileList
```

输出：

```
Arn           : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path          : /
Roles         : {ec2instancerole}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListInstanceProfiles](#)中的。

Get-IAMLoginProfile

以下代码示例演示了如何使用 Get-IAMLoginProfile。

适用于 PowerShell V4 的工具

示例 1：此示例返回密码创建日期，以及 IAM 用户 **David** 是否需要重置密码。

```
Get-IAMLoginProfile -UserName David
```

输出：

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
12/10/2014 3:39:44 PM	False	David

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetLoginProfile](#)中的。

Get-IAMMFADevice

以下代码示例演示了如何使用 Get-IAMMFADevice。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关分配给 IAM 用户 **David** 的 MFA 设备的详细信息。在此示例中，您可以看出它是虚拟设备，因为 **SerialNumber** 是 ARN，而不是物理设备的实际序列号。

```
Get-IAMMFADevice -UserName David
```

输出：

EnableDate	SerialNumber	UserName
-----	-----	-----
4/8/2015 9:41:10 AM	arn:aws:iam::123456789012:mfa/David	David

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListMfaDevices](#) 中的。

Get-IAMOpenIDConnectProvider

以下代码示例演示了如何使用 Get-IAMOpenIDConnectProvider。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关 ARN 为 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** 的 OpenID Connect 提供者的详细信息。该**ClientIDList**属性是一个集合，其中包含为此提供者 IDs 定义的所有客户端。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

输出：

ClientIDList Url	CreateDate	ThumbprintList
-----	-----	-----

{MyOIDCApp} {12345abcdefghijk67890lmnopqrst98765uvwxy}	2/3/2015 3:00:30 PM	oidc.example.com

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetOpenIdConnectProvider](#) 中的。

Get-IAMOpenIDConnectProviderList

以下代码示例演示了如何使用 Get-IAMOpenIDConnectProviderList。

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 AWS 账户中定义的所有 OpenID Connect 提供者的 ARN 列表。

```
Get-IAMOpenIDConnectProviderList
```

输出：

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListOpenIdConnectProviders](#) 中的。

Get-IAMPolicy

以下代码示例演示了如何使用 Get-IAMPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的托管策略的详细信息。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
Arn           : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount : 0
CreateDate     : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description    :
IsAttachable   : True
Path           : /
PolicyId       : Z27SI6FQMGNQ2EXAMPLE1
PolicyName     : MySamplePolicy
```

```
UpdateDate      : 2/6/2015 10:40:08 AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPolicy](#) 中的。

Get-IAMPolicyList

以下代码示例演示了如何使用 Get-IAMPolicyList。

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 AWS 账户中可用的前三个托管策略的集合。由于 **-scope** 未指定，因此它默认为 **all** 并包括 AWS 托管策略和客户托管策略。

```
Get-IAMPolicyList -MaxItem 3
```

输出：

```
Arn          : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount : 0
CreateDate    : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description   :
IsAttachable  : True
Path         : /
PolicyId      : Z27SI6FQMGNQ2EXAMPLE1
PolicyName    : AWSDirectConnectReadOnlyAccess
UpdateDate    : 2/6/2015 10:40:08 AM

Arn          : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount : 0
CreateDate    : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description   :
IsAttachable  : True
Path         : /
PolicyId      : NJKMU274MET4EEXAMPLE2
PolicyName    : AmazonGlacierReadOnlyAccess
UpdateDate    : 2/6/2015 10:40:27 AM

Arn          : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount : 0
CreateDate    : 2/11/2015 9:21:45 AM
```

```
DefaultVersionId : v1
Description      :
IsAttachable    : True
Path            : /
PolicyId        : 5ULJS02FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate      : 2/11/2015 9:21:45 AM
```

示例 2：此示例返回当前 AWS 账户中可用的前两个客户托管保单的集合。其使用 **-Scope local** 将输出限制为仅限客户管理型策略。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

输出：

```
Arn          : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount : 0
CreateDate    : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description   :
IsAttachable  : True
Path         : /
PolicyId     : SQVCBLC4VA0UCEXAMPLE4
PolicyName   : MyLocalPolicy
UpdateDate   : 2/12/2015 9:39:53 AM

Arn          : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount : 1
CreateDate    : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description   :
IsAttachable  : True
Path         : /
PolicyId     : X5JPBLJH2Z2S0EXAMPLE5
PolicyName   : policyforec2instancerole
UpdateDate   : 2/18/2015 8:52:31 AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListPolicies](#) 中的。

Get-IAMPolicyVersion

以下代码示例演示了如何使用 Get-IAMPolicyVersion。

适用于 PowerShell V4 的工具

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略 **v2** 版本的策略文档。**Document** 属性中的策略文档采用 URL 编码，在本示例中使用 **UrlDecode** .NET 方法进行解码。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy -VersionId v2
$results
```

输出：

```
CreateDate          Document
-----
IsDefaultVersion    VersionId
-----
-----
2/12/2015 9:39:53 AM %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeInstances"
    ],
    "Resource": [
      "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
    ]
  }
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPolicyVersion](#) 中的。

Get-IAMPolicyVersionList

以下代码示例演示了如何使用 Get-IAMPolicyVersionList。

适用于 PowerShell V4 的工具

示例 1：此示例返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略的可用版本列表。要获取特定版本的策略文档，请使用 **Get-IAMPolicyVersion** 命令并指定所需版本的 **VersionId**。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

输出：

CreateDate VersionId	Document	IsDefaultVersion
----- -----	-----	-----
2/12/2015 9:39:53 AM v2		True
2/12/2015 9:39:09 AM v1		False

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListPolicyVersions](#) 中的。

Get-IAMRole

以下代码示例演示了如何使用 Get-IAMRole。

适用于 PowerShell V4 的工具

示例 1：此示例返回 **lambda_exec_role** 的详细信息。其包括指定谁可以担任此角色的信任策略文档。策略文档采用 URL 编码，可使用 .NET **UrlDecode** 方法进行解码。在此示例中，原始策略在上传到策略之前已删除所有空格。要查看确定承担该角色的人员可以执行哪些操作的权限策略文档，对内联策略使用 **Get-IAMRolePolicy**，对附加的托管策略使用 **Get-IAMPolicyVersion**。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

输出：

Arn	: arn:aws:iam::123456789012:role/lambda_exec_role

```
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22Service%22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%7D%5D%7D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetRole](#) 中的。

Get-IAMRoleList

以下代码示例演示了如何使用 Get-IAMRoleList。

适用于 PowerShell V4 的工具

示例 1：此示例检索 AWS 账户中的所有 IAM 角色列表。

```
Get-IAMRoleList
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListRoles](#) 中的。

Get-IAMRolePolicy

以下代码示例演示了如何使用 Get-IAMRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例返回嵌入在 IAM 角色 **lambda_exec_role** 中的名为 **oneClick_lambda_exec_role_policy** 的策略的权限策略文档。生成的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
oneClick_lambda_exec_role_policy
$results
```

输出：

PolicyDocument	PolicyName
----- ----- %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%... oneClick_lambda_exec_role_policy lambda_exec_role	-----

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

输出：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
    }
  ]
}
```

```
        "Resource": [  
            "arn:aws:s3:::amzn-s3-demo-bucket/*"  
        ]  
    }  
]  
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetRolePolicy](#) 中的。

Get-IAMRolePolicyList

以下代码示例演示了如何使用 Get-IAMRolePolicyList。

适用于 PowerShell V4 的工具

示例 1：此示例返回 IAM 角色 **lamda_exec_role** 中嵌入的内联策略名称列表。要查看内联策略的详细信息，请使用 **Get-IAMRolePolicy** 命令。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

输出：

```
oneClick_lambda_exec_role_policy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListRolePolicies](#) 中的。

Get-IAMRoleTagList

以下代码示例演示了如何使用 Get-IAMRoleTagList。

适用于 PowerShell V4 的工具

示例 1：此示例获取与角色关联的标签。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListRoleTags](#) 中的。

Get-IAMSAMLProvider

以下代码示例演示了如何使用 Get-IAMSAMLProvider。

适用于 PowerShell V4 的工具

示例 1：此示例检索有关 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 SAML 2.0 提供者的详细信息。响应包括您从身份提供商那里获得的用于创建 AWS SAML 提供商实体的元数据文档以及创建日期和到期日期。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

输出：

CreateDate	SAMLMetadataDocument
ValidUntil	
-----	-----

12/23/2014 12:16:55 PM	<EntityDescriptor ID="_12345678-1234-5678-9012-example1...
12/23/2114 12:16:54 PM	

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetSamlProvider](#) 中的。

Get-IAMSAMLProviderList

以下代码示例演示了如何使用 Get-IAMSAMLProviderList。

适用于 PowerShell V4 的工具

示例 1：此示例将检索在当前 AWS 账户中创建的 SAML 2.0 提供者列表。其返回每个 SAML 提供者的 ARN、创建日期和到期日期。

```
Get-IAMSAMLProviderList
```

输出：

Arn	CreateDate
ValidUntil	

```

---
-----
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
12/23/2114 12:16:54 PM

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考文档 \(V 4\)](#) SAMLProviders 中的 [列表](#)。

Get-IAMServerCertificate

以下代码示例演示了如何使用 Get-IAMServerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例检索有关名为 **MyServerCertificate** 的服务器证书的详细信息。您可以在 **CertificateBody** 和 **ServerCertificateMetadata** 属性中找到证书详细信息。

```

$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list

```

输出：

```

CertificateBody          : -----BEGIN CERTIFICATE-----

MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC

VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6

b24xFDASBgNVBAstC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAd

BgkqhkiG9w0BCQEWEG5vb25lQGFTtYXpvi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD

VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAstC01BTSBDb25z

b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFT

YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMak0dn

+a4GmWIWJ

21uUSfwfEvySwTc2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/

f0wYK8m9T

```

```

                                rDHudUZg3qX4waLG5M43q7Wgc/
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE

    Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                                nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

    FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
                                NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                                -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
    Amazon.IdentityManagement.Model.ServerCertificateMetadata

```

```
$result.ServerCertificateMetadata
```

输出：

```

Arn                : arn:aws:iam::123456789012:server-certificate/0rg1/0rg2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path              : /0rg1/0rg2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetServerCertificate](#) 中的。

Get-IAMServerCertificateList

以下代码示例演示了如何使用 `Get-IAMServerCertificateList`。

适用于 PowerShell V4 的工具

示例 1：此示例检索已上传到当前 AWS 账户的服务器证书列表。

```
Get-IAMServerCertificateList
```

输出：

```
Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path               : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListServerCertificates](#) 中的。

Get-IAMServiceLastAccessedDetail

以下代码示例演示了如何使用 Get-IAMServiceLastAccessedDetail。

适用于 PowerShell V4 的工具

示例 1：此示例提供了请求调用中关联的 IAM 实体（用户、组、角色或策略）上次访问的服务的详细信息。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

输出：

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetServiceLastAccessedDetails](#) 中的。

Get-IAMServiceLastAccessedDetailWithEntity

以下代码示例演示了如何使用 Get-IAMServiceLastAccessedDetailWithEntity。

适用于 PowerShell V4 的工具

示例 1：此示例提供了相应的 IAM 实体上次访问请求中服务的时间戳。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

输出：

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error              :
IsTruncated        : False
JobCompletionDate  : 12/29/19 11:19:31 AM
JobCreationDate    : 12/29/19 11:19:31 AM
JobStatus          : COMPLETED
Marker             :
```

```
$results.EntityDetailsList
```

输出：

```
EntityInfo                                LastAuthenticated
-----                                -
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

输出：

```
Arn   : arn:aws:iam::123456789012:user/TestUser
Id    : AIDA4NBK5CXF5TZHU1234
Name  : TestUser
Path  : /
Type  : USER
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetServiceLastAccessedDetailsWithEntities](#) 中的。

Get-IAMSigningCertificate

以下代码示例演示了如何使用 Get-IAMSigningCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例检索与名为 **Bob** 的用户关联的签名证书的相关详细信息。

```
Get-IAMSigningCertificate -UserName Bob
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----
                MIIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
                VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbW
                b24xFDASBgNVBAsTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnR1b3R1b3R1b3R1
                BgkqhkiG9w0BCQEWEG5vb25lQGFTtYXpvi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                MTIwNDI1MjA0NTIxWjCBiDELMakGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
                VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWb24xFDASBgNVBAsTC0lBTSBDb25z
                b2x1MRIwEAYDVQQDEw1UZXRhbnR1b3R1b3R1b3R1BgkqhkiG9w0BCQEWEG5vb25lQGFT
                YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                21uUSfwfEvySWtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
                Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
                FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
                NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate     : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListSigningCertificates](#) 中的。

Get-IAMUser

以下代码示例演示了如何使用 Get-IAMUser。

适用于 PowerShell V4 的工具

示例 1：此示例检索名为 **David** 的用户的详细信息。

```
Get-IAMUser -UserName David
```

输出：

```
Arn          : arn:aws:iam::123456789012:user/David
CreateDate   : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path         : /
UserId       : Y4FKWQCXTA52QEXAMPLE1
UserName     : David
```

示例 2：此示例检索有关当前登录的 IAM 用户的详细信息。

```
Get-IAMUser
```

输出：

```
Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path         : /
UserId       : 7K3GJEANSKZF2EXAMPLE2
UserName     : Bob
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetUser](#) 中的。

Get-IAMUserList

以下代码示例演示了如何使用 Get-IAMUserList。

适用于 PowerShell V4 的工具

示例 1：此示例检索当前 AWS 账户用户集合。

```
Get-IAMUserList
```

输出：

```
Arn          : arn:aws:iam::123456789012:user/Administrator
CreateDate   : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
```

```
Path           : /
UserId         : 7K3GJEANSKZF2EXAMPLE1
UserName       : Administrator

Arn           : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path           : /
UserId         : L3EWNNDOM3YUEXAMPLE2
UserName       : bab

Arn           : arn:aws:iam::123456789012:user/David
CreateDate     : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path           : /
UserId         : Y4FKWQCXTA52QEXAMPLE3
UserName       : David
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListUsers](#) 中的。

Get-IAMUserPolicy

以下代码示例演示了如何使用 Get-IAMUserPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中名为 **Davids_IAM_Admin_Policy** 的内联策略的详细信息。策略文档采用 URL 编码。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

输出：

```
PolicyDocument                                     PolicyName
-----
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...  Davids_IAM_Admin_Policy
David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
```



```
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetUserPolicy](#) 中的。

Get-IAMUserPolicyList

以下代码示例演示了如何使用 Get-IAMUserPolicyList。

适用于 PowerShell V4 的工具

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中的内联策略名称列表。

```
Get-IAMUserPolicyList -UserName David
```

输出：

```
Davids_IAM_Admin_Policy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListUserPolicies](#) 中的。

Get-IAMUserTagList

以下代码示例演示了如何使用 Get-IAMUserTagList。

适用于 PowerShell V4 的工具

示例 1：此示例获取与用户关联的标签。

```
Get-IAMUserTagList -UserName joe
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListUserTags](#) 中的。

Get-IAMVirtualMFADevice

以下代码示例演示了如何使用 Get-IAMVirtualMFADevice。

适用于 PowerShell V4 的工具

示例 1：此示例检索分配给账户中 AWS 用户的虚拟 MFA 设备集合。每个设备的 **User** 属性都是一个对象，其中包含设备分配到的 IAM 用户的详细信息。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

输出：

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListVirtualMfaDevices](#) 中的。

New-IAMAccessKey

以下代码示例演示了如何使用 New-IAMAccessKey。

适用于 PowerShell V4 的工具

示例 1：此示例创建新的访问密钥和秘密访问密钥对，并将其分配给用户 **David**。确保将 **AccessKeyId** 和 **SecretAccessKey** 值保存到文件中，因为这是您唯一可以获得 **SecretAccessKey** 的时间。您以后无法检索它。如果您丢失了秘密密钥，则必须创建一个新的访问密钥对。

```
New-IAMAccessKey -UserName David
```

输出：

```
AccessKeyId      : AKIAIOSFODNN7EXAMPLE
CreateDate       : 4/13/2015 1:00:42 PM
SecretAccessKey  : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateAccessKey](#) 中的。

New-IAMAccountAlias

以下代码示例演示了如何使用 New-IAMAccountAlias。

适用于 PowerShell V4 的工具

示例 1：此示例将您 AWS 账户的账户别名更改为 **mycompanyaws**。用户登录页面的地址更改为 <https://mycompanyaws.signin.aws.amazon.com/console>。使用您的账户 ID 号而不是别名的原始 URL (<https://<accountidnumber>.signin.aws.amazon.com/console>) 继续有效。但是，任何先前定义的基于别名的都将 URLs 停止工作。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateAccountAlias](#) 中的。

New-IAMGroup

以下代码示例演示了如何使用 New-IAMGroup。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为 **Developers** 的新 IAM 组。

```
New-IAMGroup -GroupName Developers
```

输出：

```
Arn          : arn:aws:iam::123456789012:group/Developers
CreateDate   : 4/14/2015 11:21:31 AM
GroupId      : QNEJ5PM4NFSQCEXAMPLE1
GroupName    : Developers
Path         : /
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateGroup](#) 中的。

New-IAMInstanceProfile

以下代码示例演示了如何使用 New-IAMInstanceProfile。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为 **ProfileForDevEC2Instance** 的新 IAM 实例配置文件。您必须单独运行 **Add-IAMRoleToInstanceProfile** 命令，以将实例配置文件与为该实例提供权限的现有 IAM 角色相关联。最后，在启动实例时将实例配置文件附加到该 EC2 实例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile_Name** 参数的 **New-EC2Instance** cmdlet。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

输出：

```
Arn          : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate    : 4/14/2015 11:31:39 AM
InstanceId    : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path         : /
Roles        : {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateInstanceProfile](#) 中的。

New-IAMLoginProfile

以下代码示例演示了如何使用 New-IAMLoginProfile。

适用于 PowerShell V4 的工具

示例 1：此示例为名为 Bob 的 IAM 用户创建一个（临时）密码，并设置在 **Bob** 下次登录时要求该用户更改密码的标志。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

输出：

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
4/14/2015 12:26:30 PM	True	Bob

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateLoginProfile](#) 中的。

New-IAMOpenIDConnectProvider

以下代码示例演示了如何使用 New-IAMOpenIDConnectProvider。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个 IAM OIDC 提供者，该提供者与位于 URL **https://example.oidcprovider.com** 中的 OIDC 兼容提供者服务和客户端 ID **my-testapp-1** 关联。OIDC 提供者将提供指纹。要对指纹进行身份验证，请按照 <http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html> 上的步骤进行操作。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

输出：

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateOpenIdConnectProvider](#) 中的。

New-IAMPolicy

以下代码示例演示了如何使用 New-IAMPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例在当前 AWS 账户中创建了一个名为的新 IAM 策略。**MySamplePolicy**该文件**MySamplePolicy.json**提供了策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
MySamplePolicy.json)
```

输出：

```
Arn                : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount    : 0
CreateDate         : 4/14/2015 2:45:59 PM
DefaultVersionId   : v1
Description        :
IsAttachable       : True
Path               : /
PolicyId           : LD4KP6HVFE7WGEXAMPLE1
PolicyName         : MySamplePolicy
UpdateDate         : 4/14/2015 2:45:59 PM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreatePolicy](#)中的。

New-IAMPolicyVersion

以下代码示例演示了如何使用 New-IAMPolicyVersion。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个新“v2”版本的 IAM 策略，其 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy**，并设为默认版本。 **NewPolicyVersion.json** 文件提供了策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

输出：

CreateDate VersionId	Document	IsDefaultVersion
----- -----	-----	-----
4/15/2015 10:54:54 AM v2		True

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreatePolicyVersion](#) 中的。

New-IAMRole

以下代码示例演示了如何使用 New-IAMRole。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **MyNewRole** 的新角色，并将文件 **NewRoleTrustPolicy.json** 中的策略附加到该角色。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。输出中显示的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

输出：

```
Arn : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D%0A%20%20%22Statement%22%3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C%0D%0A%20%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C%0D%0A%20%20%20%20%20%20%20%22Principal%22%3A%20%7B%0D%0A%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws%3Aiam%3A%3A123456789012%3ADavid%22%0D%0A%20%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20%20%20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A%20%20%20%20%7D%0D%0A%20%20%5D%0D%0A%7D
```

```

CreateDate           : 4/15/2015 11:04:23 AM
Path                 : /
RoleId               : V5PAJI2KPN4EAEXAMPLE1
RoleName             : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateRole](#) 中的。

New-IAMSAMLProvider

以下代码示例演示了如何使用 New-IAMSAMLProvider。

适用于 PowerShell V4 的工具

示例 1：此示例在 IAM 中创建了一个新 SAML 提供者实体。其命名为 **MySAMLProvider**，通过文件 **SAMLMetaData.xml** 中的 SAML 元数据文档描述，该文件从 SAML 服务提供者网站单独下载。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

输出：

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- 有关 API 的详细信息，请参阅 SAMLProvider 在 AWS Tools for PowerShell Cmdlet 参考中 [创建](#) (V4)。

New-IAMServiceLinkedRole

以下代码示例演示了如何使用 New-IAMServiceLinkedRole。

适用于 PowerShell V4 的工具

示例 1：此示例为自动扩缩服务创建 servicelinked 角色。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix  
RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateServiceLinkedRole](#) 中的。

New-IAMUser

以下代码示例演示了如何使用 New-IAMUser。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **Bob** 的 IAM 用户。如果 Bob 需要登录 AWS 控制台，则必须单独运行命令 **New-IAMLoginProfile** 来创建带有密码的登录配置文件。如果 Bob 需要运行 AWS PowerShell 或跨平台 CLI 命令或 AWS 进行 API 调用，则必须单独运行该 **New-IAMAccessKey** 命令来创建访问密钥。

```
New-IAMUser -UserName Bob
```

输出：

```
Arn           : arn:aws:iam::123456789012:user/Bob  
CreateDate    : 4/22/2015 12:02:11 PM  
PasswordLastUsed : 1/1/0001 12:00:00 AM  
Path          : /  
UserId        : AIDAJWGEFDMEMEXAMPLE1  
UserName      : Bob
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateUser](#) 中的。

New-IAMVirtualMFADevice

以下代码示例演示了如何使用 New-IAMVirtualMFADevice。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个新的虚拟 MFA 设备。第 2 行和第 3 行提取虚拟 MFA 软件程序创建账户所需的 **Base32StringSeed** 值（作为二维码的替代方案）。使用该值配置程序后，从程序中获取两个连续的身份验证码。最后，使用最后一条命令将虚拟 MFA 设备关联到 IAM 用户 **Bob**，并将账户与两个身份验证码同步。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

输出：

```
-- Pause here to enter base-32 string seed code into virtual MFA program to register
account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

示例 2：此示例创建一个新虚拟 MFA 设备。第 2 行和第 3 行提取 **QRCodePNG** 值并将其写入文件。虚拟的 MFA 软件程序可以扫描此图像以创建账户（作为手动输入 Base32 StringSeed 值的替代方法）。在虚拟 MFA 程序中创建账户后，获取两个连续的身份验证码，然后在最后一条命令中输入验证码，将虚拟 MFA 设备关联到 IAM 用户 **Bob** 并同步账户。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path QRCode.png
```

输出：

```
-- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateVirtualMfaDevice](#) 中的。

Publish-IAMServerCertificate

以下代码示例演示了如何使用 Publish-IAMServerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例将新服务器证书上传到 IAM 账户。包含证书正文、私有密钥和（可选）证书链的文件必须均采用 PEM 编码。请注意，参数需要文件的实际内容，而不是文件名。必须使用 **-Raw** 开关参数才能成功处理文件内容。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody  
(Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

输出：

```
Arn                : arn:aws:iam::123456789012:server-certificate/MyTestCert  
Expiration         : 1/14/2018 9:52:36 AM  
Path               : /  
ServerCertificateId : ASCAJIEXAMPLE7J7HQZYW  
ServerCertificateName : MyTestCert  
UploadDate         : 4/21/2015 11:14:16 AM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UploadServerCertificate](#) 中的。

Publish-IAMSigningCertificate

以下代码示例演示了如何使用 Publish-IAMSigningCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例上传新的 X.509 签名证书并将其与名为 **Bob** 的 IAM 用户关联。包含证书正文的文件采用 PEM 编码。**CertificateBody** 参数需要证书文件的实际内容而不是文件名。必须使用 **-Raw** 开关参数才能成功处理该文件。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw  
SampleSigningCert.pem)
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----
```

```

MIICiTCCAFICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxH3Ad
BgkqhkiG9w0BCQEWEG5vb251QGFtYXpvi5jb20wHhCNMTEwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWMxH3AdBgkqhkiG9w0BCQEWEG5vb251QGFt
YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMak0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLygVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHVvYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

```

```

CertificateId      : Y3EK7RMEXAMPLESV33FCEXAMPLEHMLU
Status             : Active
UploadDate        : 4/20/2015 1:26:01 PM
UserName          : Bob

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UploadSigningCertificate](#) 中的。

Register-IAMGroupPolicy

以下代码示例演示了如何使用 Register-IAMGroupPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **TesterPolicy** 的客户管理型策略附加到 IAM 组 **Testers**。该组中的用户会立即受到该策略默认版本中所定义权限的影响。

```

Register-IAMGroupPolicy -GroupName Testers -PolicyArn
arn:aws:iam::123456789012:policy/TesterPolicy

```

示例 2：此示例将名为的 AWS 托管策略附加 **AdministratorAccess** 到 IAM 组 **Admins**。该组中的用户会立即受到该策略最新版本中所定义权限的影响。

```

Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachGroupPolicy](#) 中的。

Register-IAMRolePolicy

以下代码示例演示了如何使用 Register-IAMRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将名为的 AWS 托管策略附加**SecurityAudit**到 IAM 角色**CoSecurityAuditors**。担任该角色的用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn  
arn:aws:iam::aws:policy/SecurityAudit
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachRolePolicy](#) 中的。

Register-IAMUserPolicy

以下代码示例演示了如何使用 Register-IAMUserPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将名为的 AWS 托管策略附加**AmazonCognitoPowerUser**到 IAM 用户**Bob**。用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/  
AmazonCognitoPowerUser
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AttachUserPolicy](#) 中的。

Remove-IAMAccessKey

以下代码示例演示了如何使用 Remove-IAMAccessKey。

适用于 PowerShell V4 的工具

示例 1：此示例 **AKIAIOSFODNN7EXAMPLE** 从名为 **Bob** 的用户删除带有密钥 ID 的 AWS 访问密钥 **对 Bob**。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteAccessKey](#) 中的。

Remove-IAMAccountAlias

以下代码示例演示了如何使用 Remove-IAMAccountAlias。

适用于 PowerShell V4 的工具

示例 1：此示例从您的账户中删除了账户别名 AWS 账户。别名为 `https://mycompanyaws.signin.aws.amazon.com/console` 的用户登录页面不再起作用。相反，你必须使用带有 AWS 账户身份证号的原始 URL，网址为 `https://signin.aws.amazon.com/console.<accountidnumber>`

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteAccountAlias](#) 中的。

Remove-IAMAccountPasswordPolicy

以下代码示例演示了如何使用 Remove-IAMAccountPasswordPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例删除了密码策略 AWS 账户 并将所有值重置为其原始默认值。如果当前不存在密码策略，则会显示以下错误消息：找 PasswordPolicy 不到带有名称的账户策略。

```
Remove-IAMAccountPasswordPolicy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteAccountPasswordPolicy](#) 中的。

Remove-IAMClientIDFromOpenIDConnectProvider

以下代码示例演示了如何使用 Remove-IAMClientIDFromOpenIDConnectProvider。

适用于 PowerShell V4 的工具

示例 1：此示例将客户端 ID **My-TestApp-3** 从与 ARN 为 IAM OIDC 提供商 IDs 关联的客户列表中删除。**arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3  
-OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveClientIDFromOpenIDConnectProvider](#) 中的。

Remove-IAMGroup

以下代码示例演示了如何使用 Remove-IAMGroup。

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyTestGroup** 的 IAM 组。第一条命令删除作为该组成员的所有 IAM 用户，第二条命令删除该 IAM 组。这两条命令都可以在没有任何确认提示的情况下生效。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName  
MyTestGroup -Force  
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteGroup](#) 中的。

Remove-IAMGroupPolicy

以下代码示例演示了如何使用 Remove-IAMGroupPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例从 IAM 组 **Testers** 中删除名为 **TesterPolicy** 的内联策略。该组中的用户会立即失去该策略中定义的权限。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteGroupPolicy](#) 中的。

Remove-IAMInstanceProfile

以下代码示例演示了如何使用 Remove-IAMInstanceProfile。

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyAppInstanceProfile** 的 EC2 实例配置文件。第一条命令从实例配置文件中分离所有角色，然后第二条命令删除实例配置文件。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile  
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteInstanceProfile](#) 中的。

Remove-IAMLoginProfile

以下代码示例演示了如何使用 Remove-IAMLoginProfile。

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **Bob** 的 IAM 用户的登录配置文件。这会阻止用户登录控制台。AWS 它不会阻止用户使用可能仍附加到用户账户的 AWS 访问密钥运行任何 AWS CLI PowerShell、或 API 调用。

```
Remove-IAMLoginProfile -UserName Bob
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteLoginProfile](#) 中的。

Remove-IAMOpenIDConnectProvider

以下代码示例演示了如何使用 Remove-IAMOpenIDConnectProvider。

适用于 PowerShell V4 的工具

示例 1：此示例删除了连接到提供者 **example.oidcprovider.com** 的 IAM OIDC 提供者。确保更新或删除角色信任策略 **Principal** 元素中引用此提供者的所有角色。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteOpenIdConnectProvider](#) 中的。

Remove-IAMPolicy

以下代码示例演示了如何使用 Remove-IAMPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略。必须先通过运行 **Remove-IAMPolicyVersion** 删除默认版本以外的所有版本，然后才能删除该策略。您还必须将该策略与所有 IAM 用户、组或角色分离。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

示例 2：此示例通过先删除所有非默认策略版本，将其与所有附加的 IAM 实体分离，最后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后删除集合中的每个策略。第三行检索该策略附加到的所有 IAM 用户、组和角色。第四行到第六行将该策略与每个附加的实体分离。最后一行使用此命令删除托管策略，以及剩余的默认版本。该示例在需要抑制确认提示的所有行中都包含 **-Force** 开关参数。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy  
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |  
  Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force  
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn  
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn  
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn  
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn  
Remove-IAMPolicy $pol.Arn -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeletePolicy](#) 中的。

Remove-IAMPolicyVersion

以下代码示例演示了如何使用 Remove-IAMPolicyVersion。

适用于 PowerShell V4 的工具

示例 1：此示例从 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略中删除标识为 **v2** 的版本。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

示例 2：此示例通过先删除所有非默认策略版本，然后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后使用此命令删除集合中的每个策略。最后一行删除策略本身，以及剩余的默认版本。请注意，要成功删除托管策略，还必须使用 **Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy** 和 **Unregister-IAMRolePolicy** 命令将该策略与所有用户、组或角色分离。请参阅 **Remove-IAMPolicy** cmdlet 的示例。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
  Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeletePolicyVersion](#) 中的。

Remove-IAMRole

以下代码示例演示了如何使用 Remove-IAMRole。

适用于 PowerShell V4 的工具

示例 1：此示例从当前 IAM 账户中删除名为 **MyNewRole** 的角色。必须先使用 **Unregister-IAMRolePolicy** 命令分离所有托管策略，然后才能删除该角色。内联策略将与角色一起删除。

```
Remove-IAMRole -RoleName MyNewRole
```

示例 2：此示例将所有托管策略与名为 **MyNewRole** 的角色分离，然后删除该角色。第一行检索作为集合附加到该角色的所有托管策略，然后将集合中的每个策略与该角色分离。第二行删除该角色本身。内联策略将与角色一起删除。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -  
RoleName MyNewRole  
Remove-IAMRole -RoleName MyNewRole
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRole](#) 中的。

Remove-IAMRoleFromInstanceProfile

以下代码示例演示了如何使用 Remove-IAMRoleFromInstanceProfile。

适用于 PowerShell V4 的工具

示例 1：此示例 **MyNewRole** 从名为的 EC2 实例配置文件中删除名为的角色 **MyNewRole**。在 IAM 控制台中创建的实例配置文件始终与角色同名，如本示例所示。如果您在 API 或 CLI 中创建实例配置文件，则其可以有不同的名称。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole  
-Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveRoleFromInstanceProfile](#) 中的。

Remove-IAMRolePermissionsBoundary

以下代码示例演示了如何使用 Remove-IAMRolePermissionsBoundary。

适用于 PowerShell V4 的工具

示例 1：此示例展示了如何删除附加到 IAM 角色的权限边界。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRolePermissionsBoundary](#) 中的。

Remove-IAMRolePolicy

以下代码示例演示了如何使用 Remove-IAMRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 IAM 角色 **S3BackupRole** 中嵌入的内联策略 **S3AccessPolicy**。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteRolePolicy](#) 中的。

Remove-IAMRoleTag

以下代码示例演示了如何使用 Remove-IAMRoleTag。

适用于 PowerShell V4 的工具

示例 1：此示例从名为“MyRoleName”的角色中删除标签，标签键为“abac”。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagRole](#) 中的。

Remove-IAMSAMLProvider

以下代码示例演示了如何使用 Remove-IAMSAMLProvider。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider** 的 IAM SAML 2.0 提供者。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) SAMLProvider 中的 [删除](#)。

Remove-IAMServerCertificate

以下代码示例演示了如何使用 Remove-IAMServerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyServerCert** 的服务器证书。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteServerCertificate](#) 中的。

Remove-IAMServiceLinkedRole

以下代码示例演示了如何使用 Remove-IAMServiceLinkedRole。

适用于 PowerShell V4 的工具

示例 1：此示例删除服务关联角色。请注意，如果服务仍在使用此角色，则此命令将导致失败。

```
Remove-IAMServiceLinkedRole -RoleName  
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteServiceLinkedRole](#) 中的。

Remove-IAMSigningCertificate

以下代码示例演示了如何使用 Remove-IAMSigningCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例从名为 **Bob** 的 IAM 用户删除 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的签名证书。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteSigningCertificate](#) 中的。

Remove-IAMUser

以下代码示例演示了如何使用 Remove-IAMUser。

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **Bob** 的 IAM 用户。

```
Remove-IAMUser -UserName Bob
```

示例 2：此示例删除名为 **Theresa** 的 IAM 用户，以及必须先删除的所有元素。

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
$name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
-UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
$cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }
```

```
# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
    $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteUser](#) 中的。

Remove-IAMUserFromGroup

以下代码示例演示了如何使用 Remove-IAMUserFromGroup。

适用于 PowerShell V4 的工具

示例 1：此示例从组 **Testers** 中删除 IAM 用户 **Bob**。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

示例 2：此示例查找 IAM 用户 **Theresa** 所属的所有组，然后从这些组中删除 **Theresa**。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

示例 3：此示例显示了从 **Testers** 组中删除 IAM 用户 **Bob** 的另一种方法。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
Testers -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RemoveUserFromGroup](#) 中的。

Remove-IAMUserPermissionsBoundary

以下代码示例演示了如何使用 Remove-IAMUserPermissionsBoundary。

适用于 PowerShell V4 的工具

示例 1：此示例展示了如何删除附加到 IAM 用户的权限边界。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteUserPermissionsBoundary](#) 中的。

Remove-IAMUserPolicy

以下代码示例演示了如何使用 Remove-IAMUserPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例删除嵌入在名为 **Bob** 的 IAM 用户中的名为 **AccessToEC2Policy** 的内联策略。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

示例 2：此示例查找嵌入在名为 **Theresa** 的 IAM 用户中的所有内联策略，然后将其删除。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa  
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName  
    Theresa -Force }
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteUserPolicy](#) 中的。

Remove-IAMUserTag

以下代码示例演示了如何使用 Remove-IAMUserTag。

适用于 PowerShell V4 的工具

示例 1：此示例从名为“joe”的用户中删除标签键为“abac”和“xyzw”的标签。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```


- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagUser](#) 中的。

Remove-IAMVirtualMFADevice

以下代码示例演示了如何使用 Remove-IAMVirtualMFADevice。

适用于 PowerShell V4 的工具

示例 1：此示例删除 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 IAM 虚拟 MFA 设备。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

示例 2：此示例检查是否为 IAM 用户 Theresa 分配了 MFA 设备。如果找到设备，则会为 IAM 用户禁用该设备。如果设备是虚拟的，则会同时删除该设备。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
    $mfa.SerialNumber }
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteVirtualMfaDevice](#) 中的。

Request-IAMCredentialReport

以下代码示例演示了如何使用 Request-IAMCredentialReport。

适用于 PowerShell V4 的工具

示例 1：此示例请求生成一份新报告，报告可每四小时生成一次。如果上次报告仍然是最新的，则“状态”字段显示为 **COMPLETE**。使用 **Get-IAMCredentialReport** 查看已完成的报告。

```
Request-IAMCredentialReport
```

输出：

Description	State

```
-----  
No report exists. Starting a new report generation task  
-----  
STARTED
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GenerateCredentialReport](#) 中的。

Request-IAMServiceLastAccessedDetail

以下代码示例演示了如何使用 Request-IAMServiceLastAccessedDetail。

适用于 PowerShell V4 的工具

示例 1：此示例等同于 API 的 cmdlet。GenerateServiceLastAccessedDetails 这提供了一个任务 ID，可以在 Get-IAMServiceLastAccessedDetail 和 Get-中使用 IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GenerateServiceLastAccessedDetails](#) 中的。

Set-IAMDefaultPolicyVersion

以下代码示例演示了如何使用 Set-IAMDefaultPolicyVersion。

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy** 的策略的 v2 版本设置为默认活动版本。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -  
VersionId v2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) SetDefaultPolicyVersion](#) 中的。

Set-IAMRolePermissionsBoundary

以下代码示例演示了如何使用 Set-IAMRolePermissionsBoundary。

适用于 PowerShell V4 的工具

示例 1：此示例展示如何设置 IAM 角色的权限边界。您可以将 AWS 托管策略或自定义策略设置为权限边界。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutRolePermissionsBoundary](#) 中的。

Set-IAMUserPermissionsBoundary

以下代码示例演示了如何使用 Set-IAMUserPermissionsBoundary。

适用于 PowerShell V4 的工具

示例 1：此示例展示如何设置用户的权限边界。您可以将 AWS 托管策略或自定义策略设置为权限边界。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutUserPermissionsBoundary](#) 中的。

Sync-IAMMFADevice

以下代码示例演示了如何使用 Sync-IAMMFADevice。

适用于 PowerShell V4 的工具

示例 1：此示例将与 IAM 用户 **Bob** 关联且 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 MFA 设备与提供这两个身份验证码的身份验证器程序同步。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -  
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

示例 2：此示例将与 IAM 用户 **Theresa** 关联的 IAM MFA 设备与具有序列号 **ABCD12345678** 并提供两个身份验证码的物理设备同步。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -  
AuthenticationCode2 987654 -UserName Theresa
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResyncMfaDevice](#) 中的。

Unregister-IAMGroupPolicy

以下代码示例演示了如何使用 Unregister-IAMGroupPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterAccessPolicy** 的托管组策略与名为 **Testers** 的组分离。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

示例 2：此示例查找附加到名为 **Testers** 的组的所有托管策略，并将其与该组分离。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -  
Groupname Testers
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DetachGroupPolicy](#) 中的。

Unregister-IAMRolePolicy

以下代码示例演示了如何使用 Unregister-IAMRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** 的托管组策略与名为 **FedTesterRole** 的角色分离。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn  
arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

示例 2：此示例查找附加到名为 **FedTesterRole** 的角色的所有托管策略，并将其与该角色分离。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -  
Rolenam FedTesterRole
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DetachRolePolicy](#) 中的。

Unregister-IAMUserPolicy

以下代码示例演示了如何使用 Unregister-IAMUserPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterPolicy** 的托管策略与名为 **Bob** 的 IAM 用户分离。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/  
TesterPolicy
```

示例 2：此示例查找附加到名为 **Theresa** 的 IAM 用户的所有托管策略，并将这些策略与该用户分离。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username  
Theresa
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DetachUserPolicy](#) 中的。

Update-IAMAccessKey

以下代码示例演示了如何使用 Update-IAMAccessKey。

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **Bob** 的 IAM 用户的访问密钥 **AKIAIOSFODNN7EXAMPLE** 状态更改为 **Inactive**。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateAccessKey](#) 中的。

Update-IAMAccountPasswordPolicy

以下代码示例演示了如何使用 Update-IAMAccountPasswordPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定设置更新账户的密码策略。请注意，命令中未包含的任何参数都不会保持不变。相反，这些参数将重置为默认值。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry  
$false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20  
-RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -  
RequireUppercaseCharacters $true
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateAccountPasswordPolicy](#) 中的。

Update-IAMAssumeRolePolicy

以下代码示例演示了如何使用 Update-IAMAssumeRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例使用新的信任策略更新名为 **ClientRole** 的 IAM 角色，其内容来自文件 **ClientRolePolicy.json**。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw  
ClientRolePolicy.json)
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateAssumeRolePolicy](#) 中的。

Update-IAMGroup

以下代码示例演示了如何使用 Update-IAMGroup。

适用于 PowerShell V4 的工具

示例 1：此示例将 IAM 组 **Testers** 重命名为 **AppTesters**。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

示例 2：此示例将 IAM 组 **AppTesters** 的路径更改为 **/Org1/Org2/**。这会将该组的 ARN 更改为 **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateGroup](#) 中的。

Update-IAMLoginProfile

以下代码示例演示了如何使用 Update-IAMLoginProfile。

适用于 PowerShell V4 的工具

示例 1：此示例为 IAM 用户 **Bob** 设置了一个新的临时密码，并要求该用户在下次登录时更改密码。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateLoginProfile](#) 中的。

Update-IAMOpenIDConnectProviderThumbprint

以下代码示例演示了如何使用 Update-IAMOpenIDConnectProviderThumbprint。

适用于 PowerShell V4 的工具

示例 1：此示例更新了其 ARN 为 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 OIDC 提供者的证书指纹列表以使用新指纹。当与提供者关联的证书发生变化时，OIDC 提供者将共享新值。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList  
7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateOpenIdConnectProviderThumbprint](#) 中的。

Update-IAMRole

以下代码示例演示了如何使用 Update-IAMRole。

适用于 PowerShell V4 的工具

示例 1：此示例更新了角色描述和可以请求的角色会话的最长会话持续时间值（以秒为单位）。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -  
MaxSessionDuration 43200
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateRole](#) 中的。

Update-IAMRoleDescription

以下代码示例演示了如何使用 Update-IAMRoleDescription。

适用于 PowerShell V4 的工具

示例 1：此示例更新了您账户中 IAM 角色的描述。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateRoleDescription](#) 中的。

Update-IAMSAMLProvider

以下代码示例演示了如何使用 Update-IAMSAMLProvider。

适用于 PowerShell V4 的工具

示例 1：此示例使用文件 **SAMLMetaData.xml** 中的新 SAML 元数据文档更新 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFS** 的 IAM 中的 SAML 提供者。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。


```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/  
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateSamlProvider](#) 中的。

Update-IAMServerCertificate

以下代码示例演示了如何使用 Update-IAMServerCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **MyServerCertificate** 的证书重命名为 **MyRenamedServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -  
NewServerCertificateName MyRenamedServerCertificate
```

示例 2：此示例将名为 **MyServerCertificate** 的证书移动到路径 **/Org1/Org2/**。这会将该资源的 ARN 更改为 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /  
Org1/Org2/
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateServerCertificate](#) 中的。

Update-IAMSigningCertificate

以下代码示例演示了如何使用 Update-IAMSigningCertificate。

适用于 PowerShell V4 的工具

示例 1：此示例更新了与名为 **Bob** 且其证书 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的 IAM 用户关联的证书，以将其标记为非活动状态。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -  
UserName Bob -Status Inactive
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateSigningCertificate](#) 中的。

Update-IAMUser

以下代码示例演示了如何使用 Update-IAMUser。

适用于 PowerShell V4 的工具

示例 1：此示例将 IAM 用户 **Bob** 重命名为 **Robert**。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

示例 2：此示例将 IAM 用户 **Bob** 的路径更改为 **/Org1/Org2/**，这实际上将用户的 ARN 更改为 **arn:aws:iam::123456789012:user/Org1/Org2/bob**。

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateUser](#) 中的。

Write-IAMGroupPolicy

以下代码示例演示了如何使用 Write-IAMGroupPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **AppTesterPolicy** 的内联策略并将其嵌入到 IAM 组 **AppTesters** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **apptesterpolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -  
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutGroupPolicy](#) 中的。

Write-IAMRolePolicy

以下代码示例演示了如何使用 Write-IAMRolePolicy。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **FedTesterRolePolicy** 的内联策略并将其嵌入到 IAM 角色 **FedTesterRole** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **FedTesterPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -  
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutRolePolicy](#) 中的。

Write-IAMUserPolicy

以下代码示例演示了如何使用 Write-IAMUserPolicy。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **EC2AccessPolicy** 的内联策略并将其嵌入到 IAM 用户 **Bob** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **EC2AccessPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-  
Content -Raw EC2AccessPolicy.json)
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutUserPolicy](#) 中的。

使用适用于 V4 的工具的 Kinesis 示例 PowerShell

以下代码示例向您展示了如何使用带有 Kinesis 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

• [操作](#)

操作

Get-KINRecord

以下代码示例演示了如何使用 Get-KINRecord。

适用于 PowerShell V4 的工具

示例 1：此示例将展示如何从一条或多条记录中返回和提取数据。提供的迭代器 Get-KINRecord 用于确定要返回的记录的起始位置，在本例中，这些记录被捕获到变量 \$records 中。然后，可以通过对 \$records 集合进行索引来访问每条记录。假设记录中的数据是 UTF-8 编码的文本，则最后一个命令显示如何从对象 MemoryStream 中提取数据并将其作为文本返回到控制台。

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNpP"
```

输出：

MillisBehindLatest	NextShardIterator	Records
0	AAAAAAAAAAERNIq...uDn11HuUs	{Key1, Key2}

```
$records.Records[0]
```

输出：

ApproximateArrivalTimestamp	Data	PartitionKey	SequenceNumber
3/7/2016 5:14:33 PM	System.IO.MemoryStream	Key1	
4955986459776...931586			

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

输出：

```
test data from string
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetRecords](#) 中的。

Get-KINShardIterator

以下代码示例演示了如何使用 Get-KINShardIterator。

适用于 PowerShell V4 的工具

示例 1：返回指定分片和起始位置的分片迭代器。通过引用返回的流对象的 Shards 集合，可以从 Get-KINStream cmdlet 的输出中获取分片标识符和序列号的详细信息。返回的迭代器可以与 Get-KINRecord cmdlet 一起使用，在分片中提取数据记录。

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-0000000000000" -  
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

输出：

```
AAAAAAAAAAGIc....9VnbiRNpP
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetShardIterator](#) 中的。

Get-KINStream

以下代码示例演示了如何使用 Get-KINStream。

适用于 PowerShell V4 的工具

示例 1：返回指定流的详细信息。

```
Get-KINStream -StreamName "mystream"
```

输出：

```
HasMoreShards      : False  
RetentionPeriodHours : 24
```

```
Shards           : {}  
StreamARN        : arn:aws:kinesis:us-west-2:123456789012:stream/mystream  
StreamName       : mystream  
StreamStatus     : ACTIVE
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeStream](#) 中的。

New-KINStream

以下代码示例演示了如何使用 New-KINStream。

适用于 PowerShell V4 的工具

示例 1：创建新流。默认情况下，此 cmdlet 不返回任何输出，因此添加 -PassThru 开关以返回提供给 -StreamName 参数的值以供日后使用。

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateStream](#) 中的。

Remove-KINStream

以下代码示例演示了如何使用 Remove-KINStream。

适用于 PowerShell V4 的工具

示例 1：删除指定的流。在命令执行之前，系统会提示您进行确认。要取消确认提示，请使用 -Force 开关。

```
Remove-KINStream -StreamName "mystream"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteStream](#) 中的。

Write-KINRecord

以下代码示例演示了如何使用 Write-KINRecord。

适用于 PowerShell V4 的工具

示例 1：写入一条包含提供给 -Text 参数的字符串的记录。

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey "Key1"
```

示例 2：写入包含指定文件所包含数据的记录。该文件被视为字节序列，因此，如果它包含文本，则在将其与此 cmdlet 搭配使用之前，应使用任何必要的编码进行编写。

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey "Key2"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutRecord](#) 中的。

使用适用于 V4 的工具的 Lambda 示例 PowerShell

以下代码示例向您展示了如何使用带有 Lambda 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-LMResourceTag

以下代码示例演示了如何使用 Add-LMResourceTag。

适用于 PowerShell V4 的工具

示例 1：本示例将三个标签 (Washington、Oregon 和 California) 及其关联值添加到了由函数 ARN 标识的指定函数。

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon" = "Salem"; "California" = "Sacramento" }
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResource](#) 中的。

Get-LMAccountSetting

以下代码示例演示了如何使用 Get-LMAccountSetting。

适用于 PowerShell V4 的工具

示例 1：本示例是为了比较账户限制与账户使用情况

```
Get-LMAccountSetting | Select-Object
@{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
@{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

输出：

TotalCodeSizeLimit	TotalCodeSizeUsed
80530636800	15078795

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetAccountSettings](#) 中的。

Get-LMAlias

以下代码示例演示了如何使用 Get-LMAlias。

适用于 PowerShell V4 的工具

示例 1：本示例检索了特定 Lambda 函数别名的路由配置权重。

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
RoutingConfig
```

输出：


```
AdditionalVersionWeights
-----
{[1, 0.6]}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetAlias](#) 中的。

Get-LMFunctionConcurrency

以下代码示例演示了如何使用 Get-LMFunctionConcurrency。

适用于 PowerShell V4 的工具

示例 1：本示例获取了 Lambda 函数的预留并发

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

输出：

```
ReservedConcurrentExecutions
-----
100
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetFunctionConcurrency](#) 中的。

Get-LMFunctionConfiguration

以下代码示例演示了如何使用 Get-LMFunctionConfiguration。

适用于 PowerShell V4 的工具

示例 1：本示例返回了 Lambda 函数的版本特定配置。

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
"PowershellAlias"
```

输出：

```
CodeSha256           : uW0W0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize             : 1426
```

```

DeadLetterConfig      : Amazon.Lambda.Model.DeadLetterConfig
Description           : Version 3 to test Aliases
Environment           : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn           : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                      : PowershellAlias
FunctionName          : MylambdaFunction123
Handler               : lambda_function.launch_instance
KMSKeyArn             :
LastModified          : 2019-12-25T09:52:59.872+0000
LastUpdateStatus     : Successful
LastUpdateStatusReason :
LastUpdateStatusReasonCode :
Layers               : {}
MasterArn             :
MemorySize            : 128
RevisionId            : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                  : arn:aws:iam::123456789012:role/service-role/lambda
Runtime               : python3.8
State                 : Active
StateReason           :
StateReasonCode       :
Timeout               : 600
TracingConfig         : Amazon.Lambda.Model.TracingConfigResponse
Version               : 4
VpcConfig             : Amazon.Lambda.Model.VpcConfigDetail

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetFunctionConfiguration](#) 中的。

Get-LMFunctionList

以下代码示例演示了如何使用 Get-LMFunctionList。

适用于 PowerShell V4 的工具

示例 1：本示例展示了代码大小已排序的所有 Lambda 函数

```

Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
RunTime, Timeout, CodeSize

```

输出：

FunctionName	Runtime	Timeout
CodeSize		
-----	-----	-----
test	python2.7	3
243		
MyLambdaFunction123	python3.8	600
659		
myfuncpython1	python3.8	303
675		

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListFunctions](#) 中的。

Get-LMPolicy

以下代码示例演示了如何使用 Get-LMPolicy。

适用于 PowerShell V4 的工具

示例 1：本示例展示了 Lambda 函数的函数策略

```
Get-LMPolicy -FunctionName test -Select Policy
```

输出：

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "xxxx",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789102:function:test"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPolicy](#) 中的。

Get-LMProvisionedConcurrencyConfig

以下代码示例演示了如何使用 Get-LMProvisionedConcurrencyConfig。

适用于 PowerShell V4 的工具

示例 1：本示例获取了 Lambda 函数指定别名的预置并发配置。

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
Qualifier "NewAlias1"
```

输出：

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                    : IN_PROGRESS
StatusReason                              :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetProvisionedConcurrencyConfig`](#) 中的。

Get-LMProvisionedConcurrencyConfigList

以下代码示例演示了如何使用 `Get-LMProvisionedConcurrencyConfigList`。

适用于 PowerShell V4 的工具

示例 1：本示例检索了 Lambda 函数的预置并发配置列表。

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MyLambdaFunction123"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ListProvisionedConcurrencyConfigs`](#) 中的。

Get-LMResourceTag

以下代码示例演示了如何使用 `Get-LMResourceTag`。

适用于 PowerShell V4 的工具

示例 1：本示例检索了当前在指定函数上设置的标签及其值。

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

输出：

Key	Value
---	-----
California	Sacramento
Oregon	Salem
Washington	Olympia

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListTags](#) 中的。

Get-LMVersionsByFunction

以下代码示例演示了如何使用 Get-LMVersionsByFunction。

适用于 PowerShell V4 的工具

示例 1：本示例返回了 Lambda 函数各个版本的版本特定配置列表。

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

输出：

FunctionName	Runtime	MemorySize	Timeout	CodeSize	LastModified
RoleName					
-----	-----	-----	-----	-----	-----

MylambdaFunction123	python3.8	128	600	659	
2020-01-10T03:20:56.390+0000	lambda				
MylambdaFunction123	python3.8	128	5	1426	
2019-12-25T09:19:02.238+0000	lambda				
MylambdaFunction123	python3.8	128	5	1426	
2019-12-25T09:39:36.779+0000	lambda				
MylambdaFunction123	python3.8	128	600	1426	
2019-12-25T09:52:59.872+0000	lambda				

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListVersionsByFunction](#) 中的。

New-LMAlias

以下代码示例演示了如何使用 New-LMAlias。

适用于 PowerShell V4 的工具

示例 1：本示例为指定版本和路由配置创建了新的 Lambda 别名，可指定相应版本收到的调用请求的百分比。

```
New-LMAlias -FunctionName "MyLambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"} -Description "Alias for
version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateAlias](#) 中的。

Publish-LMFunction

以下代码示例演示了如何使用 Publish-LMFunction。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个以 MyFunction Lambda 命名的新 C# (d AWS otnetcore1.0 运行时) 函数，从本地文件系统上的 zip 文件中为该函数提供编译后的二进制文件 (可以使用相对路径或绝对路径)。C# Lambda 函数使用名称 AssemblyName:: Namespace 来指定函数的处理程序。ClassName:: MethodName。您应适当地替换处理程序规范中的程序集名称 (不带 .dll 后缀)、命名空间、类名和方法名等部分。新函数将根据提供的值设置环境变量“envvar1”和“envvar2”。

```
Publish-LMFunction -Description "My C# Lambda Function" `
-FunctionName MyFunction `
-ZipFilename .\MyFunctionBinaries.zip `
-Handler "AssemblyName::Namespace.ClassName::MethodName" `
-Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
-Runtime dotnetcore1.0 `
-Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

输出：

```
CodeSha256      : /NgBMd...gq71I=
CodeSize        : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
```

```

Handler      : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn    :
LastModified : 2016-12-29T23:50:14.207+0000
MemorySize   : 128
Role         : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime      : dotnetcore1.0
Timeout      : 3
Version      : $LATEST
VpcConfig    :

```

示例 2：本示例与前例类似，但要先将函数二进制文件上传到 Amazon S3 存储桶（该存储桶必须与预期的 Lambda 函数位于同一区域），然后在创建函数时引用生成的 S3 对象。

```

Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
    -FunctionName MyFunction `
    -BucketName amzn-s3-demo-bucket `
    -Key MyFunctionBinaries.zip `
    -Handler "AssemblyName::Namespace.ClassName::MethodName" `
    -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
    -Runtime dotnetcore1.0 `
    -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateFunction](#) 中的。

Publish-LMVersion

以下代码示例演示了如何使用 Publish-LMVersion。

适用于 PowerShell V4 的工具

示例 1：本示例创建了 Lambda 函数代码的现有快照版本

```

Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
Existing Snapshot of function code as a new version through Powershell"

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PublishVersion](#) 中的。

Remove-LMAlias

以下代码示例演示了如何使用 Remove-LMAlias。

适用于 PowerShell V4 的工具

示例 1：本示例删除了命令中提到的 Lambda 函数别名。

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteAlias](#) 中的。

Remove-LMFunction

以下代码示例演示了如何使用 Remove-LMFunction。

适用于 PowerShell V4 的工具

示例 1：本示例删除了特定版本的 Lambda 函数

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFunction](#) 中的。

Remove-LMFunctionConcurrency

以下代码示例演示了如何使用 Remove-LMFunctionConcurrency。

适用于 PowerShell V4 的工具

示例 1：本示例删除了 Lambda 函数的函数并发。

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteFunctionConcurrency](#) 中的。

Remove-LMPermission

以下代码示例演示了如何使用 Remove-LMPermission。

适用于 PowerShell V4 的工具

示例 1：此示例删除了 Lambda 函数中指定的 Lambda 函数 StatementId 的函数策略。

```
$policy = Get-LMPolicy -FunctionName "MyLambdaFunction123" -Select Policy |  
ConvertFrom-Json | Select-Object -ExpandProperty Statement  
Remove-LMPermission -FunctionName "MyLambdaFunction123" -StatementId $policy[0].Sid
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RemovePermission](#) 中的。

Remove-LMProvisionedConcurrencyConfig

以下代码示例演示了如何使用 Remove-LMProvisionedConcurrencyConfig。

适用于 PowerShell V4 的工具

示例 1：本示例删除了特定别名的预置并发配置。

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -Qualifier  
"NewAlias1"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteProvisionedConcurrencyConfig](#) 中的。

Remove-LMResourceTag

以下代码示例演示了如何使用 Remove-LMResourceTag。

适用于 PowerShell V4 的工具

示例 1：本示例从函数中删除了提供的标签。除非指定了 -Force 开关，否则在继续操作之前，cmdlet 会提示您进行确认。调用服务一次，即可删除标签。

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-  
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

示例 2：本示例从函数中删除了提供的标签。除非指定了 -Force 开关，否则在继续操作之前，cmdlet 会提示您进行确认。根据提供的标签调用了一次服务。

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource  
"arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagResource](#) 中的。

Update-LMAlias

以下代码示例演示了如何使用 Update-LMAlias。

适用于 PowerShell V4 的工具

示例 1：本示例更新了现有 Lambda 函数别名的配置。它将该 RoutingConfiguration 值更新为将 60% (0.6) 的流量转移到版本 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version  
2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight  
@{Name="1";Value="0.6"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateAlias](#) 中的。

Update-LMFunctionCode

以下代码示例演示了如何使用 Update-LMFunctionCode。

适用于 PowerShell V4 的工具

示例 1：使用指定 zip 文件中包含的新内容更新名为 MyFunction 的函数。对于 C# .NET 核心 Lambda 函数，zip 文件应包含编译后的程序集。

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

示例 2：本示例与前例类似，但使用包含了已更新代码的 Amazon S3 对象来更新函数。

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key  
UpdatedCode.zip
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateFunctionCode](#) 中的。

Update-LMFunctionConfiguration

以下代码示例演示了如何使用 Update-LMFunctionConfiguration。

适用于 PowerShell V4 的工具

示例 1：本示例更新了现有 Lambda 函数配置

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler  
"lambda_function.launch_instance" -Timeout 600 -Environment_Variable  
@{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/  
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:  
123456789101:MyfirstTopic
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateFunctionConfiguration](#) 中的。

Write-LMFunctionConcurrency

以下代码示例演示了如何使用 Write-LMFunctionConcurrency。

适用于 PowerShell V4 的工具

示例 1：本示例将函数的并发设置作为一个整体应用。

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -  
ReservedConcurrentExecution 100
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutFunctionConcurrency](#) 中的。

Write-LMProvisionedConcurrencyConfig

以下代码示例演示了如何使用 Write-LMProvisionedConcurrencyConfig。

适用于 PowerShell V4 的工具

示例 1：本示例将预置并发配置添加到了函数的别名

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -  
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutProvisionedConcurrencyConfig](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon 机器学习示例

以下代码示例向您展示如何使用带有 Amazon ML 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-MLBatchPrediction

以下代码示例演示了如何使用 Get-MLBatchPrediction。

适用于 PowerShell V4 的工具

示例 1：返回 id 为 ID 的批量预测的详细元数据。

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBatchPrediction](#) 中的。

Get-MLBatchPredictionList

以下代码示例演示了如何使用 Get-MLBatchPredictionList。

适用于 PowerShell V4 的工具

示例 1：返回 BatchPredictions 与请求中给出的搜索条件相匹配的所有数据记录及其关联数据记录的列表。

```
Get-MLBatchPredictionList
```

示例 2：返回状态为“已完成”的所有 BatchPredictions 内容的列表。

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeBatchPredictions](#) 中的。

Get-MLDataSource

以下代码示例演示了如何使用 Get-MLDataSource。

适用于 PowerShell V4 的工具

示例 1：返回 ID 为 ID 的 a DataSource 的元数据、状态和数据文件信息

```
Get-MLDataSource -DataSourceId ID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDataSource](#) 中的。

Get-MLDataSourceList

以下代码示例演示了如何使用 Get-MLDataSourceList。

适用于 PowerShell V4 的工具

示例 1：返回所有数据记录 DataSources 及其关联数据记录的列表。

```
Get-MLDataSourceList
```

示例 2：返回状态为“已完成”的所有 DataSources 内容的列表。

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeDataSources](#) 中的。

Get-MLEvaluation

以下代码示例演示了如何使用 Get-MLEvaluation。

适用于 PowerShell V4 的工具

示例 1：返回 id 为 ID 的 Evaluation 的元数据和状态。

```
Get-MLEvaluation -EvaluationId ID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetEvaluation](#) 中的。

Get-MLEvaluationList

以下代码示例演示了如何使用 Get-MLEvaluationList。

适用于 PowerShell V4 的工具

示例 1：返回所有 Evaluation 资源的列表

```
Get-MLEvaluationList
```

示例 2：返回状态为 COMPLETED 的所有 Evaluation 的列表。

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeEvaluations](#) 中的。

Get-MLModel

以下代码示例演示了如何使用 Get-MLModel。

适用于 PowerShell V4 的工具

示例 1：返回 ID 为 ID 的详细元数据、状态、架构和数据文件信息。 MLModel

```
Get-MLModel -ModelId ID
```

- 有关 API 的详细信息，请参阅 [Get MLModel in AWS Tools for PowerShell Cmdlet 参考 \(V4\)](#)。

Get-MLModelList

以下代码示例演示了如何使用 Get-MLModelList。

适用于 PowerShell V4 的工具

示例 1：返回所有 Model 及其关联数据记录的列表。

```
Get-MLModelList
```

示例 2：返回状态为 COMPLETED 的所有 Model 的列表。

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) MLModels 中的 [描述](#)。

Get-MLPrediction

以下代码示例演示了如何使用 Get-MLPrediction。

适用于 PowerShell V4 的工具

示例 1：向 id 为 ID 的模型的实时预测端点 URL 发送一条记录。

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Predict](#)。

New-MLBatchPrediction

以下代码示例演示了如何使用 New-MLBatchPrediction。

适用于 PowerShell V4 的工具

示例 1：为 id 为 ID 的模型创建新的批量预测请求，并将输出放在指定的 S3 位置。

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateBatchPrediction](#) 中的。

New-MLDataSourceFromS3

以下代码示例演示了如何使用 New-MLDataSourceFromS3。

适用于 PowerShell V4 的工具

示例 1：使用 S3 位置的数据创建数据来源，名称为 NAME，架构为 SCHEMA。

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3  
"s3://BUCKET/KEY" -DataSchema SCHEMA
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateDataSourceFromS3](#) 中的 S3。

New-MLEvaluation

以下代码示例演示了如何使用 New-MLEvaluation。

适用于 PowerShell V4 的工具

示例 1：为给定的数据来源 ID 和模型 ID 创建评估

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateEvaluation](#) 中的。

New-MLModel

以下代码示例演示了如何使用 New-MLModel。

适用于 PowerShell V4 的工具

示例 1：使用训练数据创建新模型。

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- 有关 API 的详细信息，请参阅 MLModel 在 AWS Tools for PowerShell Cmdlet 参考中 [创建](#) (V4)。

New-MLRealtimeEndpoint

以下代码示例演示了如何使用 New-MLRealtimeEndpoint。

适用于 PowerShell V4 的工具

示例 1：为给定模型 ID 创建新的实时预测端点。

```
New-MLRealtimeEndpoint -ModelId ID
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateRealtimeEndpoint](#) 中的。

使用 V4 版工具的 Macie 示例 PowerShell

以下代码示例向您展示了如何使用带有 Macie 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-MAC2FindingList

以下代码示例演示了如何使用 Get-MAC2FindingList。

适用于 PowerShell V4 的工具

示例 1：返回包含敏感数据检测结果的列表，其类型 FindingIds 为 “CREDIT_CARD_NUMBER” 或 “US_SOCIAL_SECURITY_NUMBER”

```
$criterionAddProperties = New-Object
    Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
    "CREDIT_CARD_NUMBER"
    "US_SOCIAL_SECURITY_NUMBER"
)

$FindingCriterion = @{
    'classificationDetails.result.sensitiveData.detections.type' =
        [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListFindings](#) 中的。

AWS 价目表 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS 价目表的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-PLSAttributeValue

以下代码示例演示了如何使用 Get-PLSAttributeValue。

适用于 PowerShell V4 的工具

示例 1：返回 us-east-1 区域的亚马逊属性 “VolumeType” EC2 的值。

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-east-1
```

输出：

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetAttributeValues](#) 中的。

Get-PLSProduct

以下代码示例演示了如何使用 Get-PLSProduct。

适用于 PowerShell V4 的工具

示例 1：亚马逊所有商品的退货详情 EC2。

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

输出：

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
```

```
Region","storage":"EBS only","instanceFamily":"Memory
optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
(Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
to 10 Gigabit","servicename":"Amazon Elastic Compute
Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
(Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

示例 2：返回亚马逊 EC2 在 us-east-1 地区的数据，这些数据按支持 SSD 的“通用”卷类型进行筛选。

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
@{Type="TERM_MATCH";Field="volumeType";Value="General
Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-
east-1
```

输出：

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
Purpose","maxIopsvolume":"10000"},...
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetProducts](#) 中的。

Get-PLSService

以下代码示例演示了如何使用 Get-PLSService。

适用于 PowerShell V4 的工具

示例 1：返回 us-east-1 区域中所有可用服务代码的元数据。

```
Get-PLSService -Region us-east-1
```

输出：

AttributeNames	ServiceCode
-----	-----
{productFamily, servicecode, groupDescription, termType...}	AWSBudgets

{productFamily, servicecode, termType, usagetype...}	AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}	AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}	AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}	AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}	AWSConfig
...	

示例 2：返回 us-east-1 EC2 区域中亚马逊服务的元数据。

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

输出：

AttributeNames	ServiceCode
-----	-----
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...}	AmazonEC2

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeServices](#) 中的。

使用适用于 PowerShell V4 的工具的 Resource Groups 示例

以下代码示例向您展示了如何使用带有 Resource Groups 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-RGResourceTag

以下代码示例演示了如何使用 Add-RGResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例向指定的资源组 arn 添加标签键“Instances”，其值为“workboxes”

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

输出：

Arn	Tags
---	----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes	{[Instances, workboxes]}

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Tag](#)。

Find-RGResource

以下代码示例演示了如何使用 Find-RGResource。

适用于 PowerShell V4 的工具

示例 1：此示例使用标签筛选器 ResourceQuery 创建 for Instance 资源类型并查找资源。

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @( @{
    Key = 'auto'
    Values = @('no')
  })
}

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty ResourceIdentifiers
```

输出：

ResourceArn	ResourceType
-----	-----

```
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SearchResources](#)中的。

Get-RGGroup

以下代码示例演示了如何使用 Get-RGGroup。

适用于 PowerShell V4 的工具

示例 1：此示例按组名称检索资源组

```
Get-RGGroup -GroupName auto-no
```

输出：

Description	GroupArn	Name
-----	-----	----
	arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no	auto-no

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetGroup](#)中的。

Get-RGGroupList

以下代码示例演示了如何使用 Get-RGGroupList。

适用于 PowerShell V4 的工具

示例 1：此示例列出已创建的资源组。

```
Get-RGGroupList
```

输出：

GroupArn	GroupName
-----	-----
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no	auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes	auto-yes

arn:aws:resource-groups:eu-west-1:123456789012:group/build600

build600

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListGroup](#)s 中的。

Get-RGGroupQuery

以下代码示例演示了如何使用 Get-RGGroupQuery。

适用于 PowerShell V4 的工具

示例 1：此示例获取给定资源组的资源查询

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

输出：

```
Query
-----
Type
-----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetGroupQuery](#) 中的。

Get-RGGroupResourceList

以下代码示例演示了如何使用 Get-RGGroupResourceList。

适用于 PowerShell V4 的工具

示例 1：此示例在按资源类型进行筛选的基础之上列出组资源

```
Get-RGGroupResourceList -Filter @{"Name"="resource-type";Values="AWS::EC2::Instance"}
-GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

输出：

ResourceArn	ResourceType


```

-----
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012 AWS::EC2::Instance
-----

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListGroupResources](#) 中的。

Get-RGResourceTag

以下代码示例演示了如何使用 Get-RGResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例列出给定资源组 arn 的标签

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

输出：

Key	Value
---	-----
Instances	workboxes

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetTags](#) 中的。

New-RGGroup

以下代码示例演示了如何使用 New-RGGroup。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名 TestPowerShellGroup 为的基于标签的新资源组 Resource Groups AWS 资源组。该组包括当前区域中使用标签键“名称”和标签值“test2”标记的 Amazon EC2 实例。该命令返回组的查询和类型以及操作结果。

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
```

```
$ResourceQuery.Query = '{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]} '
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
"Test resource group."
```

输出：

```
Query
      Type
-----
      ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":
["test2"]}]} TAG_FILTERS_1_0

LoggedAt      : 11/20/2018 2:40:59 PM
Group         : Amazon.ResourceGroups.Model.Group
ResourceQuery : Amazon.ResourceGroups.Model.ResourceQuery
Tags          : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength  : 338
HttpStatusCode : OK
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateGroup](#) 中的。

Remove-RGGroup

以下代码示例演示了如何使用 Remove-RGGroup。

适用于 PowerShell V4 的工具

示例 1：此示例移除指定的资源组

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

```
Description GroupArn
```

```
Name
```

```
-----
```

```
----
```

```
arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-elbv2
non-tag-cfn-elbv2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteGroup](#) 中的。

Remove-RGResourceTag

以下代码示例演示了如何使用 Remove-RGResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例从资源组中移除提到的标签

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes -Key Instances
```

输出：

```
Confirm
```

```
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-RGResourceTag (Untag)" on target "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

```
Arn
```

```
Keys
```

```
---
```

```
----
```

```
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Untag](#)。

Update-RGGroup

以下代码示例演示了如何使用 Update-RGGroup。

适用于 PowerShell V4 的工具

示例 1：此示例更新组的描述

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

输出：

```
Description          GroupArn
Name
-----
----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes auto-yes
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UpdateGroup](#) 中的。

Update-RGGroupQuery

以下代码示例演示了如何使用 Update-RGGroupQuery。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个查询对象并更新组的查询。

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
    ResourceTypeFilters = @( 'AWS::EC2::Instance' )
    TagFilters = @( @{
        Key='Environment'
        Values='Build600.11'
    })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

输出：

```
GroupName ResourceQuery
-----
```

```
build600 Amazon.ResourceGroups.Model.ResourceQuery
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateGroupQuery](#) 中的。

Resource Groups 使用适用于 V4 的 PowerShell 工具标记 API 示例

以下代码示例向您展示了如何使用带有 Resource Groups Tagging API 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-RGTResourceTag

以下代码示例演示了如何使用 Add-RGTResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例将值为“beta”和“preprod_test”的标签键“stage”和“version”添加到 Amazon S3 存储桶和 Amazon DynamoDB 表中。调用服务一次，即可应用这些标签。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
"version"="preprod_test" }
```

示例 2：此示例将指定的标签和值添加到 Amazon S3 存储桶和 Amazon DynamoDB 表中。对该服务进行了两次调用，即对于通过管道传入 cmdlet 的每个资源 ARN 进行一次调用。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
```

```
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TagResources](#) 中的。

Get-RGTResource

以下代码示例演示了如何使用 Get-RGTResource。

适用于 PowerShell V4 的工具

示例 1：返回某个区域中所有带标签的资源及与资源关联的标签键。如果没有向 cmdlet 提供-Region 参数，它将尝试从 shell 或实例元数据中推断出区域。EC2

```
Get-RGTResource
```

输出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::amzn-s3-demo-bucket	{stage, version, othertag}

示例 2：返回某个区域中指定类型的所有带标签的资源。每个服务名称和资源类型的字符串都与嵌入在资源的 Amazon 资源名称（ARN）中的字符串相同。

```
Get-RGTResource -ResourceType "s3"
```

输出：

ResourceARN	Tags
-----	----
arn:aws:s3:::amzn-s3-demo-bucket	{stage, version, othertag}

示例 3：返回某个区域中指定类型的所有带标签的资源。请注意，当资源类型通过管道传输到 cmdlet 时，将为每个提供的资源类型向服务发出一个调用。

```
"dynamodb","s3" | Get-RGTResource
```

输出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::amzn-s3-demo-bucket	{stage, version, othertag}

示例 4：返回与指定筛选条件匹配的所有带标签的资源。

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

输出：

ResourceARN	Tags
-----	----
arn:aws:s3:::amzn-s3-demo-bucket	{stage, version, othertag}

示例 5：返回与指定筛选条件和资源类型匹配的所有带标签的资源。

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

输出：

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

示例 6：返回与指定筛选条件匹配的所有带标签的资源。

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

输出：

ResourceARN	Tags
-----	----

```
arn:aws:dynamodb:us-west-2:123456789012:table/mytable {stage, version}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetResources](#) 中的。

Get-RGTTagKey

以下代码示例演示了如何使用 Get-RGTTagKey。

适用于 PowerShell V4 的工具

示例 1：返回指定区域中的所有标签键。如果未指定-Region 参数，cmdlet 将尝试从默认外壳区域或实例元数据中推断出该区域。EC2 请注意，标签键不会按任何特定顺序返回。

```
Get-RGTTagKey -region us-west-2
```

输出：

```
version
stage
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetTagKeys](#) 中的。

Get-RGTTagValue

以下代码示例演示了如何使用 Get-RGTTagValue。

适用于 PowerShell V4 的工具

示例 1：返回某个区域中指定标签的值。如果未指定-Region 参数，cmdlet 将尝试从默认外壳区域或实例元数据中推断出该区域。EC2

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

输出：

```
beta
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetTagValues](#) 中的。

Remove-RGTResourceTag

以下代码示例演示了如何使用 Remove-RGTResourceTag。

适用于 PowerShell V4 的工具

示例 1：从 Amazon S3 存储桶和 Amazon DynamoDB 表中移除标签键“stage”和“version”以及关联的值。调用服务一次，即可删除标签。在移除标签之前，cmdlet 将提示进行确认。要绕过确认，请添加 -Force 参数。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

示例 2：从 Amazon S3 存储桶和 Amazon DynamoDB 表中移除标签键“stage”和“version”以及关联的值。对该服务进行了两次调用，即对于通过管道传入 cmdlet 的每个资源 ARN 进行一次调用。每次调用之前，cmdlet 都会提示进行确认。要绕过确认，请添加 -Force 参数。

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [UntagResources](#) 中的。

使用 PowerShell V4 工具的 Route 53 示例

以下代码示例向您展示了如何使用带有 Route 53 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Edit-R53ResourceRecordSet

以下代码示例演示了如何使用 Edit-R53ResourceRecordSet。

适用于 PowerShell V4 的工具

示例 1：此示例为 www.example.com 创建 A 记录，并将 test.example.com 的 A 记录从 192.0.2.3 更改为 192.0.2.1。请注意，TXT 类型记录的更改值必须采用双引号。有关更多详细信息，请参阅 Amazon Route 53 文档。您可以使用 Get-R53Change cmdlet 进行轮询以确定更改何时完成。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
    and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
    ChangeBatch_Change=$change1,$change2,$change3
}
```

```
Edit-R53ResourceRecordSet @params
```

示例 2：此示例说明如何创建别名资源记录集。'Z22222222'是您要在其中创建别名资源记录集的 Amazon Route 53 托管区的 ID。'example.com' 是您要为其创建别名的 Zone Apex (机构根网域)，而 'www.example.com' 是您也想为其创建别名的子域。'Z1111111111111111'是负载均衡器的托管区域ID的示例，'-1111111111.us-east-1.elb.amazonaws.com'是负载均衡器域名的示例，亚马逊53号路由使用该域名回应example.com和www.example.com的查询。有关更多详细信息，请参阅 Amazon Route 53 文档。您可以使用 Get-R53Change cmdlet 进行轮询以确定更改何时完成。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z22222222"
    ChangeBatch_Comment="This change batch creates two alias resource record sets, one
    for the zone apex, example.com, and one for www.example.com, that both point to
    example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

示例 3：此示例为 `www.example.com` 创建了两条 A 记录。在四分之一的时间 ($1/(1+3)$)，Amazon Route 53 使用第一个资源记录集的两个值 (`192.0.2.9` 和 `192.0.2.10`) 来响应 `www.example.com` 的查询。在四分之三的时间 ($3/(1+3)$)，Amazon Route 53 使用第二个资源记录集的两个值 (`192.0.2.11` 和 `192.0.2.12`) 来响应 `www.example.com` 的查询。有关更多详细信息，请参阅 Amazon Route 53 文档。您可以使用 `Get-R53Change` cmdlet 进行轮询以确定更改何时完成。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets, each
of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

示例 4：此示例说明如何创建加权别名资源记录集，假设 `example.com` 是要为其创建加权别名资源记录集的域。SetIdentifier 将两个加权别名资源记录集彼此区分开来。此元素是必需的，因为两个资源记录集的“名称”和“类型”元素具有相同的值。Z11111111111111 和 Z33333333333333 是 ELB 负载均衡器的托管区域的示例，值 IDs 为指定。DNSName `example-load-balancer-2222222222.us-east-1.elb.amazonaws.com` 和 `example-load-balancer`

4444444444.us-east-1.elb.amazonaws.com 是亚马逊 Route 53 回复 example.com 查询的 Elastic Load Balancing 域名的示例。有关更多详细信息，请参阅 Amazon Route 53 文档。您可以使用 Get-R53Change cmdlet 进行轮询以确定更改何时完成。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z11111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z33333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
record sets. Amazon Route 53 responds to queries for example.com with the first ELB
domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

示例 5：此示例创建两个延迟别名资源记录集，一个用于美国西部（俄勒冈州）地区（us-west-2）的 ELB 负载均衡器，另一个用于亚太地区（新加坡）（ap-southeast-1）的负载均衡器。有关更多详细

信息，请参阅 Amazon Route 53 文档。您可以使用 Get-R53Change cmdlet 进行轮询以确定更改何时完成。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z11111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z22222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z555555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ChangeResourceRecordSets](#) 中的。

Get-R53AccountLimit

以下代码示例演示了如何使用 Get-R53AccountLimit。

适用于 PowerShell V4 的工具

示例 1：此示例返回可以使用当前账户创建的托管区的最大数量。

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

输出：

```
15
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetAccountLimit](#) 中的。

Get-R53CheckerIpRanges

以下代码示例演示了如何使用 Get-R53CheckerIpRanges。

适用于 PowerShell V4 的工具

示例 1：此示例返回 Route53 运行状况检查器的 CIDRs

```
Get-R53CheckerIpRanges
```

输出：

```
15.177.2.0/23
15.177.6.0/23
15.177.10.0/23
15.177.14.0/23
15.177.18.0/23
15.177.22.0/23
15.177.26.0/23
15.177.30.0/23
15.177.34.0/23
15.177.38.0/23
15.177.42.0/23
```

```
15.177.46.0/23
15.177.50.0/23
15.177.54.0/23
15.177.58.0/23
15.177.62.0/23
54.183.255.128/26
54.228.16.0/26
54.232.40.64/26
54.241.32.64/26
54.243.31.192/26
54.244.52.192/26
54.245.168.0/26
54.248.220.0/26
54.250.253.192/26
54.251.31.128/26
54.252.79.128/26
54.252.254.192/26
54.255.254.192/26
107.23.255.0/26
176.34.159.192/26
177.71.207.128/26
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetCheckerIpRanges](#) 中的。

Get-R53HostedZone

以下代码示例演示了如何使用 Get-R53HostedZone。

适用于 PowerShell V4 的工具

示例 1：返回 ID 为 Z1D633 的托管区域的详细信息 PJN98FT9。

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetHostedZone](#) 中的。

Get-R53HostedZoneCount

以下代码示例演示了如何使用 Get-R53HostedZoneCount。

适用于 PowerShell V4 的工具

示例 1：返回当前的公有和私有托管区域的总数 AWS 账户。

```
Get-R53HostedZoneCount
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetHostedZoneCount`](#) 中的。

Get-R53HostedZoneLimit

以下代码示例演示了如何使用 `Get-R53HostedZoneLimit`。

适用于 PowerShell V4 的工具

示例 1：此示例返回对可在指定托管区中创建的最大记录数的限制。

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

输出：

```
5
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetHostedZoneLimit`](#) 中的。

Get-R53HostedZoneList

以下代码示例演示了如何使用 `Get-R53HostedZoneList`。

适用于 PowerShell V4 的工具

示例 1：输出所有公有和私有托管区。

```
Get-R53HostedZoneList
```

示例 2：输出与 ID 为 `NZ8 X2CISAMPLE` 的可重复使用的委托集关联的所有托管区域

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListHostedZones](#) 中的。

Get-R53HostedZonesByName

以下代码示例演示了如何使用 Get-R53HostedZonesByName。

适用于 PowerShell V4 的工具

示例 1：按照域名以 ASCII 顺序返回所有公有和私有托管区。

```
Get-R53HostedZonesByName
```

示例 2：按照域名以 ASCII 顺序返回公有和私有托管区，开头为指定的 DNS 名称。

```
Get-R53HostedZonesByName -DnsName example2.com
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListHostedZonesByName](#) 中的。

Get-R53QueryLoggingConfigList

以下代码示例演示了如何使用 Get-R53QueryLoggingConfigList。

适用于 PowerShell V4 的工具

示例 1：此示例返回与当前 AWS 账户关联的 DNS 查询日志记录的所有配置。

```
Get-R53QueryLoggingConfigList
```

输出：

Id	HostedZoneId	CloudWatchLogsLogGroupArn
--	-----	-----
59b0fa33-4fea-4471-a88c-926476aaa40d	Z385PDS6EAAAZR	arn:aws:logs:us-east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063	Z94SJHBV1AAAAZ	arn:aws:logs:us-east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b	Z3MEQ8T7AAA1BF	arn:aws:logs:us-east-1:111111111112:log-group:/aws/route53/example3.com:*

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListQueryLoggingConfigs](#) 中的。

Get-R53ReusableDelegationSet

以下代码示例演示了如何使用 Get-R53ReusableDelegationSet。

适用于 PowerShell V4 的工具

示例 1：此示例检索有关指定委派集的信息，包括分配给委派集的四个名称服务器。

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

输出：

Id	CallerReference	NameServers
--	-----	-----
/delegationset/N23DS9X4AYEAAA	testcaller	{ns-545.awsdns-04.net, ns-1264.awsdns-30.org, ns-2004.awsdns-58.co.uk, ns-240.awsdns-30.com}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetReusableDelegationSet](#) 中的。

New-R53HostedZone

以下代码示例演示了如何使用 New-R53HostedZone。

适用于 PowerShell V4 的工具

示例 1：创建名为“example.com”的新托管区，并与可重复使用的委托集相关联。请注意，您必须为 CallerReference 参数提供一个值，这样在必要时需要重试的请求就不会有执行两次操作的风险。由于托管区域是在 VPC 中创建的，因此它自动为私有区域，因此您不应设置 -HostedZoneConfig _ PrivateZone 参数。

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
```

```
VPC_VPCRegion="us-east-1"
}
```

```
New-R53HostedZone @params
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateHostedZone](#) 中的。

New-R53QueryLoggingConfig

以下代码示例演示了如何使用 New-R53QueryLoggingConfig。

适用于 PowerShell V4 的工具

示例 1：此示例为指定的托管区创建新的 Route53 DNS 查询日志记录配置。Amazon Route53 将 DNS 查询日志发布到指定的 Cloudwatch 日志组。

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

输出：

QueryLoggingConfig	Location
-----	-----
Amazon.Route53.Model.QueryLoggingConfig	https://route53.amazonaws.com/2013-04-01/ queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateQueryLoggingConfig](#) 中的。

New-R53ReusableDelegationSet

以下代码示例演示了如何使用 New-R53ReusableDelegationSet。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个由 4 个名称服务器组成的可重用委派集，这些名称服务器可以由多个托管区重用。

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

输出：

DelegationSet	Location
-----	-----
Amazon.Route53.Model.DelegationSet	https://route53.amazonaws.com/2013-04-01/delegationset/N23DS9XAAAAAXM

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateReusableDelegationSet](#) 中的。

Register-R53VPCWithHostedZone

以下代码示例演示了如何使用 Register-R53VPCWithHostedZone。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的 VPC 与私有托管区相关联。

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAF -VPC_VPCId vpc-f1b9aaaa -VPC_VPCRegion us-east-1
```

输出：

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C3SCAAA633Z6DX	PENDING	01/28/2020 19:32:02	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考文档 \(V 4\) VPCWithHostedZone](#) 中的[关联](#)。

Remove-R53HostedZone

以下代码示例演示了如何使用 Remove-R53HostedZone。

适用于 PowerShell V4 的工具

示例 1：删除具有指定 ID 的托管区。在命令继续之前，您将收到进行确认的提示，除非您添加了 -Force 开关参数。

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteHostedZone](#) 中的。

Remove-R53QueryLoggingConfig

以下代码示例演示了如何使用 Remove-R53QueryLoggingConfig。

适用于 PowerShell V4 的工具

示例 1：此示例移除指定 DNS 查询日志记录的指定配置。

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteQueryLoggingConfig](#) 中的。

Remove-R53ReusableDelegationSet

以下代码示例演示了如何使用 Remove-R53ReusableDelegationSet。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的可重用委派集。

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteReusableDelegationSet](#) 中的。

Unregister-R53VPCFromHostedZone

以下代码示例演示了如何使用 Unregister-R53VPCFromHostedZone。

适用于 PowerShell V4 的工具

示例 1：此示例取消指定 VPC 与私有托管区的关联。

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa  
-VPC_VPCRegion us-east-1
```

输出：

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C2XFCAAAA9HKZG	PENDING	01/28/2020 10:35:55	

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V4\) VPCFrom HostedZone](#) 中的[取消关联](#)。

Update-R53HostedZoneComment

以下代码示例演示了如何使用 Update-R53HostedZoneComment。

适用于 PowerShell V4 的工具

示例 1：此命令更新指定托管区的注释。

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted zone"
```

输出：

```
Id                : /hostedzone/Z385PDS6AAAAAR
Name              : example.com.
CallerReference   : C5B55555-7147-EF04-8341-69131E805C89
Config           : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService     :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateHostedZoneComment](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon S3 示例

以下代码示例向您展示了如何使用带有 Amazon S3 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Copy-S3Object

以下代码示例演示了如何使用 Copy-S3Object。

适用于 PowerShell V4 的工具

示例 1：此命令将对象“sample.txt”从存储桶“test-files”复制到同一个存储桶，但新键为“sample-copy.txt”。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

示例 2：此命令将对象“sample.txt”从存储桶“test-files”复制到存储桶“backup-files”，键为“sample-copy.txt”。

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey  
sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

示例 3：此命令将对象“sample.txt”从存储桶“test-files”下载到名为“local-sample.txt”的本地文件。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

示例 4：将单个对象下载到指定的文件。下载的文件可以在 c:\downloads\data\archive.zip 中找到

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:  
\downloads
```

示例 5：将与指定的键前缀匹配的所有对象下载到本地文件夹。相对键层次结构将作为子文件夹保留在总体下载位置中。


```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CopyObject](#) 中的。

Get-S3ACL

以下代码示例演示了如何使用 Get-S3ACL。

适用于 PowerShell V4 的工具

示例 1：该命令获取 S3 对象的对象所有者的详细信息。

```
Get-S3ACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select  
AccessControlList.Owner
```

输出：

```
DisplayName Id  
----- --  
testusername 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [GetACL](#)。

Get-S3Bucket

以下代码示例演示了如何使用 Get-S3Bucket。

适用于 PowerShell V4 的工具

示例 1：此命令返回所有 S3 存储桶。

```
Get-S3Bucket
```

示例 2：此命令返回名为“test-files”的存储桶

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListBuckets](#) 中的。

Get-S3BucketAccelerateConfiguration

以下代码示例演示了如何使用 Get-S3BucketAccelerateConfiguration。

适用于 PowerShell V4 的工具

示例 1：如果为指定的存储桶启用了传输加速设置，则此命令将返回值 Enabled。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value
-----
Enabled
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetBucketAccelerateConfiguration](#) 中的。

Get-S3BucketAnalyticsConfiguration

以下代码示例演示了如何使用 Get-S3BucketAnalyticsConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶中名为“testfilter”的分析筛选条件的详细信息。

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId 'testfilter'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetBucketAnalyticsConfiguration](#) 中的。

Get-S3BucketAnalyticsConfigurationList

以下代码示例演示了如何使用 Get-S3BucketAnalyticsConfigurationList。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个分析配置。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListBucketAnalyticsConfigurations](#) 中的。

Get-S3BucketEncryption

以下代码示例演示了如何使用 Get-S3BucketEncryption。

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶关联的所有服务器端加密规则。

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetBucketEncryption](#) 中的。

Get-S3BucketInventoryConfiguration

以下代码示例演示了如何使用 Get-S3BucketInventoryConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的名为“testinventory”的清单的详细信息。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetBucketInventoryConfiguration](#) 中的。

Get-S3BucketInventoryConfigurationList

以下代码示例演示了如何使用 Get-S3BucketInventoryConfigurationList。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个清单配置。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListBucketInventoryConfigurations](#) 中的。

Get-S3BucketLocation

以下代码示例演示了如何使用 Get-S3BucketLocation。

适用于 PowerShell V4 的工具

示例 1：如果存在约束，则此命令返回存储桶“amzn-s3-demo-bucket”的位置约束。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Value
-----
ap-south-1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetBucketLocation](#) 中的。

Get-S3BucketLogging

以下代码示例演示了如何使用 Get-S3BucketLogging。

适用于 PowerShell V4 的工具

示例 1：此命令返回指定存储桶的日志记录状态。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

输出：

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketLogging](#) 中的。

Get-S3BucketMetricsConfiguration

以下代码示例演示了如何使用 Get-S3BucketMetricsConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令返回有关给定 S3 存储桶的名为“testfilter”的指标筛选条件的详细信息。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketMetricsConfiguration](#) 中的。

Get-S3BucketNotification

以下代码示例演示了如何使用 Get-S3BucketNotification。

适用于 PowerShell V4 的工具

示例 1：此示例检索给定存储桶的通知配置

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

输出：

```
Id      Topic  
--      -  
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketNotification](#) 中的。

Get-S3BucketPolicy

以下代码示例演示了如何使用 Get-S3BucketPolicy。

适用于 PowerShell V4 的工具

示例 1：此命令输出与给定 S3 存储桶关联的存储桶策略。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetBucketPolicy](#) 中的。

Get-S3BucketPolicyStatus

以下代码示例演示了如何使用 Get-S3BucketPolicyStatus。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的策略状态，此状态指示存储桶是否为公有存储桶。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetBucketPolicyStatus](#) 中的。

Get-S3BucketReplication

以下代码示例演示了如何使用 Get-S3BucketReplication。

适用于 PowerShell V4 的工具

示例 1：返回在名为“amzn-s3-demo-bucket”的存储桶上设置的复制配置信息。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetBucketReplication](#) 中的。

Get-S3BucketRequestPayment

以下代码示例演示了如何使用 Get-S3BucketRequestPayment。

适用于 PowerShell V4 的工具

示例 1：返回名为“amzn-s3-demo-bucket”的存储桶的请求付款配置。默认情况下，存储桶所有者支付从存储桶进行下载的费用。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketRequestPayment](#) 中的。

Get-S3BucketTagging

以下代码示例演示了如何使用 Get-S3BucketTagging。

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶关联的所有标签。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketTagging](#) 中的。

Get-S3BucketVersioning

以下代码示例演示了如何使用 Get-S3BucketVersioning。

适用于 PowerShell V4 的工具

示例 1：此命令返回与给定存储桶相关的版本控制状态。

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketVersioning](#) 中的。

Get-S3BucketWebsite

以下代码示例演示了如何使用 Get-S3BucketWebsite。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶的静态网站配置的详细信息。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetBucketWebsite](#) 中的。

Get-S3CORSConfiguration

以下代码示例演示了如何使用 Get-S3CORSConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令返回一个对象，其中包含与给定 S3 存储桶相对应的所有 CORS 配置规则。

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
Configuration.Rules
```

输出：

```
AllowedMethods : {PUT, POST, DELETE}  
AllowedOrigins : {http://www.example1.com}  
Id             :  
ExposeHeaders  : {}  
MaxAgeSeconds  : 0  
AllowedHeaders : {*}  
  
AllowedMethods : {PUT, POST, DELETE}  
AllowedOrigins : {http://www.example2.com}  
Id             :  
ExposeHeaders  : {}  
MaxAgeSeconds  : 0  
AllowedHeaders : {*}  
  
AllowedMethods : {GET}  
AllowedOrigins : {*}  
Id             :  
ExposeHeaders  : {}  
MaxAgeSeconds  : 0
```



```
AllowedHeaders : {}
```

- 有关 API 的详细信息，请参阅 [Get CORSConfiguration in AWS Tools for PowerShell](#) Cmdlet 参考 (V4)。

Get-S3LifecycleConfiguration

以下代码示例演示了如何使用 Get-S3LifecycleConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例检索存储桶的生命周期配置。

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

输出：

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetLifecycleConfiguration](#) 中的。

Get-S3Object

以下代码示例演示了如何使用 Get-S3Object。

适用于 PowerShell V4 的工具

示例 1：此命令检索有关存储桶“test-files”中所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

示例 2：此命令从存储桶“test-files”中检索有关项目“sample.txt”的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 3：此命令从存储桶“test-files”中检索有关前缀为“sample”的所有项目的信息。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListObjects](#) 中的。

Get-S3ObjectLockConfiguration

以下代码示例演示了如何使用 Get-S3ObjectLockConfiguration。

适用于 PowerShell V4 的工具

示例 1：如果为给定的 S3 存储桶启用了对象锁定配置，则此命令将返回值“Enabled”。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

输出：

```
Value  
-----  
Enabled
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetObjectLockConfiguration](#) 中的。

Get-S3ObjectMetadata

以下代码示例演示了如何使用 Get-S3ObjectMetadata。

适用于 PowerShell V4 的工具

示例 1：此命令返回给定 S3 存储桶中密钥为“ListTrusts.txt”的对象的元数据。

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

输出：

```
Headers           : Amazon.S3.Model.HeadersCollection  
Metadata          : Amazon.S3.Model.MetadataCollection  
DeleteMarker      :
```

```

AcceptRanges                : bytes
ContentRange                 :
Expiration                   :
RestoreExpiration            :
RestoreInProgress            : False
LastModified                 : 01/01/2020 08:02:05
ETag                         : "d000011112a222e333e3bb4ee5d43d21"
MissingMeta                  : 0
VersionId                    : null
Expires                      : 01/01/0001 00:00:00
WebsiteRedirectLocation      :
ServerSideEncryptionMethod   : AES256
ServerSideEncryptionCustomerMethod :
ServerSideEncryptionKeyManagementServiceKeyId :
ReplicationStatus            :
PartsCount                   :
ObjectLockLegalHoldStatus    :
ObjectLockMode               :
ObjectLockRetainUntilDate    : 01/01/0001 00:00:00
StorageClass                  :
RequestCharged                :

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetObjectMetadata](#) 中的。

Get-S3ObjectRetention

以下代码示例演示了如何使用 Get-S3ObjectRetention。

适用于 PowerShell V4 的工具

示例 1：该命令返回保留对象之前的模式和日期。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetObjectRetention](#) 中的。

Get-S3ObjectTagSet

以下代码示例演示了如何使用 Get-S3ObjectTagSet。

适用于 PowerShell V4 的工具

示例 1：该示例返回与给定 S3 存储桶上存在的对象关联的标签。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Key  Value
---  -
test value
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetObjectTagging`](#) 中的。

Get-S3PreSignedURL

以下代码示例演示了如何使用 `Get-S3PreSignedURL`。

适用于 PowerShell V4 的工具

示例 1：该命令返回指定键和到期日期的预签名 URL。

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires  
'2023-11-16'
```

示例 2：该命令返回带有指定键和到期日期的目录存储桶的预签名 URL。

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true  
Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key  
'testkey' -Expire '2023-11-17'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) 中的 `GetPreSigned` 网址](#)。

Get-S3PublicAccessBlock

以下代码示例演示了如何使用 `Get-S3PublicAccessBlock`。

适用于 PowerShell V4 的工具

示例 1：该命令返回给定 S3 存储桶的公共访问权限屏蔽设置。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetPublicAccessBlock](#) 中的。

Get-S3Version

以下代码示例演示了如何使用 Get-S3Version。

适用于 PowerShell V4 的工具

示例 1：此命令返回有关给定 S3 存储桶中所有对象版本的元数据。

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
IsTruncated      : False
KeyMarker        :
VersionIdMarker  :
NextKeyMarker    :
NextVersionIdMarker :
Versions         : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
  ListTrusts.json}
Name             : amzn-s3-demo-bucket
Prefix          :
MaxKeys          : 1000
CommonPrefixes   : {}
Delimiter        :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListVersions](#) 中的。

New-S3Bucket

以下代码示例演示了如何使用 New-S3Bucket。

适用于 PowerShell V4 的工具

示例 1：此命令创建一个名为“sample-bucket”的新私有存储桶。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

示例 2：此命令创建一个名为“sample-bucket”的具有读写权限的新存储桶。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

示例 3：此命令创建一个名为“sample-bucket”的具有只读权限的新存储桶。

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

示例 4：此命令使用创建名为“amzn-s3-demo-bucket-use1-az5-x-s3”的新目录存储桶。

PutBucketConfiguration

```
$bucketConfiguration = @{  
    BucketInfo = @{  
        DataRedundancy = 'SingleAvailabilityZone'  
        Type = 'Directory'  
    }  
    Location = @{  
        Name = 'usw2-az1'  
        Type = 'AvailabilityZone'  
    }  
}  
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration  
$bucketConfiguration -Region us-west-2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutBucket](#) 中的。

Read-S3Object

以下代码示例演示了如何使用 Read-S3Object。

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“amzn-s3-demo-bucket”中检索项目“sample.txt”，并将其保存到当前位置名为“local-sample.txt”的文件中。在调用此命令之前，文件“local-sample.txt”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

示例 2：此命令从存储桶“amzn-s3-demo-bucket”中检索虚拟目录“DIR”，并将其保存到当前位置名为“Local-dir”的文件夹中。在调用此命令之前，文件夹“Local-DIR”不必存在。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

示例 3：将键以“.json”结尾的所有对象从存储桶名称中带有“config”的存储桶下载到指定文件夹中的文件。对象键用于设置文件名。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetObject](#) 中的。

Remove-S3Bucket

以下代码示例演示了如何使用 Remove-S3Bucket。

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“test-files”中移除所有对象和对象版本，然后删除该存储桶。在继续操作之前，该命令将提示进行确认。添加 -Force 开关可禁止确认。请注意，不能删除不为空的存储桶。

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteBucket](#) 中的。

Remove-S3BucketAnalyticsConfiguration

以下代码示例演示了如何使用 Remove-S3BucketAnalyticsConfiguration。

适用于 PowerShell V4 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testfilter”的分析筛选条件。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId 'testfilter'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteBucketAnalyticsConfiguration](#) 中的。

Remove-S3BucketEncryption

以下代码示例演示了如何使用 Remove-S3BucketEncryption。

适用于 PowerShell V4 的工具

示例 1：这将禁用为所提供的 S3 存储桶启用的加密。

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
target "s3casetestbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteBucketEncryption](#) 中的。

Remove-S3BucketInventoryConfiguration

以下代码示例演示了如何使用 Remove-S3BucketInventoryConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令删除与给定 S3 存储桶对应的名为 testInventoryName 的库存。

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
'testInventoryName'
```

输出：

```
Confirm
```



```
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteBucketInventoryConfiguration](#) 中的。

Remove-S3BucketMetricsConfiguration

以下代码示例演示了如何使用 Remove-S3BucketMetricsConfiguration。

适用于 PowerShell V4 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testmetrics”的指标筛选条件。

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteBucketMetricsConfiguration](#) 中的。

Remove-S3BucketPolicy

以下代码示例演示了如何使用 Remove-S3BucketPolicy。

适用于 PowerShell V4 的工具

示例 1：该命令移除与给定 S3 存储桶关联的存储桶策略。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeleteBucketPolicy](#) 中的。

Remove-S3BucketReplication

以下代码示例演示了如何使用 Remove-S3BucketReplication。

适用于 PowerShell V4 的工具

示例 1：删除与名为“amzn-s3-demo-bucket”的存储桶关联的复制配置。请注意，此操作需要 s3:DeleteReplicationConfiguration 操作的权限。在操作继续之前，系统将提示您进行确认 - 要取消确认，请使用 -Force 开关。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteBucketReplication](#) 中的。

Remove-S3BucketTagging

以下代码示例演示了如何使用 Remove-S3BucketTagging。

适用于 PowerShell V4 的工具

示例 1：此命令移除与给定 S3 存储桶关联的所有标签。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteBucketTagging](#) 中的。

Remove-S3BucketWebsite

以下代码示例演示了如何使用 Remove-S3BucketWebsite。

适用于 PowerShell V4 的工具

示例 1：此命令禁用给定 S3 存储桶的静态网站托管属性。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteBucketWebsite](#) 中的。

Remove-S3CORSConfiguration

以下代码示例演示了如何使用 Remove-S3CORSConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令移除给定 S3 存储桶的 CORS 配置。

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
target "amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考文档 \(V 4\) CORSConfiguration](#) 中的 [删除](#)。

Remove-S3LifecycleConfiguration

以下代码示例演示了如何使用 Remove-S3LifecycleConfiguration。

适用于 PowerShell V4 的工具

示例 1：该命令移除给定 S3 存储桶的所有生命周期规则。

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteLifecycleConfiguration](#) 中的。

Remove-S3MultipartUpload

以下代码示例演示了如何使用 Remove-S3MultipartUpload。

适用于 PowerShell V4 的工具

示例 1：此命令中止在 5 天前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

示例 2：此命令中止在 2014 年 1 月 2 日之前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,  
January 02, 2014"
```

示例 3：此命令中止在 2014 年 1 月 2 日 10:45:37 之前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02  
10:45:37"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AbortMultipartUpload](#) 中的。

Remove-S3Object

以下代码示例演示了如何使用 Remove-S3Object。

适用于 PowerShell V4 的工具

示例 1：此命令从存储桶“test-files”中移除对象“sample.txt”。在命令执行之前，系统会提示您进行确认；要取消提示，请使用 -Force 开关。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

示例 2：假设存储桶已配置为启用对象版本，则此命令会从存储桶“test-files”中移除对象“sample.txt”的指定版本。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

示例 3：此命令通过单个批量操作，从存储桶“test-files”中移除对象“sample1.txt”、“sample2.txt”和“sample3.txt”。无论删除的成功或错误状态如何，服务响应都将列出所有已处理的键。要仅获取服务无法处理的密钥的错误，请添加-ReportErrorsOnly 参数（也可以使用别名-Quiet 来指定此参数）。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

示例 4：此示例使用带有-KeyCollection 参数的内联表达式来获取要删除的对象的密钥。Get-S3Object 返回 Amazon.S3.Model.S3Object 实例的集合，每个实例都有一个标识对象的字符串类型的 Key 成员。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

示例 5：此示例获取存储桶中所有具有键前缀“prefix/subprefix”的对象并将其删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的-InputObject（别名-S3ObjectCollection）参数，这样只需调用一次服务即可批量删除操作。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

示例 6：此示例将代表删除标记的 Amazon.S3.Model.S3 ObjectVersion 实例的集合传送到 cmdlet 进行删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的-InputObject（别名-S3ObjectCollection）参数，这样只需调用一次服务即可批量删除操作。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

示例 7：此脚本演示如何通过构造要与 `-KeyAndVersionCollection` 参数一起使用的对象数组来批量删除一组对象（在本例中为删除标记）。

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{ $_.IsDeleteMarker -eq "True" }
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteObjects](#) 中的。

Remove-S3ObjectTagSet

以下代码示例演示了如何使用 `Remove-S3ObjectTagSet`。

适用于 PowerShell V4 的工具

示例 1：此命令移除给定 S3 存储桶中与键为“testfile.txt”的对象关联的所有标签。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
'^Key'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
"testfile.txt".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
testfile.txt
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteObjectTagging](#) 中的。

Remove-S3PublicAccessBlock

以下代码示例演示了如何使用 `Remove-S3PublicAccessBlock`。

适用于 PowerShell V4 的工具

示例 1：此命令关闭给定存储桶的屏蔽公共访问权限配置。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

输出：

```
amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DeletePublicAccessBlock](#) 中的。

Set-S3BucketEncryption

以下代码示例演示了如何使用 Set-S3BucketEncryption。

适用于 PowerShell V4 的工具

示例 1：此命令使用给定存储桶上的 Amazon S3 托管密钥 (SSE-S3) 启用默认 AES256 服务器端加密。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutBucketEncryption](#) 中的。

Test-S3Bucket

以下代码示例演示了如何使用 Test-S3Bucket。

适用于 PowerShell V4 的工具

示例 1：如果存储桶存在，此命令返回 True，否则返回 False。即使存储桶不属于该用户，该命令也会返回 True。

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [Test-S3Bucket](#) 中的。

Write-S3BucketAccelerateConfiguration

以下代码示例演示了如何使用 Write-S3BucketAccelerateConfiguration。

适用于 PowerShell V4 的工具

示例 1：此命令为给定 S3 存储桶启用传输加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutBucketAccelerateConfiguration](#) 中的。

Write-S3BucketNotification

以下代码示例演示了如何使用 Write-S3BucketNotification。

适用于 PowerShell V4 的工具

示例 1：此示例为 S3 事件配置 SNS 主题配置 ObjectRemovedDelete 并启用给定 s3 存储桶的通知

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic
```

示例 2：此示例 ObjectCreatedAll 为给定存储桶启用通知，将其发送到 Lambda 函数。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
```



```

Events = "s3:ObjectCreated:*"
FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
Id = "ObjectCreated-Lambda"
Filter = @{
    S3KeyFilter = @{
        FilterRules = @(
            @{Name="Prefix";Value="dada"}
            @{Name="Suffix";Value=".pem"}
        )
    }
}
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig

```

示例 3：此示例基于不同的键后缀创建 2 个不同的 Lambda 配置，并在单个命令中配置了这两个配置。

```

#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{

```

```

        FilterRules = @(
            @{Name="Prefix";Value="dada"}
            @{Name="Suffix";Value=".json"}
        )
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutBucketNotification](#) 中的。

Write-S3BucketReplication

以下代码示例演示了如何使用 Write-S3BucketReplication。

适用于 PowerShell V4 的工具

示例 1：此示例使用一条规则设置复制配置，允许将存储桶“amzn-s3-demo-bucket”中使用密钥名称前缀“”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。TaxDocs

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

示例 2：此示例设置了具有多个规则的复制配置，允许将使用密钥名称前缀“”或“”创建的任何新对象复制到“amzn-s3-demo-bucket”存储桶。TaxDocs OtherDocs 键前缀不得重叠。

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule

```

```

$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

示例 3：此示例更新了指定存储桶上的复制配置，以禁用控制将密钥名称前缀“”的对象复制到存储桶“amzn-s3-demo-bucketTaxDocs”的规则。

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutBucketReplication](#) 中的。

Write-S3BucketRequestPayment

以下代码示例演示了如何使用 Write-S3BucketRequestPayment。

适用于 PowerShell V4 的工具

示例 1：更新名为“amzn-s3-demo-bucket”的存储桶的请求付款配置，以便向从该存储桶请求下载的人员收取下载费用。默认情况下，存储桶拥有者支付下载的费用。要将请求付款设置回默认值，请使用“BucketOwner”作为 RequestPaymentConfiguration_Payer 参数。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutBucketRequestPayment](#) 中的。

Write-S3BucketTagging

以下代码示例演示了如何使用 Write-S3BucketTagging。

适用于 PowerShell V4 的工具

示例 1：此命令将两个标签应用于名为 **cloudtrail-test-2018** 的存储桶：一个标签的键为 Stage，值为 Test；另一个标签的键为 Environment，值为 Alpha。要验证标签已添加到存储桶，请运行 **Get-S3BucketTagging -BucketName bucket_name**。结果应显示您在第一个命令中应用于存储桶的标签。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。要添加或删除各标签，请运行资源组和标记 API cmdlet **Add-RGTResourceTag** 和 **Remove-RGTResourceTag**。或者，使用 AWS 管理控制台中的标签编辑器来管理 S3 存储桶标签。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

示例 2：此命令将名为 **cloudtrail-test-2018** 的存储桶传送到 **Write-S3BucketTagging** cmdlet。它将标签 Stage:Production 和 Department:Finance 应用于存储桶。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet  
@( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutBucketTagging](#) 中的。

Write-S3BucketVersioning

以下代码示例演示了如何使用 Write-S3BucketVersioning。

适用于 PowerShell V4 的工具

示例 1：该命令为给定 S3 存储桶启用版本控制。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status Enabled
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutBucketVersioning](#) 中的。

Write-S3BucketWebsite

以下代码示例演示了如何使用 Write-S3BucketWebsite。

适用于 PowerShell V4 的工具

示例 1：该命令为给定存储桶启用网站托管，索引文档为“index.html”，错误文档为“error.html”。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket' -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -WebsiteConfiguration_ErrorDocument 'error.html'
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) PutBucketWebsite](#) 中的。

Write-S3LifecycleConfiguration

以下代码示例演示了如何使用 Write-S3LifecycleConfiguration。

适用于 PowerShell V4 的工具

示例 1：此示例写入/替换 \$ 中提供的配置NewRule。此配置确保使用给定的前缀和标签值来限制范围对象。

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = @{
        Days= 50
    }
    Id = "Test-From-Write-cmdlet-1"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "non-use"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    "Status"= 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 75
    }
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$NewRule
```

示例 2：此示例使用筛选设置了多个规则。\$ ArchiveRule 将对象设置为在 30 天内存档到 Glacier，120 天后存档到 Glacier DeepArchive。对于前缀为“py”且标签：key“已存档”设置为“是”的对象，\$将在 150 天后ExpireRule 过期当前版本和先前版本。

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = @{
        Days= 150
    }
    Id = "Remove-in-150-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
```

```

        "Prefix" = "py"
    },
    [Amazon.S3.Model.LifecycleTagPredicate] @{
        "Tag"= @{
            "Key" = "archived"
            "Value" = "yes"
        }
    }
)
}
}
Status= 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 150
}
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = $null
    Id = "Archive-to-Glacier-in-30-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "reviewed"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    Status = 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 75
    }
    Transitions = @(
        @{
            Days = 30
            "StorageClass"= 'Glacier'
        },

```

```
@{
    Days = 120
    "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
}
)
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutLifecycleConfiguration](#) 中的。

Write-S3Object

以下代码示例演示了如何使用 Write-S3Object。

适用于 PowerShell V4 的工具

示例 1：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

示例 2：此命令将单个文件“sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。如果未提供 -Key 参数，则文件名将用作 S3 对象键。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

示例 3：此命令将单个文件“local-sample.txt”上传到 Amazon S3，在存储桶“测试文件”中创建密钥为“prefix/to/sample.txt”的对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

示例 4：此命令将子目录“Scripts”中的所有文件上传到存储桶“test-files”，并将公用密钥前缀“”应用于SampleScripts每个对象。每个上传的文件都有一个密钥“SampleScripts/filename”，其中“文件名”会有所不同。


```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

示例 5：此命令将本地目录“脚本”中的所有*.ps1 文件上传到存储桶“test-files”，并将公用密钥前缀“”应用于每个对象。SampleScripts每个上传的文件都将有一个密钥“SampleScripts/filename.ps1”，其中“文件名”会有所不同。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

示例 6：此命令创建一个新的 S3 对象，其中包含键为“sample.txt”的指定内容字符串。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

示例 7：此命令上传指定的文件（文件名用作键），并将指定的标签应用于新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

示例 8：此命令以递归方式上传指定的文件夹，并将指定的标签应用于所有新对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -  
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutObject](#)中的。

Write-S3ObjectRetention

以下代码示例演示了如何使用 Write-S3ObjectRetention。

适用于 PowerShell V4 的工具

示例 1：该命令为给定 S3 存储桶中的“testfile.txt”对象启用监管保留模式，直到日期“2019 年 12 月 31 日 00:00:00”。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [PutObjectRetention](#) 中的。

使用适用于 V4 的工具的 Security Hub CSPM 示例 PowerShell

以下代码示例向您展示了如何使用带有 Security Hub CSPM 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-SHUBFinding

以下代码示例演示了如何使用 Get-SHUBFinding。

适用于 PowerShell V4 的工具

示例 1：此命令从 Amazon EC2 服务中检索 Security Hub 发现的结果。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.ResourceType = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'AwsEc2'
}
Get-SHUBFinding -Filter $filter
```

示例 2：此命令从 AWS 账户 ID 为 123456789012 中检索 Security Hub 的调查结果。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
```

```
$filter.AwsAccountId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -  
Property @{  
    Comparison = 'EQUALS'  
    Value = '123456789012'  
}  
Get-SHUBFinding -Filter $filter
```

示例 3：此命令检索为标准“pci-dss”生成的 Security Hub 调查发现。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters  
$filter.GeneratorId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -  
Property @{  
    Comparison = 'PREFIX'  
    Value = 'pci-dss'  
}  
Get-SHUBFinding -Filter $filter
```

示例 4：此命令检索工作流程状态为 NOTIFIED 的严重程度为关键的 Security Hub 调查发现。

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters  
$filter.SeverityLabel = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -  
Property @{  
    Comparison = 'EQUALS'  
    Value = 'CRITICAL'  
}  
$filter.WorkflowStatus = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter  
-Property @{  
    Comparison = 'EQUALS'  
    Value = 'NOTIFIED'  
}  
Get-SHUBFinding -Filter $filter
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetFindings](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon SES 示例

以下代码示例向您展示如何使用带有 Amazon SES 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Get-SESIIdentity

以下代码示例演示了如何使用 Get-SESIIdentity。

适用于 PowerShell V4 的工具

示例 1：无论验证状态如何，此命令都会返回包含特定 AWS 账户的所有身份（电子邮件地址和域名）的列表。

```
Get-SESIIdentity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListIdentities](#) 中的。

Get-SESSendQuota

以下代码示例演示了如何使用 Get-SESSendQuota。

适用于 PowerShell V4 的工具

示例 1：此命令返回用户的当前发送限制。

```
Get-SESSendQuota
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetSendQuota](#) 中的。

Get-SESSendStatistic

以下代码示例演示了如何使用 Get-SESSendStatistic。

适用于 PowerShell V4 的工具

示例 1：此命令返回用户的发送统计信息。结果是数据点的列表，表示过去两周的发送活动。列表中的每个数据点都包含 15 分钟间隔的统计信息。

```
Get-SESSendStatistic
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `GetSendStatistics`](#) 中的。

使用适用 PowerShell 于 V4 的工具的 Amazon SES API v2 示例

以下代码示例向您展示了如何使用带有 Amazon SES API v2 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Send-SES2Email

以下代码示例演示了如何使用 Send-SES2Email。

适用于 PowerShell V4 的工具

示例 1：此示例演示如何发送标准电子邮件。

```
Send-SES2Email -FromEmailAddress "sender@example.com" -Destination_ToAddress  
"recipient@example.com" -Subject_Data "Email Subject" -Text_Data "Email Body"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `SendEmail`](#) 中的。

使用适用于 V4 的工具的 Amazon SNS 示例 PowerShell

以下代码示例向您展示如何使用带有 Amazon SNS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Publish-SNSMessage

以下代码示例演示了如何使用 Publish-SNSMessage。

适用于 PowerShell V4 的工具

示例 1：此示例显示发布一条 MessageAttribute 声明为内联的消息。

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
"Hello" -MessageAttribute
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
StringValue = 'AnyCity'}}
```

示例 2：此示例显示发布一条事先 MessageAttributes 声明了多个消息的情况。

```
$cityAttributeValue = New-Object
Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"
```

```
$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
"Hello" -MessageAttribute $messageAttributes
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet Reference (V4)》中的 [Publish](#)。

使用适用于 V4 的工具的亚马逊 SQS 示例 PowerShell

以下代码示例向您展示如何使用带有 Amazon SQS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-SQSPermission

以下代码示例演示了如何使用 Add-SQSPermission。

适用于 PowerShell V4 的工具

示例 1：此示例 AWS 账户 允许指定用户从指定队列发送消息。

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddPermission](#) 中的。

Clear-SQSQueue

以下代码示例演示了如何使用 Clear-SQSQueue。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定队列中的所有消息。

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PurgeQueue](#) 中的。

Edit-SQSMessageVisibility

以下代码示例演示了如何使用 Edit-SQSMessageVisibility。

适用于 PowerShell V4 的工具

示例 1：此示例将指定队列中具有指定接收句柄的消息的可见性超时更改为 10 小时 (10 小时 * 60 分钟 * 60 秒 = 36000 秒)。

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ChangeMessageVisibility](#) 中的。

Edit-SQSMessageVisibilityBatch

以下代码示例演示了如何使用 Edit-SQSMessageVisibilityBatch。

适用于 PowerShell V4 的工具

示例 1：此示例更改了指定队列中具有指定接收句柄的 2 条消息的可见性超时。第一条消息的超时可见性更改为 10 小时 (10 小时 * 60 分钟 * 60 秒 = 36000 秒)。第二条消息的超时可见性更改为 5 小时 (5 小时 * 60 分钟 * 60 秒 = 18000 秒)。

```
$changeVisibilityRequest1 = New-Object  
Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
```



```
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
    $changeVisibilityRequest2
```

输出：

```
Failed      Successful
-----
{}          {Request2, Request1}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ChangeMessageVisibilityBatch](#) 中的。

Get-SQSDeadLetterSourceQueue

以下代码示例演示了如何使用 Get-SQSDeadLetterSourceQueue。

适用于 PowerShell V4 的工具

示例 1：此示例列 URLs 出了依赖指定队列作为死信队列的所有队列。

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

输出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDeadLetterSourceQueues](#) 中的。

Get-SQSQueue

以下代码示例演示了如何使用 Get-SQSQueue。

适用于 PowerShell V4 的工具

示例 1：此示例列出了所有队列。

```
Get-SQSQueue
```

输出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

示例 2：此示例列出了具有指定名称的所有队列。

```
Get-SQSQueue -QueueNamePrefix My
```

输出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListQueues](#) 中的。

Get-SQSQueueAttribute

以下代码示例演示了如何使用 Get-SQSQueueAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例列出了指定队列的所有属性。

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

输出：

```
VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp      : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                     : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
  [{"Sid":"Sid14
                                495134224EX","Effect":"Allow","Principal":
  {"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                398EXAMPLE:MyQueue","Condition":
  {"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
  {"Sid":
    "SendMessageFromMyQueue","Effect":"Allow","Principal":
  {"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":
    "arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[QueueArn, arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                [ApproximateNumberOfMessagesNotVisible, 0],
                                [ApproximateNumberOfMessagesDelayed, 0]...}
```

示例 2：此示例仅单独列出了指定队列的指定属性。

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

输出：

```
VisibilityTimeout           : 30
```

```

DelaySeconds                : 0
MaximumMessageSize          : 262144
MessageRetentionPeriod      : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp            : 2/11/2015 5:53:35 PM
LastModifiedTimestamp       : 12/29/2015 2:23:17 PM
QueueARN                    : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                      : {"Version":"2012-10-17",
  "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy", "Statement":
  [{"Sid":"Sid14
                                495134224EX", "Effect":"Allow", "Principal":
{"AWS":"*"}, "Action":"SQS:SendMessage", "Resource":"arn:aws:sqs:us-east-1:80
                                398EXAMPLE:MyQueue", "Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}}],
  {"Sid":
    "SendMessageFromMyQueue", "Effect":"Allow", "Principal":
{"AWS":"80398EXAMPLE"}, "Action":"SQS:SendMessage", "Resource":":
                                arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[MaximumMessageSize, 262144],
  [VisibilityTimeout, 30]}

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetQueueAttributes](#) 中的。

Get-SQSQueueUrl

以下代码示例演示了如何使用 Get-SQSQueueUrl。

适用于 PowerShell V4 的工具

示例 1：此示例列出了具有指定名称的队列的 URL。

```
Get-SQSQueueUrl -QueueName MyQueue
```

输出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetQueueUrl](#) 中的。

New-SQSQueue

以下代码示例演示了如何使用 New-SQSQueue。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定名称创建队列。

```
New-SQSQueue -QueueName MyQueue
```

输出：

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateQueue](#) 中的。

Receive-SQSMessage

以下代码示例演示了如何使用 Receive-SQSMessage。

适用于 PowerShell V4 的工具

示例 1：此示例列出了指定队列将要接收的消息（最多 10 条）的信息。该信息将包含指定消息属性的值（如果存在）。

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName  
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

输出：

```
Attributes           : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,  
  1451495923744]}
```

```
Body                 : Information about John Doe's grade.  
MD50fBody           : ea572796e3c231f974fe75d89EXAMPLE  
MD50fMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
```

```

MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
                          [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReceiveMessage](#) 中的。

Remove-SQSMessage

以下代码示例演示了如何使用 Remove-SQSMessage。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定队列中具有指定接收句柄的消息。

```

Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
-ReceiptHandle AQEBd329...v6gl8Q==

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteMessage](#) 中的。

Remove-SQSMessageBatch

以下代码示例演示了如何使用 Remove-SQSMessageBatch。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定队列中具有指定接收句柄的 2 条消息。

```

$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBq0VY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2

```

输出：

```
Failed      Successful
-----
{}          {Request1, Request2}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteMessageBatch](#) 中的。

Remove-SQSPermission

以下代码示例演示了如何使用 Remove-SQSPermission。

适用于 PowerShell V4 的工具

示例 1：此示例从指定队列中删除具有指定标签的权限设置。

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RemovePermission](#) 中的。

Remove-SQSQueue

以下代码示例演示了如何使用 Remove-SQSQueue。

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定队列。

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeleteQueue](#) 中的。

Send-SQSMessage

以下代码示例演示了如何使用 Send-SQSMessage。

适用于 PowerShell V4 的工具

示例 1：此示例向指定队列发送一条具有指定属性和消息正文的消息，消息传递延迟 10 秒。

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl https://
sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

输出：

MD50fMessageAttributes	MD50fMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SendMessage](#) 中的。

Send-SQSMessageBatch

以下代码示例演示了如何使用 Send-SQSMessageBatch。

适用于 PowerShell V4 的工具

示例 1：此示例向指定队列发送 2 条具有指定属性和消息正文的消息。第一条消息的传递延迟了 15 秒，第二条消息的传递延迟了 10 秒。

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

输出：

```
Failed    Successful
```

```
-----
{}          {FirstMessage, SecondMessage}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SendMessageBatch](#) 中的。

Set-SQSQueueAttribute

以下代码示例演示了如何使用 Set-SQSQueueAttribute。

适用于 PowerShell V4 的工具

示例 1：此示例展示了如何设置策略，让队列订阅 SNS 主题。发布到主题的消息会被发送到订阅了该主题的队列。

```
# 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 -AttributeName "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
        }
      }
    }
  ]
}
```

```
]
}
"@

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

示例 2：此示例为指定队列设置指定属性。

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
"131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [SetQueueAttributes](#) 中的。

AWS STS 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS STS 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Convert-STSAuthorizationMessage

以下代码示例演示了如何使用 Convert-STSAuthorizationMessage。

适用于 PowerShell V4 的工具

示例 1：对为响应请求而返回的所提供编码消息内容中包含的其他信息进行解码。对其他信息进行编码的原因是，授权状态的详细信息可能构成请求操作的用户不应看到的特权信息。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DecodeAuthorizationMessage](#) 中的。

Get-STSFederationToken

以下代码示例演示了如何使用 Get-STSFederationToken。

适用于 PowerShell V4 的工具

示例 1：使用“Bob”作为联合用户的名称，请求有效期为一小时的联合令牌。此名称可用于引用基于资源的策略（例如 Amazon S3 存储桶策略）中的联合用户名。以 JSON 格式提供的 IAM 策略可用于缩小 IAM 用户可用权限的范围。提供的策略授予的权限不能超过授予请求用户的权限，联合用户的最终权限是，基于传递的策略和 IAM 用户策略交集的限制性最强的集合。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds 3600
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetFederationToken](#) 中的。

Get-STSSessionToken

以下代码示例演示了如何使用 Get-STSSessionToken。

适用于 PowerShell V4 的工具

示例 1：返回包含在设定时间段内有效的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于请求临时凭证的凭证是根据当前 shell 默认值推断出来的。要指定其他凭据，请使用 -ProfileName 或 -AccessKey /-SecretKey 参数。

```
Get-STSSessionToken
```

输出：

AccessKeyId	Expiration
SecretAccessKey	SessionToken

```

-----
-----
EXAMPLEACCESSKEYID          2/16/2015 9:12:28 PM
examplesecretaccesskey...    SamPleToken.....

```

示例 2：返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于发出请求的凭证是从指定的配置文件中获得的。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

输出：

```

AccessKeyId          Expiration
SecretAccessKey      SessionToken
-----
-----
EXAMPLEACCESSKEYID  2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleToken.....

```

示例 3：使用与其凭证在配置文件“myprofilename”中指定的账户关联的 MFA 设备的标识号和该设备提供的值，返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

输出：

```

AccessKeyId          Expiration
SecretAccessKey      SessionToken
-----
-----
EXAMPLEACCESSKEYID  2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleToken.....

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetSessionToken](#) 中的。

Use-STSRole

以下代码示例演示了如何使用 Use-STSRole。

适用于 PowerShell V4 的工具

示例 1：返回一组临时证书（访问密钥、私有密钥和会话令牌），这些证书可用一小时来访问请求用户通常可能无法访问的 AWS 资源。返回的凭证具有所担任角色的访问策略和所提供策略允许的权限（您不能使用所提供策略授予超出所担任角色访问策略定义权限的权限）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
Policy "...JSON policy..." -DurationInSeconds 3600
```

示例 2：返回一组有效期为一小时的临时凭证，其拥有的权限与所担任角色访问策略中定义的权限相同。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600
```

示例 3：返回一组临时凭证，提供与用于执行 cmdlet 的用户凭证关联的 MFA 的序列号和生成的令牌。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

示例 4：返回一组临时凭证，其已承担客户账户中定义的角色。对于第三方可以担任的每个角色，客户账户都必须使用标识符创建角色，每次担任该角色时都必须在 `-ExternalId` 参数中传递该标识符。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -  
DurationInSeconds 3600 -ExternalId "ABC123"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AssumeRole](#) 中的。

Use-STSWebIdentityRole

以下代码示例演示了如何使用 `Use-STSWebIdentityRole`。

适用于 PowerShell V4 的工具

示例 1：为已通过 Login with Amazon 身份提供者进行身份验证的用户返回一组临时凭证，有效期为一小时。这些凭证采用与角色 ARN 所标识角色关联的访问策略。或者，您可以将 JSON 策略传

递给 `-Policy` 参数，以进一步细化访问权限（您授予的权限不能超过与该角色关联的权限中可用的权限）。提供给 `-WebIdentityToken` 的值是身份提供商返回的唯一用户标识符。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"  
-RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/  
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) AssumeRoleWithWebIdentity](#) 中的。

支持 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带支持的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-ASACommunicationToCase

以下代码示例演示了如何使用 `Add-ASACommunicationToCase`。

适用于 PowerShell V4 的工具

示例 1：将电子邮件通信的正文添加到指定案例中。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CommunicationBody "Some text about the case"
```

示例 2：将电子邮件通信的正文添加到指定案例中，另加上电子邮件抄送行中包含的一个或多个电子邮件地址。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CcEmailAddress @"email1@address.com", "email2@address.com") -CommunicationBody  
"Some text about the case"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AddCommunicationToCase](#) 中的。

Get-ASACase

以下代码示例演示了如何使用 Get-ASACase。

适用于 PowerShell V4 的工具

示例 1：返回所有支持案例的详细信息。

```
Get-ASACase
```

示例 2：返回自指定日期和时间以来所有支持案例的详细信息。

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

示例 3：返回前 10 个支持案例的详细信息，包括已解决的支持案例。

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

示例 4：返回单个指定支持案例的详细信息。

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

示例 5：返回指定支持案例的详细信息。

```
Get-ASACase -CaseIdList @"case-12345678910-2013-c4c1d2bf33c5cf47",  
"case-18929034710-2011-c4fdeabf33c5cf47")
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeCases](#) 中的。

Get-ASACommunication

以下代码示例演示了如何使用 Get-ASACommunication。

适用于 PowerShell V4 的工具

示例 1：返回指定案例的所有通信。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

示例 2：返回自协调世界时 2012 年 1 月 1 日午夜以来针对指定案例的所有通信。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime  
"2012-01-10T00:00Z"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeCommunications](#) 中的。

Get-ASAService

以下代码示例演示了如何使用 Get-ASAService。

适用于 PowerShell V4 的工具

示例 1：返回所有可用的服务代码、名称和类别。

```
Get-ASAService
```

示例 2：返回带有指定代码的服务的名称和类别。

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

示例 3：返回指定服务代码的名称和类别。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

示例 4：返回指定服务代码的名称和类别（日语）。目前支持英语（“en”）和日语（“ja”）语言代码。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -  
Language "ja"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeServices](#) 中的。

Get-ASASeverityLevel

以下代码示例演示了如何使用 Get-ASASeverityLevel。

适用于 PowerShell V4 的工具

示例 1：返回可分配给 Support 案例的 AWS 严重性级别列表。

```
Get-ASASeverityLevel
```

示例 2：返回可以分配给 Support 案例的 AWS 严重性级别列表。级别名称以日语返回。

```
Get-ASASeverityLevel -Language "ja"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeSeverityLevels](#) 中的。

Get-ASATrustedAdvisorCheck

以下代码示例演示了如何使用 Get-ASATrustedAdvisorCheck。

适用于 PowerShell V4 的工具

示例 1：返回 Trusted Advisor 检查的集合。必须指定 Language 参数，该参数可以接受“en”（表示英语输出），也可以接受“ja”（表示日语输出）。

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrustedAdvisorChecks](#) 中的。

Get-ASATrustedAdvisorCheckRefreshStatus

以下代码示例演示了如何使用 Get-ASATrustedAdvisorCheckRefreshStatus。

适用于 PowerShell V4 的工具

示例 1：返回指定检查的刷新请求的当前状态。Request-ASATrusted AdvisorCheckRefresh 可用于请求刷新支票的状态信息。

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @"checkid1", "checkid2")
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrustedAdvisorCheckRefreshStatuses](#) 中的。

Get-ASATrustedAdvisorCheckResult

以下代码示例演示了如何使用 Get-ASATrustedAdvisorCheckResult。

适用于 PowerShell V4 的工具

示例 1：返回 Trusted Advisor 检查的结果。可用 Trusted Advisor 支票列表可以使用 Get-获取 ASATrustedAdvisorChecks。输出是检查的总体状态、上次运行检查的时间戳以及特定检查的唯一 checkid。要以日语输出结果，请添加 -Language“ja”参数。

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrustedAdvisorCheckResult](#) 中的。

Get-ASATrustedAdvisorCheckSummary

以下代码示例演示了如何使用 Get-ASATrustedAdvisorCheckSummary。

适用于 PowerShell V4 的工具

示例 1：返回指定 Trusted Advisor 检查的最新摘要。

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

示例 2：返回指定 Trusted Advisor 检查的最新摘要。

```
Get-ASATrustedAdvisorCheckSummary -CheckId @"checkid1", "checkid2")
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTrustedAdvisorCheckSummaries](#) 中的。

New-ASACase

以下代码示例演示了如何使用 New-ASACase。

适用于 PowerShell V4 的工具

示例 1：在 Su AWS pport Center 中创建新案例。-ServiceCode 和 -CategoryCode 参数的值可以使用 Get-ASAService cmdlet 获取。-SeverityCode 参数的值可以使用 Get-ASASeverityLevel cmdlet 获得。-IssueType 参数值可以是“客户服务”或“技术”。如果成功，则 AWS 输出 Support 案例编号。默认情况下，案例将用英语处理，要使用日语，请添加 -Language“ja”参数。-ServiceCode、-CategoryCode、-主题和 -CommunicationBody 参数是必需的。

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode  
"low" -Subject "subject text" -CommunicationBody "description of the case" -  
CcEmailAddress @"email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateCase](#) 中的。

Request-ASATrustedAdvisorCheckRefresh

以下代码示例演示了如何使用 Request-ASATrustedAdvisorCheckRefresh。

适用于 PowerShell V4 的工具

示例 1：请求刷新指定的 Trusted Advisor 检查。

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RefreshTrustedAdvisorCheck](#) 中的。

Resolve-ASACase

以下代码示例演示了如何使用 Resolve-ASACase。

适用于 PowerShell V4 的工具

示例 1：返回指定案例的初始状态和完成用于解决该案例的调用后的当前状态。

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ResolveCase](#) 中的。

使用 PowerShell V4 版工具的 Systems Manager 示例

以下代码示例向您展示了如何使用带有 Systems Manager 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Add-SSMResourceTag

以下代码示例演示了如何使用 Add-SSMResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例使用新标签更新维护时段。如果此命令成功，则无任何输出。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$option1 = @{Key="Stack";Value=@"Production"}}  
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType  
"MaintenanceWindow" -Tag $option1
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个标签。如果此命令成功，则无任何输出。

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow" -Tag $tag1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [AddTagsToResource](#) 中的。

Edit-SSMDocumentPermission

以下代码示例演示了如何使用 Edit-SSMDocumentPermission。

适用于 PowerShell V4 的工具

示例 1：此示例为文档的所有账户添加“共享”权限。如果此命令成功，则无任何输出。

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

示例 2：此示例为文档的特定账户添加“共享”权限。如果此命令成功，则无任何输出。

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ModifyDocumentPermission](#) 中的。

Get-SSMActivation

以下代码示例演示了如何使用 Get-SSMActivation。

适用于 PowerShell V4 的工具

示例 1：此示例提供有关您账户激活的详细信息。

```
Get-SSMActivation
```

输出：

```

ActivationId      : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate       : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description        :
ExpirationDate     : 3/2/2017 12:01:51 AM
Expired           : False
IamRole            : AutomationRole
RegistrationLimit   : 10
RegistrationsCount : 0

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeActivations](#) 中的。

Get-SSMAssociation

以下代码示例演示了如何使用 Get-SSMAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例描述实例和文档之间的关联。

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

输出：

```

Name              : AWS-UpdateSSMAgent
InstanceId         : i-0000293ffd8c57862
Date              : 2/23/2017 6:55:22 PM
Status.Name       : Pending
Status.Date       : 2/20/2015 8:31:11 AM
Status.Message    : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeAssociation](#) 中的。

Get-SSMAssociationExecution

以下代码示例演示了如何使用 Get-SSMAssociationExecution。

适用于 PowerShell V4 的工具

示例 1：此示例返回所提供关联 ID 的执行

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

输出：

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion  : 2
CreatedTime        : 3/2/2019 8:53:29 AM
DetailedStatus     :
ExecutionId        : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate   : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status             : Success
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAssociationExecutions](#) 中的。

Get-SSMAssociationExecutionTarget

以下代码示例演示了如何使用 Get-SSMAssociationExecutionTarget。

适用于 PowerShell V4 的工具

示例 1：此示例显示作为关联执行目标组成部分的资源 ID 及其执行状态

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
Select-Object ResourceId, Status
```

输出：

ResourceId	Status
-----	-----
i-0b1b2a3456f7a890b	Success
i-01c12a45d6fc7a89f	Success
i-0a1caf234f56d7dc8	Success
i-012a3fd45af6dbcf	Failed
i-0ddc1df23c4a5fb67	Success

示例 2：此命令检查自昨天以来与命令文档关联的特定自动化的特定执行。它会进一步检查关联执行是否失败；如果失败，它将显示执行的命令调用详细信息以及实例 ID

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId $execution.OutputSource.OutputSourceId -
Detail:$true | Select-Object -ExpandProperty CommandPlugins
    }
}
```

输出：

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0 on
i-0a1caf234f56d7dc8

Name                : aws:runPowerShellScript
Output              :
                    -----ERROR-----
                    failed to run commands: exit status 1
OutputS3BucketName  :
OutputS3KeyPrefix   :
OutputS3Region      : eu-west-1
ResponseCode        : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl     :
StandardOutputUrl    :
Status              : Failed
StatusDetails       : Failed
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeAssociationExecutionTargets](#) 中的。

Get-SSMAssociationList

以下代码示例演示了如何使用 Get-SSMAssociationList。

适用于 PowerShell V4 的工具

示例 1：此示例列出实例的所有关联。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$filter1 = @{Key="InstanceId";Value=@"i-0000293ffd8c57862"}  
Get-SSMAssociationList -AssociationFilterList $filter1
```

输出：

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0  
DocumentVersion    :  
InstanceId         : i-0000293ffd8c57862  
LastExecutionDate  : 2/20/2015 8:31:11 AM  
Name              : AWS-UpdateSSMAgent  
Overview          : Amazon.SimpleSystemsManagement.Model.AssociationOverview  
ScheduleExpression :  
Targets           : {InstanceIds}
```

示例 2：此示例列出配置文档的所有关联。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$filter2 = @{Key="Name";Value=@"AWS-UpdateSSMAgent"}  
Get-SSMAssociationList -AssociationFilterList $filter2
```

输出：

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0  
DocumentVersion    :  
InstanceId         : i-0000293ffd8c57862  
LastExecutionDate  : 2/20/2015 8:31:11 AM  
Name              : AWS-UpdateSSMAgent  
Overview          : Amazon.SimpleSystemsManagement.Model.AssociationOverview  
ScheduleExpression :  
Targets           : {InstanceIds}
```

示例 3：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个滤镜。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
```

```
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

输出：

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListAssociations](#) 中的。

Get-SSMAssociationVersionList

以下代码示例演示了如何使用 Get-SSMAssociationVersionList。

适用于 PowerShell V4 的工具

示例 1：此示例检索所提供关联的所有版本。

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

输出：

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName     :
AssociationVersion  : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
```

```

Parameters      : {}
ScheduleExpression :
Targets         : {InstanceIds}

AssociationId    : 123a45a0-c678-9012-3456-78901234db5e
AssociationName  : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate     : 3/2/2019 8:53:29 AM
DocumentVersion :
MaxConcurrency  :
MaxErrors       :
Name            : AWS-GatherSoftwareInventory
OutputLocation  :
Parameters      : {}
ScheduleExpression : rate(30minutes)
Targets         : {InstanceIds}

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [ListAssociationVersions](#) 中的。

Get-SSMAutomationExecution

以下代码示例演示了如何使用 Get-SSMAutomationExecution。

适用于 PowerShell V4 的工具

示例 1：此示例显示自动化执行的详细信息。

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-f944-11e6-9d32-8fb2db27a909"
```

输出：

```

AutomationExecutionId    : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
                           are not authorized to perform this operation. Encoded

```

```
authorization failure message:
B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYek0RIrA42-
fv1x-04q5Fjff6glh
Yb6TI5b0GQeeNrpwNvpDzm0-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC10u6v30XU1KC9nZ7mPlWMFZNkSioQqpWEvMw-
GZktsQzm67q0hUhBN0LWYhbS
pkfiqzY-5nw3S0obx30fhd3EJa50_-
GjV_a0nFXQJa70ik40bF0rEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84X0F5WmA60f1pi5GI08i-
nRfZS6oDeU
gELBjjoFKD8s3L2aI0B6umWVxnQ0jqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUG0AZG-xjQFuvU_JZedWpla3xi-MZsmbIAifBI
(Service: AmazonEC2; Status Code: 403; Error Code:
UnauthorizedOperation; Request ID:
6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs : {[createImage.ImageId,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters : {[AutomationAssumeRole,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,
Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions : {launchInstance, updateOSSoftware, stopInstance,
createImage...}
```

示例 2：此示例列出给定自动化执行 ID 的步骤详细信息

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
StepName, Action, StepStatus, ValidNextSteps
```

输出：

StepName	Action	StepStatus	ValidNextSteps
-----	-----	-----	-----
LaunchInstance	aws:runInstances	Success	{OSCompatibilityCheck}
OSCompatibilityCheck	aws:runCommand	Success	{RunPreUpdateScript}
RunPreUpdateScript	aws:runCommand	Success	{UpdateEC2Config}
UpdateEC2Config	aws:runCommand	Cancelled	{}
UpdateSSMAgent	aws:runCommand	Pending	{}
UpdateAWSPVDriver	aws:runCommand	Pending	{}
UpdateAWSEnaNetworkDriver	aws:runCommand	Pending	{}
UpdateAWSNVMe	aws:runCommand	Pending	{}

InstallWindowsUpdates	aws:runCommand	Pending	{}
RunPostUpdateScript	aws:runCommand	Pending	{}
RunSysprepGeneralize	aws:runCommand	Pending	{}
StopInstance	aws:changeInstanceState	Pending	{}
CreateImage	aws:createImage	Pending	{}
TerminateInstance	aws:changeInstanceState	Pending	{}

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetAutomationExecution](#) 中的。

Get-SSMAutomationExecutionList

以下代码示例演示了如何使用 Get-SSMAutomationExecutionList。

适用于 PowerShell V4 的工具

示例 1：此示例描述与您的账户关联的所有活动和已终止的自动化执行。

```
Get-SSMAutomationExecutionList
```

输出：

```
AutomationExecutionId      : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus   : Failed
DocumentName                 : AWS-UpdateLinuxAmi
DocumentVersion              : 1
ExecutedBy                   : admin
ExecutionEndTime             : 2/22/2017 9:17:08 PM
ExecutionStartTime           : 2/22/2017 9:17:02 PM
LogFile                      :
Outputs                      : {[createImage.ImageId,
    Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

示例 2：此示例显示了“成功”以 AutomationExecutionStatus 外的执行的 executionID、文档、执行 start/end 时间戳

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
    -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
    AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
    AutoSize
```

输出：

AutomationExecutionId	DocumentName	AutomationExecutionStatus	ExecutionStartTime	ExecutionEndTime	
-----	-----				
-----	-----				
e1d2bad3-4567-8901-ae23-456c7c8901be	AWS-UpdateWindowsAmi	Cancelled	4/16/2019 5:37:04 AM	4/16/2019 5:47:29 AM	
61234567-a7f8-90e1-2b34-567b8bf9012c	Fixed-UpdateAmi	Cancelled	4/16/2019 5:33:04 AM	4/16/2019 5:40:15 AM	
91234d56-7e89-0ac1-2aee-34ea5d6a7c89	AWS-UpdateWindowsAmi		4/16/2019 5:22:46 AM	4/16/2019 5:27:29 AM	Failed

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAutomationExecutions](#) 中的。

Get-SSMAutomationStepExecution

以下代码示例演示了如何使用 Get-SSMAutomationStepExecution。

适用于 PowerShell V4 的工具

示例 1：此示例显示有关自动化工作流中所有活动和已终止步骤执行的信息。

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

输出：

StepName	Action	StepStatus
-----	-----	-----
LaunchInstance	aws:runInstances	Success
OSCompatibilityCheck	aws:runCommand	Success
RunPreUpdateScript	aws:runCommand	Success
UpdateEC2Config	aws:runCommand	Cancelled
UpdateSSMAgent	aws:runCommand	Pending
UpdateAWSPVDriver	aws:runCommand	Pending
UpdateAWSEnaNetworkDriver	aws:runCommand	Pending
UpdateAWSNVMe	aws:runCommand	Pending
InstallWindowsUpdates	aws:runCommand	Pending
RunPostUpdateScript	aws:runCommand	Pending
RunSysprepGeneralize	aws:runCommand	Pending

StopInstance	aws:changeInstanceState	Pending
CreateImage	aws:createImage	Pending
TerminateInstance	aws:changeInstanceState	Pending

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAutomationStepExecutions](#) 中的。

Get-SSMAvailablePatch

以下代码示例演示了如何使用 Get-SSMAvailablePatch。

适用于 PowerShell V4 的工具

示例 1：此示例获取 MSRC 严重性为“严重”的、适用于 Windows Server 2012 的所有可用补丁。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$filter1 = @{"Key"="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{"Key"="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

输出：

```
Classification : SecurityUpdates
ContentUrl      : https://support.microsoft.com/en-us/kb/2727528
Description     : A security issue has been identified that could allow an
                  unauthenticated remote attacker to compromise your system and gain control
                  over it. You can help protect your system by installing this update
                  from Microsoft. After you install this update, you may have to
                  restart your system.
Id              : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber        : KB2727528
Language        : All
MsrcNumber      : MS12-072
MsrcSeverity    : Critical
Product         : WindowsServer2012
ProductFamily   : Windows
ReleaseDate     : 11/13/2012 6:00:00 PM
Title           : Security Update for Windows Server 2012 (KB2727528)
Vendor          : Microsoft
...
```


示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个滤镜。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

示例 3：此示例提取了过去 20 天内发布且适用于与 2019 年匹配 WindowsServer 的商品的所有更新

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
Title
```

输出：

ReleaseDate	Product	Title
4/9/2019 5:00:12 PM	WindowsServer2019	2019-04 Security Update for Adobe Flash Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM	WindowsServer2019	2019-04 Cumulative Update for Windows Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM	WindowsServer2019	2019-03 Servicing Stack Update for Windows Server 2019 for x64-based Systems (KB4493510)

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeAvailablePatches](#) 中的。

Get-SSMCommand

以下代码示例演示了如何使用 Get-SSMCommand。

适用于 PowerShell V4 的工具

示例 1：此示例列出请求的所有命令。

```
Get-SSMCommand
```

输出：

```

CommandId           : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment             : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount      : 1
DocumentName        : AWS-RefreshAssociation
ErrorCount           : 0
ExpiresAfter        : 2/24/2017 3:19:08 AM
InstanceIds         : {i-0cb2b964d3e14fd9f}
MaxConcurrency       : 50
MaxErrors            : 0
NotificationConfig   : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName   :
OutputS3KeyPrefix    :
OutputS3Region       :
Parameters           : {[associationIds,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime    : 2/24/2017 3:18:08 AM
ServiceRole          :
Status               : Success
StatusDetails        : Success
TargetCount          : 1
Targets              : {}

```

示例 2：此示例获取特定命令的状态。

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

示例 3：此示例检索 2019-04-01T00:00:00Z 之后调用的所有 SSM 命令

```

Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
RequestedDateTime -Descending

```

输出：

CommandId	DocumentName	Status
RequestedDateTime		
-----	-----	-----

```

edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled 4/16/2019
5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success   4/6/2019
9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed    4/2/2019
4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed    4/1/2019
2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success   4/1/2019
1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed    4/1/2019
11:13:09 AM

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListCommands](#) 中的。

Get-SSMCommandInvocation

以下代码示例演示了如何使用 Get-SSMCommandInvocation。

适用于 PowerShell V4 的工具

示例 1：此示例列出命令的所有调用。

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
>true
```

输出：

```

CommandId           : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins      : {aws:runShellScript}
Comment             : IP config
DocumentName        : AWS-RunShellScript
InstanceId           : i-0cb2b964d3e14fd9f
InstanceName        :
NotificationConfig   : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime    : 2/22/2017 8:13:16 PM
ServiceRole         :
StandardErrorUrl     :
StandardOutputUrl    :
Status              : Success
StatusDetails        : Success

```

```
TraceOutput      :
```

示例 2：此示例列出了命令 ID e1eb2e3c-ed4c- CommandPlugins 5123-45c1-234f5612345f 的调用

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

输出：

```
Name                : aws:runPowerShellScript
Output              : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
                    remainingdownload: s3://dd-aess-r-ctmer/KUM0.png to ..\..\programdata\KUM0.png
                    kumo available

OutputS3BucketName  :
OutputS3KeyPrefix   :
OutputS3Region      : eu-west-1
ResponseCode        : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime : 4/3/2019 11:53:21 AM
StandardErrorUrl    :
StandardOutputUrl   :
Status              : Success
StatusDetails       : Success
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListCommandInvocations](#) 中的。

Get-SSMCommandInvocationDetail

以下代码示例演示了如何使用 Get-SSMCommandInvocationDetail。

适用于 PowerShell V4 的工具

示例 1：此示例显示在实例上执行的命令的详细信息。

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
"b8eac879-0541-439d-94ec-47a80d554f44"
```

输出：

```

CommandId           : b8eac879-0541-439d-94ec-47a80d554f44
Comment             : IP config
DocumentName        : AWS-RunShellScript
ExecutionElapsedTime : PT0.004S
ExecutionEndDateTime : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId           : i-0cb2b964d3e14fd9f
PluginName           : aws:runShellScript
ResponseCode         : 0
StandardErrorContent :
StandardErrorUrl      :
StandardOutputContent :
StandardOutputUrl     :
Status               : Success
StatusDetails        : Success

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetCommandInvocation](#) 中的。

Get-SSMComplianceItemList

以下代码示例演示了如何使用 Get-SSMComplianceItemList。

适用于 PowerShell V4 的工具

示例 1：此示例列出给定资源 ID 和类型的合规性项目列表，筛选合规性类型为“关联”

```

Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}

```

输出：

```

ComplianceType      : Association
Details              : {[DocumentName, AWS-GatherSoftwareInventory], [DocumentVersion,
1]}
ExecutionSummary     : Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id                   : 123a45a1-c234-1234-1245-67891236db4e
ResourceId           : i-1a2caf345f67d0dc2
ResourceType         : ManagedInstance
Severity             : UNSPECIFIED
Status               : COMPLIANT

```

Title :

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListComplianceItems](#) 中的。

Get-SSMComplianceSummaryList

以下代码示例演示了如何使用 Get-SSMComplianceSummaryList。

适用于 PowerShell V4 的工具

示例 1：此示例返回所有合规性类型的合规和不合规资源的摘要数。

```
Get-SSMComplianceSummaryList
```

输出：

```
ComplianceType CompliantSummary
NonCompliantSummary
-----
-----
FleetTotal      Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association      Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec   Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch           Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListComplianceSummaries](#) 中的。

Get-SSMConnectionStatus

以下代码示例演示了如何使用 Get-SSMConnectionStatus。

适用于 PowerShell V4 的工具

示例 1：此示例检索实例的 Session Manager 连接状态，以确定其是否已连接并准备好接收 Session Manager 连接。

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

输出：

```
Status      Target
-----
Connected i-0a1caf234f12d3dc4
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetConnectionStatus](#) 中的。

Get-SSMDefaultPatchBaseline

以下代码示例演示了如何使用 Get-SSMDefaultPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例显示默认补丁基准。

```
Get-SSMDefaultPatchBaseline
```

输出：

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetDefaultPatchBaseline](#) 中的。

Get-SSMDeployablePatchSnapshotForInstance

以下代码示例演示了如何使用 Get-SSMDeployablePatchSnapshotForInstance。

适用于 PowerShell V4 的工具

示例 1：此示例显示实例使用的补丁基准的当前快照。此命令必须使用实例凭证从实例运行。为确保其使用实例证书，该示例将 **Amazon.Runtime.InstanceProfileAWSCredentials** 对象传递给 Credentials 参数。

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
```

```
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

输出：

```
InstanceId          SnapshotDownloadUrl
-----
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

示例 2：此示例说明如何获取完整内容 SnapshotDownloadUrl。此命令必须使用实例凭证从实例运行。为确保其使用实例证书，该示例将 PowerShell 会话配置为使用 **Amazon.Runtime.InstanceProfileAWSCredentials** 对象。

```
Set-AWSCredential -Credential
([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

输出：

```
https://patch-baseline-snapshot-us-west-2.s3-us-west-2.amazonaws.com/853d0d3db0f0cafe...
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetDeployablePatchSnapshotForInstance](#) 中的。

Get-SSMDocument

以下代码示例演示了如何使用 Get-SSMDocument。

适用于 PowerShell V4 的工具

示例 1：此示例返回文档的内容。

```
Get-SSMDocument -Name "RunShellScript"
```

输出：

Content

{...}

示例 2：此示例显示文档的完整内容。

```
(Get-SSMDocument -Name "RunShellScript").Content
{
  "schemaVersion":"2.0",
  "description":"Run an updated script",
  "parameters":{"
    "commands":{"
      "type":"StringList",
      "description":"(Required) Specify a shell script or a command to run.",
      "minItems":1,
      "displayType":"textarea"
    }
  },
  "mainSteps":[
    {
      "action":"aws:runShellScript",
      "name":"runShellScript",
      "inputs":{"
        "commands":"{{ commands }}"
      }
    },
    {
      "action":"aws:runPowerShellScript",
      "name":"runPowerShellScript",
      "inputs":{"
        "commands":"{{ commands }}"
      }
    }
  ]
}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetDocument](#) 中的。

Get-SSMDocumentDescription

以下代码示例演示了如何使用 Get-SSMDocumentDescription。

适用于 PowerShell V4 的工具

示例 1：此示例返回有关文档的信息。

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

输出：

```
CreatedDate      : 2/24/2017 5:25:13 AM
DefaultVersion   : 1
Description      : Run an updated script
DocumentType     : Command
DocumentVersion  : 1
Hash             : f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion    : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion    : 2.0
Sha1            :
Status          : Active
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeDocument](#) 中的。

Get-SSMDocumentList

以下代码示例演示了如何使用 Get-SSMDocumentList。

适用于 PowerShell V4 的工具

示例 1：列出您账户中的所有配置文档。

```
Get-SSMDocumentList
```

输出：

```
DocumentType     : Command
```

```

DocumentVersion : 1
Name             : AWS-ApplyPatchBaseline
Owner            : Amazon
PlatformTypes    : {Windows}
SchemaVersion    : 1.2

DocumentType     : Command
DocumentVersion  : 1
Name             : AWS-ConfigureAWSPackage
Owner            : Amazon
PlatformTypes    : {Windows, Linux}
SchemaVersion    : 2.0

DocumentType     : Command
DocumentVersion  : 1
Name             : AWS-ConfigureCloudWatch
Owner            : Amazon
PlatformTypes    : {Windows}
SchemaVersion    : 1.2
...

```

示例 2：此示例检索名称与“Platform”匹配的所有自动化文档

```

Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
Where-Object Name -Match "Platform"

```

输出：

```

DocumentFormat   : JSON
DocumentType     : Automation
DocumentVersion  : 7
Name             : KT-Get-Platform
Owner            : 987654123456
PlatformTypes    : {Windows, Linux}
SchemaVersion    : 0.3
Tags             : {}
TargetType       :
VersionName      :

```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListDocuments](#) 中的。

Get-SSMDocumentPermission

以下代码示例演示了如何使用 Get-SSMDocumentPermission。

适用于 PowerShell V4 的工具

示例 1：此示例列出文档的所有版本。

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

输出：

CreatedDate	DocumentVersion	IsDefaultVersion	Name
-----	-----	-----	----
2/24/2017 5:25:13 AM	1	True	RunShellScript

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeDocumentPermission](#) 中的。

Get-SSMDocumentVersionList

以下代码示例演示了如何使用 Get-SSMDocumentVersionList。

适用于 PowerShell V4 的工具

示例 1：此示例列出文档的所有版本。

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

输出：

```

CreatedDate      : 6/1/2021 5:19:10 PM
DocumentFormat   : JSON
DocumentVersion  : 1
IsDefaultVersion : True
Name             : AWS-UpdateSSMAgent
Status          : Active

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListDocumentVersions](#) 中的。

Get-SSMEffectiveInstanceAssociationList

以下代码示例演示了如何使用 Get-SSMEffectiveInstanceAssociationList。

适用于 PowerShell V4 的工具

示例 1：此示例描述实例的有效关联。

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
5
```

输出：

AssociationId	Content
-----	-----
d8617c07-2079-4c18-9847-1655fc2698b0	{...

示例 2：此示例显示实例有效关联的内容。

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

输出：

```
{
  "schemaVersion": "1.2",
  "description": "Update the Amazon SSM Agent to the latest version or specified
version.",
  "parameters": {
    "version": {
      "default": "",
      "description": "(Optional) A specific version of the Amazon SSM Agent to
install. If not specified, the agen
t will be updated to the latest version.",
      "type": "String"
    },
    "allowDowngrade": {
      "default": "false",
      "description": "(Optional) Allow the Amazon SSM Agent service to be
downgraded to an earlier version. If set
to false, the service can be upgraded to newer versions only (default). If set to
true, specify the earlier version.",

```

```

        "type": "String",
        "allowedValues": [
            "true",
            "false"
        ]
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                    "agentName": "amazon-ssm-agent",
                    "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json",
                    "allowDowngrade": "{{ allowDowngrade }}",
                    "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeEffectiveInstanceAssociations](#) 中的。

Get-SSMEffectivePatchesForPatchBaseline

以下代码示例演示了如何使用 Get-SSMEffectivePatchesForPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例列出所有补丁基准，最大结果列表为 1。

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

输出：

```

Patch                                PatchStatus
-----
Amazon.SimpleSystemsManagement.Model.Patch
Amazon.SimpleSystemsManagement.Model.PatchStatus

```

示例 2：此示例显示所有补丁基准的补丁状态，最大结果列表为 1。

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

输出：

```
ApprovalDate      DeploymentStatus
-----
12/21/2010 6:00:00 PM APPROVED
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeEffectivePatchesForPatchBaseline](#) 中的。

Get-SSMInstanceAssociationsStatus

以下代码示例演示了如何使用 Get-SSMInstanceAssociationsStatus。

适用于 PowerShell V4 的工具

示例 1：此示例显示实例关联的详细信息。

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

输出：

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus     : Pending
DocumentVersion    : 1
ErrorCode          :
ExecutionDate       : 2/20/2015 8:31:11 AM
ExecutionSummary    : temp_status_change
InstanceId          : i-0000293ffd8c57862
Name                : AWS-UpdateSSMAgent
OutputUrl           :
Status             : Pending
```

示例 2：此示例检查给定实例 ID 的实例关联状态，并进一步显示这些关联的执行状态

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
{Get-SSMAssociationExecution -AssociationId .AssociationId}
```

输出：

```

AssociationId      : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion  : 2
CreatedTime        : 3/2/2019 8:53:29 AM
DetailedStatus     :
ExecutionId        : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate   : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status             : Success

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeInstanceAssociationsStatus](#) 中的。

Get-SSMInstanceInformation

以下代码示例演示了如何使用 Get-SSMInstanceInformation。

适用于 PowerShell V4 的工具

示例 1：此示例显示每个实例的详细信息。

```
Get-SSMInstanceInformation
```

输出：

```

ActivationId      :
AgentVersion      : 2.0.672.0
AssociationOverview :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus  : Success
ComputerName      : ip-172-31-44-222.us-west-2.compute.internal
IamRole           :
InstanceId        : i-0cb2b964d3e14fd9f
IPAddress         : 172.31.44.222
IsLatestVersion    : True
LastAssociationExecutionDate : 2/24/2017 3:18:09 AM
LastPingDateTime   : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name              :
PingStatus        : ConnectionLost
PlatformName      : Amazon Linux AMI

```



```
PlatformType           : Linux
PlatformVersion        : 2016.09
RegistrationDate        : 1/1/0001 12:00:00 AM
ResourceType           : EC2Instance
```

示例 2：此示例说明如何使用 `-Filter` 参数将结果筛选到区域 **us-east-1** 中仅为 **AgentVersion** 的 **2.2.800.0** 那些 AWS 系统管理器实例。你可以在 `InstanceInformation` API 参考主题 (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type--InstanceInformation) 中找到有效的 `-Filter` 键值列表。ActivationId

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

输出：

```
ActivationId           :
AgentVersion           : 2.2.800.0
AssociationOverview    :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus      : Success
ComputerName           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                :
InstanceId             : i-EXAMPLEb0792d98ce
IPAddress              : 10.0.0.01
IsLatestVersion        : False
LastAssociationExecutionDate : 8/16/2018 12:02:50 AM
LastPingDateTime       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                   :
PingStatus             : Online
PlatformName           : Microsoft Windows Server 2016 Datacenter
PlatformType           : Windows
PlatformVersion        : 10.0.14393
RegistrationDate        : 1/1/0001 12:00:00 AM
ResourceType           : EC2Instance

ActivationId           :
AgentVersion           : 2.2.800.0
```

```

AssociationOverview           :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus              : Success
ComputerName                   : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                        :
InstanceId                     : i-EXAMPLEac7501d023
IPAddress                      : 10.0.0.02
IsLatestVersion                : False
LastAssociationExecutionDate    : 8/16/2018 12:00:20 AM
LastPingDateTime               : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                           :
PingStatus                     : Online
PlatformName                   : Microsoft Windows Server 2016 Datacenter
PlatformType                   : Windows
PlatformVersion                : 10.0.14393
RegistrationDate                : 1/1/0001 12:00:00 AM
ResourceType                    : EC2Instance

```

示例 3：此示例说明如何使用 `-InstanceInformationFilterList` 参数筛选结果，仅筛选区域中 **us-east-1** 带 **Windows** 或 **PlatformTypes** 的那些 AWS 系统管理器实例 **Linux**。您可以在 `InstanceInformationFilter` API 参考主题 (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html) 中找到有效的 `InstanceInformationFilterList` 键值列表。

```

$Filters = @{
    Key="PlatformTypes"
    ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters

```

输出：

```

ActivationId                   :
AgentVersion                   : 2.2.800.0
AssociationOverview            :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus               : Success
ComputerName                   : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                        :
InstanceId                     : i-EXAMPLEeb0792d98ce
IPAddress                      : 10.0.0.27

```

```

IsLatestVersion           : False
LastAssociationExecutionDate : 8/16/2018 12:02:50 AM
LastPingDateTime          : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                      :
PingStatus                : Online
PlatformName              : Ubuntu Server 18.04 LTS
PlatformType              : Linux
PlatformVersion           : 18.04
RegistrationDate           : 1/1/0001 12:00:00 AM
ResourceType              : EC2Instance

ActivationId              :
AgentVersion              : 2.2.800.0
AssociationOverview       :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus         : Success
ComputerName              : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                   :
InstanceId                 : i-EXAMPLEac7501d023
IPAddress                 : 10.0.0.100
IsLatestVersion           : False
LastAssociationExecutionDate : 8/16/2018 12:00:20 AM
LastPingDateTime          : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                      :
PingStatus                : Online
PlatformName              : Microsoft Windows Server 2016 Datacenter
PlatformType              : Windows
PlatformVersion           : 10.0.14393
RegistrationDate           : 1/1/0001 12:00:00 AM
ResourceType              : EC2Instance

```

示例 4：此示例列出了 ssm 托管实例 InstanceId，并导出 PlatformName 到 csv 文件 PingStatus、LastPingDateTime 和。

```

Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
PlatformName | Export-Csv Instance-details.csv -NoTypeInfo

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeInstanceInformation](#) 中的。

Get-SSMInstancePatch

以下代码示例演示了如何使用 Get-SSMInstancePatch。

适用于 PowerShell V4 的工具

示例 1：此示例获取实例的补丁合规性详细信息。

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeInstancePatches](#) 中的。

Get-SSMInstancePatchState

以下代码示例演示了如何使用 Get-SSMInstancePatchState。

适用于 PowerShell V4 的工具

示例 1：此示例获取实例的补丁摘要状态。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

示例 2：此示例获取两个实例的补丁摘要状态。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeInstancePatchStates](#) 中的。

Get-SSMInstancePatchStatesForPatchGroup

以下代码示例演示了如何使用 Get-SSMInstancePatchStatesForPatchGroup。

适用于 PowerShell V4 的工具

示例 1：此示例获取补丁组每个实例的补丁摘要状态。

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeInstancePatchStatesForPatchGroup](#) 中的。

Get-SSMInventory

以下代码示例演示了如何使用 Get-SSMInventory。

适用于 PowerShell V4 的工具

示例 1：此示例获取清单的自定义元数据。

```
Get-SSMInventory
```

输出：

```
Data
  Id
----
--
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetInventory](#) 中的。

Get-SSMInventoryEntriesList

以下代码示例演示了如何使用 Get-SSMInventoryEntriesList。

适用于 PowerShell V4 的工具

示例 1：此示例列出实例的所有自定义清单条目。

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo"
```

输出：

```
CaptureTime    : 2016-08-22T10:01:01Z
Entries        :
{Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
```

```
InstanceId      : i-0cb2b964d3e14fd9f
NextToken       :
SchemaVersion   : 1.0
TypeName        : Custom:RackInfo
```

示例 2：此示例列出详细信息。

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo").Entries
```

输出：

```
Key           Value
---           -
RackLocation  Bay B/Row C/Rack D/Shelf E
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListInventoryEntries](#) 中的。

Get-SSMInventoryEntryList

以下代码示例演示了如何使用 Get-SSMInventoryEntryList。

适用于 PowerShell V4 的工具

示例 1：此示例检索实例的 **AWS:Network** 类型清单条目。

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
Select-Object -ExpandProperty Entries
```

输出：

```
Key           Value
---           -
DHCPServer    172.31.11.2
DNSServer     172.31.0.1
Gateway       172.31.11.2
IPV4          172.31.11.222
IPV6          fe12::3456:7da8:901a:12a3
MacAddress    1A:23:4E:5B:FB:67
Name          Amazon Elastic Network Adapter
```

```
SubnetMask 255.255.240.0
```

- 有关 API 的详细信息，请参阅 [Get-SSMInventoryEntryList in AWS Tools for PowerShell Cmdlet 参考 \(V4\)](#)。

Get-SSMInventorySchema

以下代码示例演示了如何使用 Get-SSMInventorySchema。

适用于 PowerShell V4 的工具

示例 1：此示例返回账户清单类型名称列表。

```
Get-SSMInventorySchema
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetInventorySchema](#) 中的。

Get-SSMLatestEC2Image

以下代码示例演示了如何使用 Get-SSMLatestEC2Image。

适用于 PowerShell V4 的工具

示例 1：此示例列出了所有最新的 Windows AMIs。

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

输出：

Name	Value
----	-----
Windows_Server-2008-R2_SP1-English-64Bit-SQL_2012_SP4_Express ami-0e5ddd288daff4fab	
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base ami-0c5ea64e6bec1cb50	
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base ami-09775eff0bf8c113d	
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base ami-025064b67e28cf5df	
...	

示例 2：此示例检索 us-west-2 区域的特定 Amazon Linux 映像的 AMI ID。

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

输出：

```
ami-09b92cd132204c704
```

示例 3：此示例列出了所有与指定通配符表达式 AMIs 匹配的最新窗口。

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

输出：

Name	Value
----	-----
Windows_Server-2019-English-Full-SQL_2017_Web	ami-085e9d27da5b73a42
Windows_Server-2019-English-STIG-Core	ami-0bfd85c29148c7f80
Windows_Server-2019-English-Full-SQL_2019_Web	ami-02099560d7fb11f20
Windows_Server-2019-English-Full-SQL_2016_SP2_Standard	ami-0d7ae2d81c07bd598
...	

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考文档 (V 4) 中的[获取 SSMLatestEC2 图片](#)。

Get-SSMMaintenanceWindow

以下代码示例演示了如何使用 Get-SSMMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例获取有关维护时段的详细信息。

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

输出：

```
AllowUnassociatedTargets : False
CreatedDate               : 2/20/2017 6:14:05 PM
```



```

Cutoff           : 1
Duration         : 2
Enabled          : True
ModifiedDate     : 2/20/2017 6:14:05 PM
Name             : TestMaintWin
Schedule         : cron(0 */30 * * * ? *)
WindowId         : mw-03eb9db42890fb82d

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetMaintenanceWindow](#) 中的。

Get-SSMMaintenanceWindowExecution

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowExecution。

适用于 PowerShell V4 的工具

示例 1：此示例列出有关作为维护时段执行组成部分来执行的任务的信息。

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

输出：

```

EndTime          : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status           : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds          : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) GetMaintenanceWindowExecution](#) 中的。

Get-SSMMaintenanceWindowExecutionList

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowExecutionList。

适用于 PowerShell V4 的工具

示例 1：此示例列出维护时段的所有执行。

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

输出：

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime          : 2/20/2017 6:30:16 PM
Status             : FAILED
StatusDetails      : One or more tasks in the orchestration failed.
WindowExecutionId  : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId           : mw-03eb9db42890fb82d
```

示例 2：此示例列出在指定日期之前的维护时段内的所有执行。

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

示例 3：此示例列出指定日期之后维护时段内的所有执行。

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindowExecutions](#) 中的。

Get-SSMMaintenanceWindowExecutionTask

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowExecutionTask。

适用于 PowerShell V4 的工具

示例 1：此示例列出有关维护时段执行组成部分的任务的信息。

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
-WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

输出：

```
EndTime           : 2/21/2017 4:00:35 PM
```

```

MaxConcurrency      : 1
MaxErrors           : 1
Priority             : 10
ServiceRole         : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime           : 2/21/2017 4:00:34 PM
Status              : FAILED
StatusDetails       : The maximum error count was exceeded.
TaskArn             : AWS-RunShellScript
TaskExecutionId     : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters      :
    {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsManag
        meterValueExpression]}
Type                : RUN_COMMAND
WindowExecutionId   : 518d5565-5969-4cca-8f0e-da3b2a638355

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetMaintenanceWindowExecutionTask](#) 中的。

Get-SSMMaintenanceWindowExecutionTaskInvocationList

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowExecutionTaskInvocationList。

适用于 PowerShell V4 的工具

示例 1：此示例列出作为维护时段执行组成部分来执行的任务的调用。

```

Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"

```

输出：

```

EndTime            : 2/21/2017 4:00:34 PM
ExecutionId        :
InvocationId       : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation   :
Parameters         : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"],"maxConcurrency":"1",
    "maxErrors":"1"}}
StartTime          : 2/21/2017 4:00:34 PM
Status             : FAILED
StatusDetails      : The instance IDs list contains an invalid entry.

```

```
TaskExecutionId    : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId  : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId     :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindowExecutionTaskInvocations](#) 中的。

Get-SSMMaintenanceWindowExecutionTaskList

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowExecutionTaskList。

适用于 PowerShell V4 的工具

示例 1：此示例列出与维护时段执行相关的任务。

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
"518d5565-5969-4cca-8f0e-da3b2a638355"
```

输出：

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindowExecutionTasks](#) 中的。

Get-SSMMaintenanceWindowList

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowList。

适用于 PowerShell V4 的工具

示例 1：此示例列出您账户中的所有维护时段。

```
Get-SSMMaintenanceWindowList
```

输出：

```
Cutoff      : 1
Duration    : 4
Enabled     : True
Name        : My-First-Maintenance-Window
WindowId    : mw-06d59c1a07c022145
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindows](#) 中的。

Get-SSMMaintenanceWindowTarget

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowTarget。

适用于 PowerShell V4 的工具

示例 1：此示例列出维护时段的所有目标。

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

输出：

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindowTargets](#) 中的。

Get-SSMMaintenanceWindowTaskList

以下代码示例演示了如何使用 Get-SSMMaintenanceWindowTaskList。

适用于 PowerShell V4 的工具

示例 1：此示例列出维护时段的所有任务。

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

输出：

```
LoggingInfo      :
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRoleArn    : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets           : {InstanceIds}
TaskArn           : AWS-RunShellScript
TaskParameters    : {[commands,
  Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type              : RUN_COMMAND
WindowId          : mw-06cf17cbefcb4bf4f
WindowTaskId      : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeMaintenanceWindowTasks](#) 中的。

Get-SSMParameterHistory

以下代码示例演示了如何使用 Get-SSMParameterHistory。

适用于 PowerShell V4 的工具

示例 1：此示例列出参数的值历史记录。

```
Get-SSMParameterHistory -Name "Welcome"
```

输出：

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
```

```
Type      : String
Value     : helloWorld
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetParameterHistory](#) 中的。

Get-SSMParameterList

以下代码示例演示了如何使用 Get-SSMParameterList。

适用于 PowerShell V4 的工具

示例 1：此示例列出所有参数。

```
Get-SSMParameterList
```

输出：

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type            : String
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeParameters](#) 中的。

Get-SSMParameterValue

以下代码示例演示了如何使用 Get-SSMParameterValue。

适用于 PowerShell V4 的工具

示例 1：此示例列出参数的值。

```
Get-SSMParameterValue -Name "Welcome"
```

输出：

```
InvalidParameters Parameters
-----
{}                {Welcome}
```

示例 2：此示例列出值的详细信息。

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

输出：

```
Name      Type      Value
----      -
Welcome String Good day, Sunshine!
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [GetParameters](#) 中的。

Get-SSMPatchBaseline

以下代码示例演示了如何使用 Get-SSMPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例列出所有补丁基准。

```
Get-SSMPatchBaseline
```

输出：

```
BaselineDescription                                     BaselineId
-----
Default Patch Baseline Provided by AWS.                arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
Production-B...
```


示例 2：此示例列出了提供的所有补丁基准。AWS 此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

输出：

```
Get-SSMPatchBaseline -Filter $filter1
```

示例 3：此示例列出您作为所有者的所有补丁基准。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

输出：

```
Get-SSMPatchBaseline -Filter $filter1
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个标签。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

输出：

BaselineDescription	BaselineName	BaselineId	DefaultBaseline
Default Patch Baseline Provided by AWS.	arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966	AWS-DefaultPatchBaseline	True

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribePatchBaselines](#) 中的。

Get-SSMPatchBaselineDetail

以下代码示例演示了如何使用 Get-SSMPatchBaselineDetail。

适用于 PowerShell V4 的工具

示例 1：此示例显示补丁基准的详细信息。

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

输出：

```
ApprovalRules      : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches    : {}
BaselineId         : pb-03da896ca3b68b639
CreatedDate        : 3/3/2017 5:02:19 PM
Description         : Baseline containing all updates approved for production systems
GlobalFilters      : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate       : 3/3/2017 5:02:19 PM
Name               : Production-Baseline
PatchGroups        : {}
RejectedPatches    : {}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetPatchBaseline](#) 中的。

Get-SSMPatchBaselineForPatchGroup

以下代码示例演示了如何使用 Get-SSMPatchBaselineForPatchGroup。

适用于 PowerShell V4 的工具

示例 1：此示例显示补丁组的补丁基准。

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

输出：

BaselineId	PatchGroup
-----	-----
pb-045f10b4f382baeda	Production

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [GetPatchBaselineForPatchGroup](#) 中的。

Get-SSMPatchGroup

以下代码示例演示了如何使用 Get-SSMPatchGroup。

适用于 PowerShell V4 的工具

示例 1：此示例列出补丁组注册。

```
Get-SSMPatchGroup
```

输出：

BaselineIdentity	PatchGroup
-----	-----
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity	Production

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribePatchGroups](#) 中的。

Get-SSMPatchGroupState

以下代码示例演示了如何使用 Get-SSMPatchGroupState。

适用于 PowerShell V4 的工具

示例 1：此示例获取补丁组的高级补丁合规性摘要。

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

输出：

Instances	: 4
InstancesWithFailedPatches	: 1
InstancesWithInstalledOtherPatches	: 4
InstancesWithInstalledPatches	: 3
InstancesWithMissingPatches	: 0
InstancesWithNotApplicablePatches	: 0

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribePatchGroupState](#) 中的。

Get-SSMResourceComplianceSummaryList

以下代码示例演示了如何使用 Get-SSMResourceComplianceSummaryList。

适用于 PowerShell V4 的工具

示例 1：此示例获取资源级摘要计数。摘要包含有关合规和不合规状态的信息以及与“Windows10”匹配产品的详细合规性项目严重性计数。由于如果未指定参数，则 MaxResult 默认值为 100，且此值无效，因此会添加 MaxResult 参数，并将该值设置为 50。

```
$FilterValues = @{
    "Key"="Product"
    "Type"="EQUAL"
    "Values"="Windows10"
}

Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) ListResourceComplianceSummaries](#) 中的。

Get-SSMResourceTag

以下代码示例演示了如何使用 Get-SSMResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例列出维护时段的标签。

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow"
```

输出：

```
Key    Value
---    -
Stack  Production
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ListTagsForResource](#) 中的。

New-SSMActivation

以下代码示例演示了如何使用 New-SSMActivation。

适用于 PowerShell V4 的工具

示例 1：此示例创建托管式实例。

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

输出：

```
ActivationCode      ActivationId
-----
KWChh0xBTiWdCKE9B1KC 08e51e79-1e36-446c-8e63-9458569c1363
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateActivation](#) 中的。

New-SSMAssociation

以下代码示例演示了如何使用 New-SSMAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例使用实例将配置文档与实例关联起来 IDs。

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

输出：

```
Name           : AWS-UpdateSSMAgent
InstanceId      : i-0000293ffd8c57862
Date           : 2/23/2017 6:55:22 PM
Status.Name     : Associated
```

```
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

示例 2：此示例使用目标将配置文档与实例关联起来。

```
$target = @{{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

输出：

```
Name           : AWS-UpdateSSMAgent
InstanceId      :
Date            : 3/1/2017 6:22:21 PM
Status.Name     :
Status.Date     :
Status.Message  :
Status.AdditionalInfo :
```

示例 3：此示例使用目标和参数将配置文档与实例关联起来。

```
$target = @{{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}}
$params = @{"action"="configure"
            "mode"="ec2"
            "optionalConfigurationSource"="ssm"
            "optionalConfigurationLocation"=""
            "optionalRestart"="yes"
            }
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName "CWConfiguration" -
Target $target -Parameter $params
```

输出：

```
Name           : Configure-CloudWatch
InstanceId      :
Date            : 5/17/2018 3:17:44 PM
Status.Name     :
Status.Date     :
Status.Message  :
Status.AdditionalInfo :
```

示例 4：此示例使用 **AWS-GatherSoftwareInventory** 创建了与该区域中所有实例的关联。它还在要收集的参数中提供自定义文件和注册表位置

```
$params =
[Collections.Generic.Dictionary[String,Collections.Generic.List[String]]::new()
$params["windowsRegistry"] = '[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
-Target @{Key="instanceids";Values=""} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

输出：

```
Name                : AWS-GatherSoftwareInventory
InstanceId           :
Date                 : 6/9/2019 8:57:56 AM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreateAssociation](#) 中的。

New-SSMAssociationFromBatch

以下代码示例演示了如何使用 New-SSMAssociationFromBatch。

适用于 PowerShell V4 的工具

示例 1：此示例将一个配置文档与多个实例相关联。如果适用，输出将返回成功和失败操作的列表。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

输出：

```
Failed    Successful
-----
{}        {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

示例 2：此示例将显示成功操作的完整详细信息。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) CreateAssociationBatch](#) 中的。

New-SSMDocument

以下代码示例演示了如何使用 New-SSMDocument。

适用于 PowerShell V4 的工具

示例 1：此示例在您的账户中创建一个文档。该文档必须采用 JSON 格式。有关编写配置文档的更多信息，请参阅《SSM API Reference》中的配置文档。

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
"RunShellScript" -DocumentType "Command"
```

输出：

```
CreatedDate      : 3/1/2017 1:21:33 AM
DefaultVersion   : 1
Description      : Run an updated script
DocumentType     : Command
DocumentVersion  : 1
Hash             : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion    : 1
Name             : RunShellScript
Owner           : 809632081692
Parameters       : {commands}
PlatformTypes    : {Linux}
SchemaVersion    : 2.0
```



```
Sha1      :  
Status    : Creating
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateDocument](#) 中的。

New-SSMMaintenanceWindow

以下代码示例演示了如何使用 New-SSMMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例使用指定名称创建了一个新的维护时段，该时段在每周二下午 4 点运行 4 小时，中断 1 小时，且允许未关联的目标。

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -  
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

输出：

```
mw-03eb53e1ea7383998
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateMaintenanceWindow](#) 中的。

New-SSMPatchBaseline

以下代码示例演示了如何使用 New-SSMPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例创建补丁基准，用于在 Microsoft 发布补丁 7 天后为在生产环境中运行 Windows Server 2019 的托管式实例批准补丁。

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule  
$rule.ApproveAfterDays = 7  
  
$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup  
  
$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
```

```
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description "Baseline
containing all updates approved for production systems" -ApprovalRules_PatchRule
$rule
```

输出：

```
pb-0z4z6221c4296b23z
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [CreatePatchBaseline](#) 中的。

Register-SSMDefaultPatchBaseline

以下代码示例演示了如何使用 Register-SSMDefaultPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例将补丁基准注册为默认补丁基准。

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

输出：

```
pb-03da896ca3b68b639
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterDefaultPatchBaseline](#) 中的。

Register-SSMPatchBaselineForPatchGroup

以下代码示例演示了如何使用 Register-SSMPatchBaselineForPatchGroup。

适用于 PowerShell V4 的工具

示例 1：此示例为补丁组注册补丁基准。

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

输出：

```
BaselineId          PatchGroup
-----
pb-03da896ca3b68b639 Production
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterPatchBaselineForPatchGroup](#) 中的。

Register-SSMTargetWithMaintenanceWindow

以下代码示例演示了如何使用 Register-SSMTargetWithMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例向维护时段注册实例。

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

输出：

```
d8e47760-23ed-46a5-9f28-927337725398
```

示例 2：此示例向维护时段注册多个实例。

```
$option1 =  
@{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target  
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

输出：

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

示例 3：此示例使用 EC2 标签在维护时段注册实例。

```
$option1 = @{Key="tag:Environment";Values=@("Production")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target  
$option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

输出：

```
2994977e-aefb-4a71-beac-df620352f184
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [RegisterTargetWithMaintenanceWindow](#) 中的。

Register-SSMTaskWithMaintenanceWindow

以下代码示例演示了如何使用 Register-SSMTaskWithMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例使用实例 ID 向维护时段注册任务。输出为任务 ID。

```
$parameters = @{}  
$parameterValues = New-Object  
Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression  
$parameterValues.Values = @("Install")  
$parameters.Add("Operation", $parameterValues)
```

```
Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
-ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
-MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
@{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

输出：

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

示例 2：此示例使用目标 ID 向维护时段注册任务。输出为任务 ID。

```
$parameters = @{}
$parameterValues = New-Object
Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
-ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
-MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
@{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -TaskType
"RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

输出：

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

示例 3：此示例为运行命令文档 **AWS-RunPowerShellScript** 创建参数对象，并使用目标 ID 创建具有给定维护时段的任务。返回的输出是任务 ID。

```
$parameters =
[Collections.Generic.Dictionary[String,Collections.Generic.List[String]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
```

```

TaskType = "RUN_COMMAND"
TaskArn = "AWS-RunPowerShellScript"
Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
Priority = 1
RunCommand_Parameter = $parameters
Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props

```

输出：

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

示例 4：此示例使用名为的文档注册 AWS Systems Manager 自动化任务**Create-Snapshots**。

```

$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{ TARGET_ID }") )
$automationParameters.Add( "AutomationAssumeRole",
    @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
$automationParameters -Name "Create-Snapshots"

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RegisterTaskWithMaintenanceWindow](#) 中的。

Remove-SSMActivation

以下代码示例演示了如何使用 Remove-SSMActivation。

适用于 PowerShell V4 的工具

示例 1：此示例删除激活。如果此命令成功，则无任何输出。

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteActivation](#) 中的。

Remove-SSMAssociation

以下代码示例演示了如何使用 Remove-SSMAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例删除实例和文档之间的关联。如果此命令成功，则无任何输出。

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteAssociation](#) 中的。

Remove-SSMDocument

以下代码示例演示了如何使用 Remove-SSMDocument。

适用于 PowerShell V4 的工具

示例 1：此示例删除文档。如果此命令成功，则无任何输出。

```
Remove-SSMDocument -Name "RunShellScript"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteDocument](#) 中的。

Remove-SSMMaintenanceWindow

以下代码示例演示了如何使用 Remove-SSMMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例删除维护时段。

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

输出：

```
mw-06d59c1a07c022145
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteMaintenanceWindow](#) 中的。

Remove-SSMParameter

以下代码示例演示了如何使用 Remove-SSMParameter。

适用于 PowerShell V4 的工具

示例 1：此示例删除一个参数。如果此命令成功，则无任何输出。

```
Remove-SSMParameter -Name "helloWorld"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteParameter](#) 中的。

Remove-SSMPatchBaseline

以下代码示例演示了如何使用 Remove-SSMPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例删除补丁基准。

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

输出：

```
pb-045f10b4f382baeda
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeletePatchBaseline](#) 中的。

Remove-SSMResourceTag

以下代码示例演示了如何使用 Remove-SSMResourceTag。

适用于 PowerShell V4 的工具

示例 1：此示例从维护时段中删除标签。如果此命令成功，则无任何输出。

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow" -TagKey "Production"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RemoveTagsFromResource](#) 中的。

Send-SSMCommand

以下代码示例演示了如何使用 Send-SSMCommand。

适用于 PowerShell V4 的工具

示例 1：此示例在目标实例上运行 echo 命令。

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
"echo helloWorld"} -Target @{Key="instanceids";Values=@"i-0cb2b964d3e14fd9f"}}
```

输出：

```
CommandId           : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment             :
CompletedCount      : 0
DocumentName        : AWS-RunPowerShellScript
ErrorCount          : 0
ExpiresAfter        : 3/7/2017 10:48:37 PM
InstanceIds         : {}
MaxConcurrency      : 50
MaxErrors           : 0
NotificationConfig  : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName  :
OutputS3KeyPrefix   :
OutputS3Region      :
Parameters          : {[commands,
    Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

```
RequestedDateTime : 3/7/2017 9:48:37 PM
ServiceRole       :
Status            : Pending
StatusDetails     : Pending
TargetCount       : 0
Targets           : {instanceids}
```

示例 2：此示例展示如何运行接受嵌套参数的命令。

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
@{ sourceType="GitHub";sourceInfo='{ "owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32openSSH"}'; "commandLine"=".\\Install-
Win32openSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [SendCommand](#) 中的。

Start-SSMAutomationExecution

以下代码示例演示了如何使用 Start-SSMAutomationExecution。

适用于 PowerShell V4 的工具

示例 1：此示例运行一个文档，指定自动化角色、AMI 源 ID 和 Amazon EC2 实例角色。

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{ 'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole'; 'SourceAmiId'='ami-f173cc91'; 'InstanceIamRole'='EC2InstanceRole' }
```

输出：

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartAutomationExecution](#) 中的。

Start-SSMSession

以下代码示例演示了如何使用 Start-SSMSession。

适用于 PowerShell V4 的工具

示例 1：此示例为会话管理器会话启动与目标的连接，从而启用端口转发。

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-  
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =  
'80' }
```

输出：

SessionId	StreamUrl
-----	-----
random-id0	wss://ssmmessages.amazonaws.com/v1/data-channel/random-id

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StartSession](#) 中的。

Stop-SSMAutomationExecution

以下代码示例演示了如何使用 Stop-SSMAutomationExecution。

适用于 PowerShell V4 的工具

示例 1：此示例停止自动化执行。如果此命令成功，则无任何输出。

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-  
f944-11e6-9d32-8fb2db27a909"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopAutomationExecution](#) 中的。

Stop-SSMCommand

以下代码示例演示了如何使用 Stop-SSMCommand。

适用于 PowerShell V4 的工具

示例 1：此示例尝试取消命令。如果操作成功，则无任何输出。

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CancelCommand](#) 中的。

Unregister-SSMManagedInstance

以下代码示例演示了如何使用 Unregister-SSMManagedInstance。

适用于 PowerShell V4 的工具

示例 1：此示例取消注册托管式实例。如果此命令成功，则无任何输出。

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeregisterManagedInstance](#) 中的。

Unregister-SSMPatchBaselineForPatchGroup

以下代码示例演示了如何使用 Unregister-SSMPatchBaselineForPatchGroup。

适用于 PowerShell V4 的工具

示例 1：此示例从补丁基准中取消注册补丁组。

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -  
PatchGroup "Production"
```

输出：

BaselineId	PatchGroup
-----	-----
pb-045f10b4f382baeda	Production

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeregisterPatchBaselineForPatchGroup](#) 中的。

Unregister-SSMTargetFromMaintenanceWindow

以下代码示例演示了如何使用 Unregister-SSMTargetFromMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例从维护时段中删除目标。

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

输出：

WindowId	WindowTargetId
-----	-----
mw-06cf17cbefcb4bf4f	6ab5c208-9fc4-4697-84b7-b02a6cc25f7d

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeregerTargetFromMaintenanceWindow](#) 中的。

Unregister-SSMTaskFromMaintenanceWindow

以下代码示例演示了如何使用 Unregister-SSMTaskFromMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例从维护时段中删除任务。

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

输出：

WindowId	WindowTaskId
-----	-----
mw-03a342e62c96d31b0	f34a2c47-ddfd-4c85-a88d-72366b69af1b

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DeregerTaskFromMaintenanceWindow](#) 中的。

Update-SSMAssociation

以下代码示例演示了如何使用 Update-SSMAssociation。

适用于 PowerShell V4 的工具

示例 1：此示例使用新文档版本更新关联。

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

输出：

```
Name           : AWS-UpdateSSMAgent
InstanceId      :
Date           : 3/1/2017 6:22:21 PM
Status.Name     :
Status.Date     :
Status.Message  :
Status.AdditionalInfo :
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateAssociation](#) 中的。

Update-SSMAssociationStatus

以下代码示例演示了如何使用 Update-SSMAssociationStatus。

适用于 PowerShell V4 的工具

示例 1：此示例更新实例与配置文档之间关联的关联状态。

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
"i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
-AssociationStatus_Name "Pending" -AssociationStatus_Message
"temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

输出：

```
Name           : AWS-UpdateSSMAgent
InstanceId      : i-0000293ffd8c57862
Date           : 2/23/2017 6:55:22 PM
Status.Name     : Pending
Status.Date     : 2/20/2015 8:31:11 AM
Status.Message  : temporary_status_change
```

```
Status.AdditionalInfo : Additional-Config-Needed
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateAssociationStatus](#) 中的。

Update-SSMDocument

以下代码示例演示了如何使用 Update-SSMDocument。

适用于 PowerShell V4 的工具

示例 1：此示例将使用您指定的 json 文件更新内容创建文档的新版本。该文档必须采用 JSON 格式。您可以使用 “Get-SSMDocumentVersionList” cmdlet 获取文档版本。

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -Raw "c:\temp\RunShellScript.json")
```

输出：

```

CreatedDate      : 3/1/2017 2:59:17 AM
DefaultVersion   : 1
Description      : Run an updated script
DocumentType     : Command
DocumentVersion  : 2
Hash             : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion    : 2
Name             : RunShellScript
Owner           : 809632081692
Parameters       : {commands}
PlatformTypes    : {Linux}
SchemaVersion    : 2.0
Sha1             :
Status           : Updating

```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateDocument](#) 中的。

Update-SSMDocumentDefaultVersion

以下代码示例演示了如何使用 Update-SSMDocumentDefaultVersion。

适用于 PowerShell V4 的工具

示例 1：此实例更新文档的默认版本。您可以通过“Get-SSMDocumentVersionList”cmdlet 获取可用的文档版本。

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

输出：

```
DefaultVersion Name
-----
2              RunShellScript
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateDocumentDefaultVersion](#) 中的。

Update-SSMMaintenanceWindow

以下代码示例演示了如何使用 Update-SSMMaintenanceWindow。

适用于 PowerShell V4 的工具

示例 1：此示例更新维护时段的名称。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

输出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

示例 2：此示例启用维护时段。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```


输出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

示例 3：此示例禁用维护时段。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

输出：

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateMaintenanceWindow](#) 中的。

Update-SSMManagedInstanceRole

以下代码示例演示了如何使用 Update-SSMManagedInstanceRole。

适用于 PowerShell V4 的工具

示例 1：此示例更新托管式实例的角色。如果此命令成功，则无任何输出。

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
"AutomationRole"
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdateManagedInstanceRole](#) 中的。

Update-SSMPatchBaseline

以下代码示例演示了如何使用 Update-SSMPatchBaseline。

适用于 PowerShell V4 的工具

示例 1：此示例将两个补丁（作为已拒绝的补丁）和一个补丁（作为已批准的补丁）添加到现有补丁基准。

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch  
"KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

输出：

```
ApprovalRules      : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup  
ApprovedPatches    : {KB2124261}  
BaselineId         : pb-03da896ca3b68b639  
CreatedDate        : 3/3/2017 5:02:19 PM  
Description         : Baseline containing all updates approved for production systems  
GlobalFilters       : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup  
ModifiedDate        : 3/3/2017 5:22:10 PM  
Name                : Production-Baseline  
RejectedPatches    : {KB2032276, MS10-048}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [UpdatePatchBaseline](#) 中的。

Write-SSMComplianceItem

以下代码示例演示了如何使用 Write-SSMComplianceItem。

适用于 PowerShell V4 的工具

示例 1：此示例为给定的托管式实例编写自定义合规性项目

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()  
$item.Id = "07Jun2019-3"  
$item.Severity="LOW"  
$item.Status="COMPLIANT"  
$item.Title="Fin-test-1 - custom"
```

```
Write-SSMComplianceItem -ResourceId mi-012dc3ecea45b678 -ComplianceType
Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutComplianceItems](#) 中的。

Write-SSMInventory

以下代码示例演示了如何使用 Write-SSMInventory。

适用于 PowerShell V4 的工具

示例 1：此示例将机架位置信息分配给某个实例。如果此命令成功，则无任何输出。

```
$data = New-Object
"System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
"System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
System.String]]"
$items.Add($data)

$customInventoryItem = New-Object Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutInventory](#) 中的。

Write-SSMParameter

以下代码示例演示了如何使用 Write-SSMParameter。

适用于 PowerShell V4 的工具

示例 1：此示例创建一个参数。如果此命令成功，则无任何输出。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

示例 2：此示例更改了参数。如果此命令成功，则无任何输出。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -  
Overwrite $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [PutParameter](#) 中的。

使用适用于 PowerShell V4 的工具的 Amazon Translate 示例

以下代码示例向您展示了如何使用带有 Amazon Translate 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

ConvertTo-TRNTargetLanguage

以下代码示例演示了如何使用 ConvertTo-TRNTargetLanguage。

适用于 PowerShell V4 的工具

示例 1：将指定的英文文本转换为法语。要转换的文本也可以作为 -Text 参数传递。

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -  
TargetLanguageCode fr
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [TranslateText](#) 中的。

AWS WAFV2 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 AWS WAFV2 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

New-WAF2WebACL

以下代码示例演示了如何使用 New-WAF2WebACL。

适用于 PowerShell V4 的工具

示例 1：此命令创建一个名为“waf-test”的新 Web ACL。请注意，根据服务 API 文档，“DefaultAction”是必填属性。因此，应指定'-_All DefaultAction ow' '-DefaultAction _ and/or Block' 的值。由于'-DefaultAction _Allow'和'-DefaultAction _Block'不是必需的属性，因此可以将值 '@ {}' 用作占位符，如上例所示。

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-  
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -  
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-  
test" -Description "Test" -DefaultAction-Allow @{}
```

输出：

```
ARN          : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-  
test/19460b3f-db14-4b9a-8e23-a417e1eb007f  
Description  : Test  
Id           : 19460b3f-db14-4b9a-8e23-a417e1eb007f  
LockToken    : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab  
Name         : waf-test
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateWebAcl](#) 中的。

WorkSpaces 使用适用于 PowerShell V4 的工具的示例

以下代码示例向您展示了如何使用带 WorkSpaces 的 AWS Tools for PowerShell V4 来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

Approve-WKSIpRule

以下代码示例演示了如何使用 Approve-WKSIpRule。

适用于 PowerShell V4 的工具

示例 1：此示例向现有 IP 组添加规则

```
$Rule = @(  
@{IPRule = "10.1.0.0/0"; RuleDesc = "First Rule Added"},  
@{IPRule = "10.2.0.0/0"; RuleDesc = "Second Rule Added"})
```

```
)  
  
Approve-WKSipRule -GroupId wsipg-abcnx2fcw -UserRule $Rule
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AuthorizeIpRules](#) 中的。

Copy-WKSWorkspaceImage

以下代码示例演示了如何使用 Copy-WKSWorkspaceImage。

适用于 PowerShell V4 的工具

示例 1：此示例将具有指定 ID 的工作空间图像从 us-west-2 复制到名为 “” 的当前区域 CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId  
wsi-djfoedhw6
```

输出：

```
wsi-456abaqfe
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CopyWorkspaceImage](#) 中的。

Edit-WKSClientProperty

以下代码示例演示了如何使用 Edit-WKSClientProperty。

适用于 PowerShell V4 的工具

示例 1：此示例为 Workspaces 客户端启用重新连接

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled  
"ENABLED" -ResourceId d-123414a369
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyClientProperties](#) 中的。

Edit-WKSSelfServicePermission

以下代码示例演示了如何使用 Edit-WKSSelfServicePermission。

适用于 PowerShell V4 的工具

示例 1：此示例启用自助服务权限，以便为指定的目录更改计算类型并增加卷大小

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId  
d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -  
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifySelfservicePermissions](#) 中的。

Edit-WKSWorkspaceAccessProperty

以下代码示例演示了如何使用 Edit-WKSWorkspaceAccessProperty。

适用于 PowerShell V4 的工具

示例 1：此示例在 Android 和 Chrome 操作系统上针对指定目录启用工作空间访问

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -  
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ModifyWorkspaceAccessProperties](#) 中的。

Edit-WKSWorkspaceCreationProperty

以下代码示例演示了如何使用 Edit-WKSWorkspaceCreationProperty。

适用于 PowerShell V4 的工具

示例 1：此示例支持在创建工作空间时将互联网访问和维护模式设置为 true 作为默认值

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -  
WorkspaceCreationProperties_EnableMaintenanceMode $true
```


- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyWorkspaceCreationProperties`](#) 中的。

Edit-WKSWorkspaceProperty

以下代码示例演示了如何使用 `Edit-WKSWorkspaceProperty`。

适用于 PowerShell V4 的工具

示例 1：此示例将指定工作空间的“工作空间运行模式”属性更改为“自动停止”

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -  
WorkspaceProperties_RunningMode AUTO_STOP
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyWorkspaceProperties`](#) 中的。

Edit-WKSWorkspaceState

以下代码示例演示了如何使用 `Edit-WKSWorkspaceState`。

适用于 PowerShell V4 的工具

示例 1：此示例将指定工作空间的状态更改为“可用”

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState  
AVAILABLE
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) `ModifyWorkspaceState`](#) 中的。

Get-WKSClientProperty

以下代码示例演示了如何使用 `Get-WKSClientProperty`。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定目录的工作空间客户端的客户端属性

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeClientProperties](#) 中的。

Get-WKSIpGroup

以下代码示例演示了如何使用 Get-WKSIpGroup。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定区域中指定 IP 组的详细信息

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

输出：

```
GroupDesc GroupId      GroupName UserRules
-----
wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
Amazon.WorkSpaces.Model.IpRuleItem}
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribelpGroups](#) 中的。

Get-WKSTag

以下代码示例演示了如何使用 Get-WKSTag。

适用于 PowerShell V4 的工具

示例 1：此示例获取给定工作空间的标签

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

输出：

```
Key      Value
---      -
auto-delete no
purpose   Workbench
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeTags](#) 中的。

Get-WKSWorkspace

以下代码示例演示了如何使用 Get-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：WorkSpaces 将您的所有详细信息检索到管道。

```
Get-WKSWorkspace
```

输出：

```
BundleId           : wsb-1a2b3c4d
ComputerName       :
DirectoryId        : d-1a2b3c4d
ErrorCode          :
ErrorMessage       :
IpAddress          :
RootVolumeEncryptionEnabled : False
State              : PENDING
SubnetId           :
UserName           : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey :
WorkspaceId        : ws-1a2b3c4d
WorkspaceProperties : Amazon.WorkSpaces.Model.WorkspaceProperties
```

示例 2：此命令显示 **us-west-2** 区域中某个工作空间的 **WorkspaceProperties** 的各个子属性的值。有关子属性的更多信息 **WorkspaceProperties**，请参阅 https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html。

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties
```

输出：

```
ComputeTypeName    : STANDARD
RootVolumeSizeGib  : 80
RunningMode        : AUTO_STOP
```

```
RunningModeAutoStopTimeoutInMinutes : 60
UserVolumeSizeGib                    : 50
```

示例 3：此命令显示 **us-west-2** 区域中某个工作空间的 **WorkspaceProperties** 的子属性 **RootVolumeSizeGib** 的值。根卷大小 (以 GiB 为单位) 为 80。

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

输出：

```
80
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeWorkspaces](#) 中的。

Get-WKSWorkspaceBundle

以下代码示例演示了如何使用 `Get-WKSWorkspaceBundle`。

适用于 PowerShell V4 的工具

示例 1：此示例获取当前区域中所有工作空间捆绑包的详细信息

```
Get-WKSWorkspaceBundle
```

输出：

```
BundleId       : wsb-sfhgvd342
ComputeType    : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId        : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name           : CustomBundleTest
Owner          : 233816212345
RootStorage    : Amazon.WorkSpaces.Model.RootStorage
UserStorage    : Amazon.WorkSpaces.Model.UserStorage
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeWorkspaceBundles](#) 中的。

Get-WKSWorkspaceDirectory

以下代码示例演示了如何使用 Get-WKSWorkspaceDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例列出已注册目录的目录详细信息

```
Get-WKSWorkspaceDirectory
```

输出：

```
Alias                : TestWorkspace
CustomerUserName     : Administrator
DirectoryId          : d-123414a369
DirectoryName        : TestDirectory.com
DirectoryType        : MicrosoftAD
DnsIpAddresses       : {172.31.43.45, 172.31.2.97}
IamRoleId            : arn:aws:iam::761234567801:role/workspaces_RoleDefault
IpGroupIds           : {}
RegistrationCode      : WSpdx+4RRT43
SelfservicePermissions : Amazon.WorkSpaces.Model.SelfservicePermissions
State               : REGISTERED
SubnetIds            : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy              : SHARED
WorkspaceAccessProperties : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
    Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId : sg-0ed2441234a123c43
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeWorkspaceDirectories](#) 中的。

Get-WKSWorkspaceImage

以下代码示例演示了如何使用 Get-WKSWorkspaceImage。

适用于 PowerShell V4 的工具

示例 1：此示例获取区域中所有映像的所有详细信息

```
Get-WKSWorkspaceImage
```

输出：

```

Description      : This image is copied from another image
ErrorCode        :
ErrorMessage     :
ImageId          : wsi-345ahdjgo
Name             : CopiedImageTest
OperatingSystem  : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy  : DEFAULT
State            : AVAILABLE
  
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeWorkpaceImages](#) 中的。

Get-WKSWorkspaceSnapshot

以下代码示例演示了如何使用 Get-WKSWorkspaceSnapshot。

适用于 PowerShell V4 的工具

示例 1：此示例演示为指定工作空间创建的最新快照的时间戳

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

输出：

```

RebuildSnapshots      RestoreSnapshots
-----
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
  
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeWorkspaceSnapshots](#) 中的。

Get-WKSWorkspacesConnectionStatus

以下代码示例演示了如何使用 Get-WKSWorkspacesConnectionStatus。

适用于 PowerShell V4 的工具

示例 1：此示例获取指定工作空间的连接状态

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeWorkspacesConnectionStatus](#) 中的。

New-WKSIpGroup

以下代码示例演示了如何使用 New-WKSIpGroup。

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为的空 IP 组 FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

输出：

```
wsipg-w45rty4ty
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateIpGroup](#) 中的。

New-WKSTag

以下代码示例演示了如何使用 New-WKSTag。

适用于 PowerShell V4 的工具

示例 1：此示例向名为 **ws-wsname** 的工作空间添加一个新标签。该标签的键为“Name”，键值为 **AWS_Workspace**。

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag  
$tag.Key = "Name"  
$tag.Value = "AWS_Workspace"  
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

示例 2：此示例向名为 **ws-wsname** 的工作空间添加多个标签。一个标签的键为“Name”，键值为 **AWS_Workspace**；另一个标签的标签键为“Stage”，键值为“Test”。

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateTags](#) 中的。

New-WKSWorkspace

以下代码示例演示了如何使用 New-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：Workspace 为提供的捆绑包、目录和用户创建。

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
"d-1a2b3c4d"; "UserName" = "USERNAME"}
```

示例 2：此示例创建了多个 WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
= "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
"DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateWorkspaces](#) 中的。

Register-WKSIpGroup

以下代码示例演示了如何使用 Register-WKSIpGroup。

适用于 PowerShell V4 的工具

示例 1：此示例将指定的 IP 组注册到指定的目录

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```


- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [AssociateIpGroups](#) 中的。

Register-WKSWorkspaceDirectory

以下代码示例演示了如何使用 Register-WKSWorkspaceDirectory。

适用于 PowerShell V4 的工具

示例 1：此示例注册 Workspaces 服务的指定目录

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterWorkspaceDirectory](#) 中的。

Remove-WKSIpGroup

以下代码示例演示了如何使用 Remove-WKSIpGroup。

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的 IP 组

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSIpGroup (DeleteIpGroup)" on target
"wsipg-32fhgtred".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteIpGroup](#) 中的。

Remove-WKSTag

以下代码示例演示了如何使用 Remove-WKSTag。

适用于 PowerShell V4 的工具

示例 1：此示例移除与工作空间关联的标签

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DeleteTags](#) 中的。

Remove-WKSWorkspace

以下代码示例演示了如何使用 Remove-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：终止多个 WorkSpaces。使用 -Force 开关会阻止 cmdlet 提示确认。

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

示例 2：检索所有你的集合，WorkSpaces 然后通过管道传递 IDs 到 Remove-的-WorkspaceId 参数 WKSWorkspace，终止所有的。WorkSpacescmdlet 将在每个 Workspace 终止之前进行提示。要禁止确认提示，请添加 -Force 开关。

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

示例 3：此示例说明如何传递定义 WorkSpaces 要终止的 TerminateRequest 对象。除非也指定了 -Force 开关参数，否则在继续操作之前，cmdlet 会提示您进行确认。

```
$arrRequest = @(
```

```
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) TerminateWorkspaces](#) 中的。

Reset-WKSWorkspace

以下代码示例演示了如何使用 Reset-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：重建指定 Workspace 的。

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

示例 2：检索所有你的集合，WorkSpaces 然后通过管道 IDs 将它们传递给 Reset-的-WorkspaceId 参数 WKSWorkspace，从而重新构建。WorkSpaces

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) RebuildWorkspaces](#) 中的。

Restart-WKSWorkspace

以下代码示例演示了如何使用 Restart-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：重新启动指定 Workspace 的。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

示例 2：多次 WorkSpaces 重启。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

示例 3：检索所有你的集合，WorkSpaces 并将通过管道传递 IDs 给 Restart-的-WorkSpaceId 参数 WKSWorkspace，从而 WorkSpaces 使重新启动。

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RebootWorkspaces](#) 中的。

Stop-WKSWorkspace

以下代码示例演示了如何使用 Stop-WKSWorkspace。

适用于 PowerShell V4 的工具

示例 1：停止多个 WorkSpaces。

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

示例 2：检索所有你的集合，WorkSpaces 然后通过管道传递 IDs 给 Stop-的-WorkSpaceId 参数，WKSWorkspace WorkSpaces 从而导致停止。

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

示例 3：此示例说明如何传递定义 WorkSpaces 要停止的 StopRequest 对象。

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [StopWorkspaces](#) 中的。

Unregister-WKSIpGroup

以下代码示例演示了如何使用 Unregister-WKSIpGroup。

适用于 PowerShell V4 的工具

示例 1：此示例从指定的目录中取消注册指定的 IP 组

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DisassociateIpGroups](#) 中的。

本 AWS 产品或服务的安全性

云安全性一直是 Amazon Web Services (AWS) 的重中之重。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。安全是双方共同承担 AWS 的责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性。

云安全 — AWS 负责保护运行 AWS 云中提供的所有服务的基础架构，并为您提供可以安全使用的服务。我们的安全责任是重中之重 AWS，作为[AWS 合规计划](#)的一部分，第三方审计师定期测试和验证我们安全的有效性。

云端安全 — 您的责任由您使用的 AWS 服务以及其他因素决定，包括数据的敏感性、组织的要求以及适用的法律和法规。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划合 AWS 规工作范围内的 AWS 服务](#)。

主题

- [这方面的数据保护 AWS 产品或服务](#)
- [身份和访问管理](#)
- [为此进行合规性验证 AWS 产品或服务](#)
- [在适用于 PowerShell 的工具中强制实施最低的 TLS 版本](#)
- [工具的其他安全注意事项 PowerShell](#)

这方面的数据保护 AWS 产品或服务

[责任 AWS 共担模式](#)适用于本 AWS 产品或服务中的数据保护。如本模型所述 AWS，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 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[数据隐私常见问题解答 AWS](#)条款。有关欧洲数据保护的信息，请参阅[通用数据保护条例 \(GDPR\) 中心](#)。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。

- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅 AWS CloudTrail 用户指南中的[使用跟 CloudTrail 踪](#)。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》<https://aws.amazon.com/compliance/fips/>。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或 AWS SDK AWS 服务使用本 AWS 产品或服务或其他产品或服务时。AWS CLI 在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

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 AWS 服务进行交互所需的凭据外，本身 AWS Tools for PowerShell 不存储任何客户数据。

如果您使用调用将客户数据传输 AWS Tools for PowerShell 到本地计算机进行存储的 AWS 服务，则请参阅该服务的《用户指南》中的“安全与合规性”一章，了解如何存储、保护和加密这些数据的信息。

传输中加密

默认情况下，从运行 AWS Tools for PowerShell 和 AWS 服务端点的客户端计算机传输的所有数据都通过 HTTPS/TLS 连接发送所有数据进行加密。

您无需执行任何操作即可使用 HTTPS/TLS。它始终处于启用状态。

身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证 (登录) 和授权 (拥有权限) 使用 AWS 资源。您可以使用 IAM AWS 服务 , 无需支付额外费用。

主题

- [受众](#)
- [使用身份进行身份验证](#)
- [使用策略管理访问](#)
- [如何 AWS 服务 使用 IAM](#)
- [对 AWS 身份和访问进行故障排除](#)

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同, 具体取决于您所做的工作 AWS。

服务用户-如果您 AWS 服务 曾经完成工作, 则您的管理员会为您提供所需的凭证和权限。当您使用更多 AWS 功能来完成工作时, 您可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 AWS, 请参阅[对 AWS 身份和访问进行故障排除](#)或 AWS 服务 您正在使用的用户指南。

服务管理员-如果您负责公司的 AWS 资源, 则可能拥有完全访问权限 AWS。您的工作是确定您的服务用户应访问哪些 AWS 功能和资源。然后, 您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何使用 IAM AWS, 请参阅 AWS 服务 您正在使用的用户指南。

IAM 管理员: 如果您是 IAM 管理员, 您可能希望了解如何编写策略以管理对 AWS 的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS 基于身份的策略示例, 请参阅 AWS 服务 您正在使用的用户指南。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户, 或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center (例如 (IAM Identity Center)、单点登录身份验证或 Google/Facebook 证书, 以联合身份登录。有关登录的更多信息, 请参阅《AWS 登录 用户指南》中的[如何登录您的 AWS 账户](#)。

对于编程访问, AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息, 请参阅《IAM 用户指南》中的[适用于 API 请求的 AWS 签名版本 4](#)。

AWS 账户 root 用户

创建时 AWS 账户, 首先会有一个名为 AWS 账户 root 用户的登录身份, 该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务, 请参阅《IAM 用户指南》中的[需要根用户凭证的任务](#)。

联合身份

作为最佳实践, 要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service , 或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限, 建议使用。AWS IAM Identity Center 有关更多信息, 请参阅《AWS IAM Identity Center 用户指南》中的[什么是 IAM Identity Center ?](#)。

IAM 用户和群组

[IAM 用户](#)是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证, 而非具有长期凭证的 IAM 用户。有关更多信息, 请参阅 IAM 用户指南中的[要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问](#)。

[IAM 组](#)指定一组 IAM 用户, 便于更轻松地对大量用户进行权限管理。有关更多信息, 请参阅《IAM 用户指南》中的[IAM 用户使用案例](#)。

IAM 角色

[IAM 角色](#)是具有特定权限的身份, 可提供临时凭证。您可以通过[从用户切换到 IAM 角色 \(控制台\)](#)或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息, 请参阅《IAM 用户指南》中的[担任角色的方法](#)。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息, 请参阅《IAM 用户指南》中的[IAM 中的跨账户资源访问](#)。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 [JSON 策略概述](#)。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的[使用客户管理型策略定义自定义 IAM 权限](#)。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的[在托管策略与内联策略之间进行选择](#)。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中[指定主体](#)。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。AWS WAF 要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的[访问控制列表 \(ACL\) 概述](#)。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《IAM 用户指南》中的 [IAM 实体的权限边界](#)。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的[服务控制策略](#)。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的[资源控制策略 \(RCPs\)](#)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的[会话策略](#)。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的[策略评估逻辑](#)。

如何 AWS 服务 使用 IAM

要全面了解如何 AWS 服务 使用大多数 IAM 功能，请参阅 IAM 用户指南中的与 IAM [配合使用的AWS 服务](#)。

要了解如何在 IAM 中 AWS 服务 使用特定的，请参阅相关服务的《用户指南》的安全部分。

对 AWS 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 AWS 和 IAM 时可能遇到的常见问题。

主题

- [我无权在以下位置执行操作 AWS](#)
- [我无权执行 iam : PassRole](#)
- [我想允许我以外的人 AWS 账户 访问我的 AWS 资源](#)

我无权在以下位置执行操作 AWS

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 *awes:GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 `aws:GetWidget` 操作访问 `my-example-widget` 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam : PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给。AWS

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 AWS 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 AWS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS 支持这些功能，请参阅[如何 AWS 服务 使用 IAM](#)。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户，请参阅[IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问](#)权限。
- 要了解如何向第三方提供对您的资源的访问[权限 AWS 账户](#)，请参阅[IAM 用户指南中的向第三方提供访问](#)权限。AWS 账户

- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的[为经过外部身份验证的用户 \(身份联合验证\) 提供访问权限](#)。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的[IAM 中的跨账户资源访问](#)。

为此进行合规性验证 AWS 产品或服务

要了解是否属于特定合规计划的范围，请参阅AWS 服务“[按合规计划划分的范围](#)”，然后选择您感兴趣的合规计划。AWS 服务 有关一般信息，请参阅[AWS 合规计划AWS](#)。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的“[下载报告](#)”中的“[AWS Artifact](#)”。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅[AWS 安全文档](#)。

本 AWS 产品或服务通过其支持的特定Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划合 AWS 规工作范围内的AWS 服务](#)。

在适用于 PowerShell 的工具中强制实施最低的 TLS 版本

为了提高与 AWS 服务通信时的安全性，应将适用于 PowerShell 的工具配置为使用适当的 TLS 版本。有关如何执行此操作的信息，请参阅[《适用于 .NET 的 AWS SDK 开发人员指南》](#)中的[强制实施最低的 TLS 版本](#)。

工具的其他安全注意事项 PowerShell

本主题包含除前面部分涵盖的安全主题之外的其他安全注意事项。

敏感信息的记录

此工具的某些操作可能会返回被视为敏感的信息，包括来自环境变量的信息。在某些情况下，泄露这些信息可能构成安全风险；例如，这些信息可能包含在持续集成和持续部署 (CI/CD) 日志中。因此，请务必核查何时将此类输出作为日志的一部分，并在不需要时隐藏该输出。有关保护敏感数据的其他信息，请参阅[这方面的数据保护 AWS 产品或服务](#)。

考虑下面的最佳实践：

- 不要使用环境变量来存储无服务器资源的敏感值。而是让您的无服务器代码以编程方式从密钥存储库中检索密钥 (例如)。AWS Secrets Manager
- 查看构建日志的内容，确保其中不包含敏感信息。考虑诸如通过管道传输到 `/dev/null` 或将输出捕获为 `bash` 或 PowerShell 变量来抑制命令输出之类的方法。
- 考虑日志的访问权限，并根据您的使用案例适当确定访问范围。

Tools for PowerShell Cmdlet 参考

Tools for PowerShell 提供了可用于访问 AWS 服务的 cmdlet。要查看有哪些 cmdlet 可用，请参阅 [AWS Tools for PowerShell Cmdlet 参考](#)。

文档历史记录

本主题介绍了对AWS 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AWS Tools for PowerShell的更改和更新的更多信息，请参阅[发行说明](#)。如需获取对此文档的更新的通知，您可以订阅 [RSS 源](#)。

变更	说明	日期
新增内容	已宣布终止对 AWS Tools for PowerShell V4 的支持。	2025 年 9 月 17 日
新增内容	AWS Tools for PowerShell 的版本 5 (V5) 已正式发布！有关更多信息，请参阅 《AWS Tools for PowerShell 用户指南 (V5) 》 ，尤其是 迁移到 V5 主题。	2025 年 6 月 23 日
新增内容	发布了 AWS Tools for PowerShell V5 的预览内容。	2025 年 6 月 13 日
新增内容	发布了 AWS Tools for PowerShell 版本 5 (预览版) 的 《用户指南》 。	2025 年 5 月 28 日
可观测性	宣布推出可观测性的 GA 版本	2025 年 2 月 10 日
新增内容	添加了关于完整性保护新默认行为的信息。	2025 年 1 月 15 日
新增内容	添加了有关 AWS Tools for PowerShell 版本 5 的第一个预览版的信息。	2024 年 11 月 18 日

管道、输出和迭代	替换了关于 \$AWSHistory 的内容，该变量已被弃用。	2024 年 10 月 10 日
可观测性	AWS Tools for PowerShell 中添加了有关可观测性的预览信息，该功能支持收集遥测数据。	2024 年 9 月 13 日
在 Windows 上安装 AWS Tools for PowerShell	添加了在提取内容之前解除锁定 ZIP 文件的信息。	2024 年 8 月 5 日
关于 EC2-Classical 的信息	移除了关于已停用的 EC2-Classical 的信息。	2024 年 8 月 1 日
代码示例	包含了一个带有 cmdlet 示例的章节。	2024 年 4 月 17 日
其它安全注意事项	包含了关于敏感数据可能被记录的信息。	2024 年 4 月 16 日
使用 AWS 配置工具身份验证	添加了有关 AWS Tools for PowerShell 中支持 SSO 的信息。	2024 年 3 月 15 日
Tools for PowerShell Cmdlet 参考	添加了包含 Tools for PowerShell cmdlet 参考链接的部分。	2023 年 11 月 17 日
加入了更多 IAM 最佳实践更新	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践 。	2023 年 10 月 12 日
在 Windows 上安装	删除了有关使用已弃用的 MSI 安装适用于 Windows PowerShell 的工具的信息。	2023 年 9 月 25 日
IAM 最佳实践更新	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践 。	2023 年 9 月 8 日

管道传输和 \$AWSHistory	向 Set-AWSHistoryConfiguration cmdlet 添加了 IncludeSensitiveData 参数。	2023 年 3 月 9 日
在 cmdlet 中使用 ClientConfig 参数	添加了对 ClientConfig 参数的支持的相关信息。	2022 年 10 月 28 日
使用 Windows PowerShell 启动 Amazon EC2 实例	添加了有关停用 EC2-Classical 的说明。	2022 年 7 月 26 日
AWS Tools for PowerShell 版本 4	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 主题和对新特征的介绍。	2019 年 11 月 21 日
AWS Tools for PowerShell 3.3.563	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 AWS 服务中的一个模块。	2019 年 10 月 18 日
AWS Tools for PowerShell 3.3.343.0	在 使用 AWS Tools for PowerShell 部分中添加了信息，介绍可供 PowerShell Core 开发人员构建 AWS Lambda 函数的 AWS Lambda Tools for PowerShell。	2018 年 9 月 11 日
AWS Tools for Windows PowerShell 3.1.31.0	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

[AWS Tools for Windows PowerShell 2.3.19](#)

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找到所需的 AWS cmdlet。

2015 年 2 月 5 日

[AWS Tools for Windows PowerShell 1.1.1.0](#)

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 的可用性問題。Initialize-AWSDefaults 从其获取凭证和区域数据的位置进行报告。Stop-EC2Instance 接受 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。

2013 年 5 月 15 日

[AWS Tools for Windows PowerShell 1.0.1.0](#)

Tools for Windows PowerShell 2012 年 12 月 21 日

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。

[AWS Tools for Windows PowerShell 1.0.0.0](#)

初始版本

2012 年 12 月 6 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。