Guía del usuario

AWS Tools for PowerShell

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Tools for PowerShell: Guía del usuario

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué son las AWS Tools for PowerShell?

AWS Tools for PowerShell Son un conjunto de PowerShell módulos que se basan en la funcionalidad expuesta por el AWS SDK for .NET. Le AWS Tools for PowerShell permiten programar operaciones en sus AWS recursos desde la línea de PowerShell comandos.

Los cmdlets proporcionan una PowerShell experiencia idiomática para especificar parámetros y gestionar los resultados, aunque se implementen mediante las distintas consultas de servicios. AWS HTTP APIs Por ejemplo, los cmdlets para la canalización de AWS Tools for PowerShell soporte, es PowerShell decir, se pueden canalizar objetos dentro y fuera de los cmdlets. PowerShell

AWS Tools for PowerShell Son flexibles en cuanto a la forma en que permiten gestionar las credenciales, incluida la compatibilidad con la infraestructura (). AWS Identity and Access Management IAM Puede utilizar las herramientas con credenciales IAM de usuario, símbolos de seguridad temporales y IAM funciones.

Son AWS Tools for PowerShell compatibles con el mismo conjunto de servicios y AWS regiones que admite elSDK. Puede instalarlo AWS Tools for PowerShell en ordenadores con sistemas operativos Windows, Linux o macOS.

> **Note**
>
> AWS Tools for PowerShell la versión 4 es la última versión principal y es una actualización de la versión 3.3 compatible con versiones anteriores. AWS Tools for PowerShell Agrega mejoras significativas a la vez que mantiene el comportamiento existente del cmdlet. Los scripts existentes deberían seguir funcionando después de actualizar a la nueva versión, pero recomendamos que los pruebe a fondo antes de actualizar. Para obtener más información sobre los cambios de la versión 4, consulte Migración de la AWS Tools for PowerShell versión 3.3 a la versión 4.

AWS Tools for PowerShell Están disponibles en los siguientes tres paquetes distintos:

- `AWS.Tools`
- AWSPowerShell.NetCore
- AWSPowerShell

Mantenimiento y soporte para SDK las versiones principales

Para obtener información sobre el mantenimiento y el soporte de las versiones SDK principales y sus dependencias subyacentes, consulte lo siguiente en la Guía de referencia de las herramientas AWS SDKs y herramientas:

- AWS SDKsy política de mantenimiento de herramientas
- AWS SDKsmatriz de soporte de versiones y herramientas

`AWS.Tools`- Una versión modularizada del AWS Tools for PowerShell

Esta versión de AWS Tools for PowerShell es la recomendada para cualquier ordenador que funcione PowerShell en un entorno de producción. Como esta versión está dividida en módulos, solo debe descargar y utilizar los módulos de los servicios que desee utilizar. Esto reduce los tiempos de descarga, el uso de memoria y permite, en la mayoría de los casos, importar automáticamente los cmdlets de `AWS.Tools` sin la necesidad de llamar manualmente a `Import-Module` primero.

Esta es la versión más reciente AWS Tools for PowerShell y se ejecuta en todos los sistemas operativos compatibles, incluidos Windows, Linux y macOS. Este paquete proporciona un módulo de instalación`AWS.Tools.Installer`, un módulo común y un módulo para cada AWS servicio, por ejemplo,`AWS.Tools.EC2`, `AWS.Tools.IdentityManagement`AWS.Tools.S3, y así sucesivamente. `AWS.Tools.Common`

El `AWS.Tools.Installer` módulo proporciona cmdlets que permiten instalar, actualizar y eliminar los módulos de cada uno de los AWS servicios. Los cmdlets de este módulo garantizan automáticamente que dispone de todos los módulos dependientes necesarios para admitir los módulos que desea utilizar.

El módulo `AWS.Tools.Common` proporciona cmdlets para la configuración y la autenticación que no son específicos del servicio. Para usar los cmdlets para un AWS servicio, basta con ejecutar el comando. PowerShellimporta automáticamente el `AWS.Tools.Common` módulo y el módulo

del AWS servicio cuyo cmdlet desee ejecutar. Este módulo se instala automáticamente si utiliza el módulo `AWS.Tools.Installer` para instalar los módulos de servicio.

Puede instalar esta versión de AWS Tools for PowerShell en equipos que ejecuten:

- PowerShell Core 6.0 o posterior en Windows, Linux o macOS.
- Windows PowerShell 5.1 o posterior en Windows con. NETFramework 4.7.2 o posterior.

A lo largo de esta guía, cuando necesitemos especificar esta versión solamente, nos referiremos a ella por su nombre de módulo: *`AWS.Tools`* .

AWSPowerShell. NetCore - Una versión de un solo módulo del AWS Tools for PowerShell

Esta versión consta de un único módulo grande que contiene soporte para todos los AWS servicios. Antes de poder utilizar este módulo, debe importarlo manualmente.

Puede instalar esta versión de AWS Tools for PowerShell en equipos que ejecuten:

- PowerShell Core 6.0 o posterior en Windows, Linux o macOS.
- Windows PowerShell 3.0 o posterior en Windows con. NETFramework 4.7.2 o posterior.

A lo largo de esta guía, cuando necesitamos especificar solo esta versión, nos referimos a ella por el nombre de su módulo:AWSPowerShell. NetCore.

AWSPowerShell- Una versión de un solo módulo para Windows PowerShell

Esta versión de solo AWS Tools for PowerShell es compatible e instalable en ordenadores Windows que ejecuten las PowerShell versiones 2.0 a 5.1 de Windows. No es compatible con PowerShell Core

6.0 o posterior, ni con ningún otro sistema operativo (Linux o macOS). Esta versión consta de un único módulo grande que contiene soporte para todos los AWS servicios.

A lo largo de esta guía, cuando necesitamos especificar solo esta versión, nos referimos a ella por el nombre de su módulo: AWSPowerShell.

Cómo usar esta guía

Esta guía se divide en las siguientes secciones principales.

Instalación de AWS Tools for PowerShell

En esta sección se explica cómo instalar el AWS Tools for PowerShell. Incluye cómo registrarse AWS si aún no tiene una cuenta y cómo crear un IAM usuario que pueda usar para ejecutar los cmdlets.

Comenzar a utilizar la AWS Tools for Windows PowerShell

En esta sección se describen los aspectos básicos del uso de los AWS Tools for PowerShell, como especificar las credenciales y AWS las regiones, buscar los cmdlets para un servicio concreto y utilizar los alias de los cmdlets.

Trabaje con AWS los servicios del AWS Tools for PowerShell

En esta sección se incluye información sobre su uso AWS Tools for PowerShell para realizar algunas de las tareas más comunes. AWS

Temas adicionales en esta sección

- Qué hay de nuevo en el AWS Tools for PowerShell

Qué hay de nuevo en el AWS Tools for PowerShell

Para obtener información de alto nivel sobre los nuevos desarrollos relacionados con el AWS Tools for PowerShell, consulte la página del producto en https://aws.amazon.com/powershell/.

A continuación se muestran las novedades de las Herramientas para PowerShell.

13 de septiembre de 2024: versión preliminar para la observabilidad

Esta es una documentación preliminar para una característica en versión de vista previa. Está sujeta a cambios.

La observabilidad es la medida en que se puede deducir el estado actual de un sistema a partir de los datos que emite. Se ha agregado la observabilidad a las herramientas PowerShell, incluida la implementación de un proveedor de telemetría.

Instalación de AWS Tools for PowerShell

Para instalar y utilizar correctamente los cmdlets de las AWS Tools for PowerShell, consulte los pasos descritos en los temas siguientes.

Temas

Instalación del AWS Tools for PowerShell en Windows

Un equipo basado en Windows puede ejecutar cualquiera de las opciones del AWS Tools for PowerShell paquete:

- **`AWS.Tools`**- La versión modularizada de. AWS Tools for PowerShell Cada AWS servicio está respaldado por su propio módulo pequeño e individual, con módulos `AWS.Tools.Common` de soporte compartidos y. `AWS.Tools.Installer`
- AWSPowerShell. NetCore- La versión única de módulos grandes de AWS Tools for PowerShell. Todos los AWS servicios son compatibles con este módulo único y grande.

  > **Note**
  >
  > Tenga en cuenta que el módulo individual puede ser demasiado grande para usarlo con funciones de AWS Lambda. En su lugar, utilice la versión modularizada que se muestra arriba.

- AWSPowerShell- La versión antigua de un solo módulo grande específica para Windows de. AWS Tools for PowerShell Todos los AWS servicios son compatibles con este módulo único y de gran tamaño.

El paquete que elija depende de la versión y edición de Windows que esté ejecutando.

> **Note**
>
> El AWSPowerShell módulo Herramientas para Windows PowerShell se instala de forma predeterminada en todas las Amazon Machine Images (AMIs) basadas en Windows.

La configuración AWS Tools for PowerShell implica las siguientes tareas de alto nivel, que se describen en detalle en este tema.

1. Instale la opción de AWS Tools for PowerShell paquete adecuada para su entorno.
2. Compruebe que la ejecución de scripts está habilitada ejecutando el cmdlet `Get-ExecutionPolicy`.
3. Importe el AWS Tools for PowerShell módulo a su PowerShell sesión.

Requisitos previos

Las versiones más recientes de PowerShell, incluida PowerShell Core, están disponibles como descargas en Microsoft en Instalación de varias versiones de PowerShell en el sitio web de Microsoft.

Instalación de **`AWS.Tools`** en Windows.

Puede instalar la versión modularizada de AWS Tools for PowerShell en equipos que ejecuten Windows con Windows PowerShell 5.1, PowerShell Core 6.0 o posterior. Para obtener información acerca de cómo instalar PowerShell Core, consulte Instalación de varias versiones de PowerShell en el sitio web de Microsoft.

Puede instalar `AWS.Tools` de tres maneras:

- Utilizando los cmdlets del módulo `AWS.Tools.Installer`. Este módulo simplifica la instalación y actualización de otros `AWS.Tools` módulos. `AWS.Tools.Installer`requiere `PowerShellGet` y descarga e instala automáticamente una versión actualizada del mismo. `AWS.Tools.Installer`mantiene sincronizadas automáticamente las versiones de sus módulos. Al instalar o actualizar a una versión más reciente de un módulo, los cmdlets `AWS.Tools.Installer` actualizan automáticamente todos los demás `AWS.Tools` módulos a la misma versión.

  Este método se describe en el procedimiento siguiente.

- Descargando los módulos de `AWS.Tools.zip` y extrayéndolos en una de las carpetas del módulo. Puede descubrir cuáles son las carpetas del módulo mostrando el valor de la variable de entorno `PSModulePath`.

  > **Warning**
  >
  > Después de descargar el ZIP archivo y antes de extraer el contenido, puede que tenga que desbloquearlo. Por lo general, esto se hace abriendo las propiedades del archivo, viendo la pestaña General y activando la casilla Desbloquear, si existe alguna.
  > Si necesitas ZIP desbloquear el archivo pero no lo haces, es posible que recibas errores similares a los siguientes: «Módulo de importación: no se pudo cargar el archivo o el ensamblaje».

- Instalar cada módulo de servicio de la PowerShell Galería mediante el cmdlet. `Install-Module`

Para instalar **`AWS.Tools`** en Windows mediante el módulo **`AWS.Tools.Installer`**

1. Iniciar una PowerShell sesión.

   > **Note**
   >
   > Le recomendamos que no se postule PowerShell como administrador con permisos elevados, excepto cuando lo exija la tarea en cuestión. Esto puede suponer un riesgo para la seguridad y no se atiene al principio de privilegios mínimos.

2. Para instalar el paquete de `AWS.Tools` por módulos, ejecute el siguiente comando.

   ```
   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
   ```

   Si aparece una notificación en la que se indica que el repositorio no es de confianza, se le preguntará si desea realizar la instalación de todos modos. Introduzca **y** para PowerShell

permitir la instalación del módulo. Para evitar este mensaje e instalar el módulo sin confiar en el repositorio, puede ejecutar el comando con el parámetro `-Force`.

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

3. Ahora puede instalar el módulo para cada AWS servicio que desee usar mediante el `Install-AWSToolsModule` cmdlet. Por ejemplo, el siguiente comando instala los módulos Amazon EC2 y Amazon S3. Este comando también instala los módulos dependientes necesarios para que el módulo especificado funcione. Por ejemplo, cuando instala el primer módulo de servicio `AWS.Tools`, también se instala `AWS.Tools.Common`. Se trata de un módulo compartido que requieren todos los módulos AWS de servicio. También elimina las versiones anteriores de los módulos y actualiza otros módulos a la misma versión.

```
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**
>
> El `Install-AWSToolsModule` cmdlet descarga todos los módulos solicitados del `PSRepository` nombre `PSGallery` (https://www.powershellgallery.com/) y lo considera una fuente de confianza. Utilice el comando `Get-PSRepository -Name PSGallery` para obtener más información sobre este repositorio de `PSRepository`.

De forma predeterminada, el comando anterior instala los módulos en la carpeta `%USERPROFILE%\Documents\WindowsPowerShell\Modules`. Para instalarlo AWS Tools for PowerShell para todos los usuarios de un equipo, debe ejecutar el siguiente comando en una PowerShell sesión que haya iniciado como administrador. Por ejemplo, el siguiente comando instala el IAM módulo en la `%ProgramFiles%\WindowsPowerShell\Modules` carpeta a la que pueden acceder todos los usuarios.

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

Para instalar otros módulos, ejecute comandos similares con los nombres de módulo correspondientes, tal y como se encuentra en la PowerShell Galería.

Instalar AWSPowerShell. NetCore en Windows

Puede instalar el AWSPowerShell. NetCore en equipos que ejecutan Windows con las PowerShell versiones 3 a 5.1 o PowerShell Core 6.0 o posterior. Para obtener información acerca de cómo instalar PowerShell Core, consulte Instalación de varias versiones de PowerShell en el PowerShell sitio web de Microsoft.

Puede instalarlo AWSPowerShell. NetCore de dos maneras

- Descargando el módulo desde AWSPowerShell. NetCore.zip y extrayéndolo en uno de los directorios del módulo. Puede descubrir cuáles son los directorios del módulo mostrando el valor de la variable de entorno `PSModulePath`.

> **Warning**
>
> Después de descargar el ZIP archivo y antes de extraer el contenido, puede que tengas que desbloquearlo. Por lo general, esto se hace abriendo las propiedades del archivo, viendo la pestaña General y activando la casilla Desbloquear, si existe alguna.
> Si necesitas ZIP desbloquear el archivo pero no lo haces, es posible que recibas errores similares a los siguientes: «Módulo de importación: no se pudo cargar el archivo o el ensamblaje».

- Instalación desde la PowerShell Galería mediante el `Install-Module` cmdlet, tal y como se describe en el siguiente procedimiento.

Para instalar. AWSPowerShell NetCore desde la PowerShell Galería mediante el cmdlet Install-Module

Para instalar el. AWSPowerShell NetCore desde la PowerShell Galería, su ordenador debe ejecutar la PowerShell versión 5.0 o una versión posterior, o ejecutar PowerShellGetla versión PowerShell 3 o una versión posterior. Ejecute el siguiente comando de la .

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

Si se ejecuta PowerShell como administrador, el comando anterior se instala AWS Tools for PowerShell para todos los usuarios del equipo. Si se ejecuta PowerShell como un usuario estándar sin permisos de administrador, ese mismo comando se instala solo AWS Tools for PowerShell para el usuario actual.

Para realizar la instalación solo para el usuario actual cuando ese usuario tiene permisos de administrador, ejecute el comando con el conjunto de parámetros `-Scope CurrentUser`, como se indica a continuación.

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

Aunque la PowerShell versión 3.0 y las versiones posteriores suelen cargar módulos en la PowerShell sesión la primera vez que se ejecuta un cmdlet en el módulo, el. AWSPowerShell NetCore el módulo es demasiado grande para admitir esta funcionalidad. En su lugar, debe cargar explícitamente el AWSPowerShell. NetCore Incorpore el módulo principal a su PowerShell sesión ejecutando el siguiente comando.

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

Para cargar el AWSPowerShell. NetCore agregue el módulo a una PowerShell sesión automáticamente, añada ese comando a su PowerShell perfil. Para obtener más información sobre cómo editar su PowerShell perfil, consulte Acerca de los perfiles en la PowerShell documentación.

Instálelo AWSPowerShell en Windows PowerShell

Puede instalarlo AWS Tools for Windows PowerShell de dos maneras:

- Descargando el módulo de AWSPowerShell.zip y extrayéndolo en uno de los directorios del módulo. Puede descubrir cuáles son los directorios del módulo mostrando el valor de la variable de entorno `PSModulePath`.

> **Warning**
>
> Después de descargar el ZIP archivo y antes de extraer el contenido, puede que tengas que desbloquearlo. Por lo general, esto se hace abriendo las propiedades del archivo, viendo la pestaña General y activando la casilla Desbloquear, si existe alguna.
> Si necesitas ZIP desbloquear el archivo pero no lo haces, es posible que recibas errores similares a los siguientes: «Módulo de importación: no se pudo cargar el archivo o el ensamblaje».

- Instalación desde la PowerShell Galería mediante el `Install-Module` cmdlet, tal y como se describe en el siguiente procedimiento.

Para realizar la instalación AWSPowerShell desde la PowerShell Galería mediante el cmdlet Install-Module

Puede instalarlo AWSPowerShell desde la PowerShell Galería si está ejecutando la PowerShell versión 5.0 o una versión posterior, o si la ha instalado PowerShellGeten PowerShell la versión 3 o una versión posterior. Puedes instalarlo y actualizarlo AWSPowerShell desde la PowerShellGalería de Microsoft ejecutando el siguiente comando.

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

Para cargar el AWSPowerShell módulo en una PowerShell sesión automáticamente, añada el `import-module` cmdlet anterior a su PowerShell perfil. Para obtener más información sobre cómo editar su PowerShell perfil, consulte Acerca de los perfiles en la PowerShell documentación.

> **Note**
>
> Las herramientas para Windows PowerShell se instalan de forma predeterminada en todas las Amazon Machine Images (AMIs) basadas en Windows.

Habilitar la ejecución de scripts

Para cargar los AWS Tools for PowerShell módulos, debe habilitar la ejecución del PowerShell script. Para habilitar la ejecución de scripts, ejecute el cmdlet `Set-ExecutionPolicy` para definir la

política `RemoteSigned`. Para obtener más información, vea Acerca de las directivas de ejecución en el sitio web de Microsoft Technet.

> **Note**
>
> Este es un requisito solo para equipos que ejecutan Windows. La restricción de seguridad de `ExecutionPolicy` no está presente en otros sistemas operativos.

Para habilitar la ejecución de scripts

1. Se requieren derechos de administrador para definir la política de ejecución. Si no ha iniciado sesión como usuario con derechos de administrador, abra una PowerShell sesión como administrador. Elija Inicio y, a continuación, elija Todos los programas. Seleccione Accesorios y, a continuación, Windows PowerShell. Haga clic con el botón derecho en Windows y PowerShell, en el menú contextual, seleccione Ejecutar como administrador.
2. En el símbolo del sistema, escriba lo siguiente.

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

> **Note**
>
> En un sistema de 64 bits, debe hacerlo por separado para la versión de 32 bits de PowerShell Windows PowerShell (x86).

Si la política de ejecución no está configurada correctamente, PowerShell muestra el siguiente error cada vez que intenta ejecutar un script, como su perfil.

```
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
```

El PowerShell instalador de Herramientas para Windows lo actualiza automáticamente PSModulePathpara incluir la ubicación del directorio que contiene el `AWSPowerShell` módulo.

Como `PSModulePath` incluye la ubicación del directorio del AWS módulo, el `Get-Module -ListAvailable` cmdlet muestra el módulo.

```
PS > Get-Module -ListAvailable

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

Control de versiones

AWS publica nuevas versiones del AWS Tools for PowerShell periódicamente para admitir nuevos AWS servicios y funciones. Para determinar la versión de las herramientas que ha instalado, ejecute el AWSPowerShellVersion cmdlet Get-.

```
PS > Get-AWSPowerShellVersion

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

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

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

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

También puede agregar el `-ListServiceVersionInfo` parámetro a un AWSPowerShellVersion comando Get- para ver una lista de los AWS servicios compatibles con la versión actual de las

herramientas. Si utiliza la opción de módulos de `AWS.Tools.*`, solo se muestran los módulos que ha importado actualmente.

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

Service                           Noun Prefix Module Name                       SDK

 Assembly

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

Para determinar la versión PowerShell que está ejecutando, introduzca `$PSVersionTable` para ver el contenido de la variable PSVersionTable automática $.

```
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
```

Actualizando el AWS Tools for PowerShell en Windows

Periódicamente, a medida que AWS Tools for PowerShell se publiquen versiones actualizadas del, debe actualizar la versión que está ejecutando localmente.

Actualice los módulos modularizados **`AWS.Tools`**

Para actualizar `AWS.Tools` los módulos a la última versión, ejecute el siguiente comando:

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

Este comando actualiza todos los módulos `AWS.Tools` que están instalados actualmente y, si esta operación se realiza correctamente, elimina otras versiones instaladas.

> **Note**
>
> El `Update-AWSToolsModule` cmdlet descarga todos los módulos del `PSRepository` nombre `PSGallery` (https://www.powershellgallery.com/) y lo considera una fuente de confianza. Utilice el comando `Get-PSRepository -Name PSGallery` para obtener más información sobre este repositorio de `PSRepository`.

Actualice las herramientas para Core PowerShell

Ejecute el `Get-AWSPowerShellVersion` cmdlet para determinar la versión que está ejecutando y compárela con la versión de Tools para Windows PowerShell que está disponible en el sitio web de PowerShell Gallery. Le sugerimos que revise cada dos o tres semanas. Support para nuevos comandos y AWS servicios solo está disponible después de actualizar a una versión con ese soporte.

Antes de instalar una versión más reciente de AWSPowerShell. NetCore, desinstale el módulo existente. Cierre todas PowerShell las sesiones abiertas antes de desinstalar el paquete existente. Ejecute el siguiente comando para desinstalar el paquete.

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

Cuando se haya completado la desinstalación del paquete, instale el módulo actualizado ejecutando el siguiente comando.

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

Tras la instalación, ejecute el comando `Import-Module AWSPowerShell.NetCore` para cargar los cmdlets actualizados en la sesión PowerShell .

Actualice las herramientas para Windows PowerShell

Ejecute el `Get-AWSPowerShellVersion` cmdlet para determinar la versión que está ejecutando y compárela con la versión de Herramientas para Windows PowerShell que está disponible en el sitio web de la PowerShell Galería. Le sugerimos que revise cada dos o tres semanas. Support para nuevos comandos y AWS servicios solo está disponible después de actualizar a una versión con ese soporte.

- Si realizó la instalación mediante el cmdlet `Install-Module`, ejecute los siguientes comandos.

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

- Si la instaló mediante un ZIP archivo descargado:
  1. Descargue la versión más reciente del sitio PowerShell web Herramientas para. Compare el número de versión del paquete en el nombre del archivo descargado con el número de versión que obtiene cuando se ejecuta el cmdlet `Get-AWSPowerShellVersion`.
  2. Si el número de la versión de descarga es superior al de la versión que ha instalado, cierre todas las PowerShell consolas de Herramientas para Windows.
  3. Instale la versión más reciente de las Herramientas para Windows PowerShell.

Tras la instalación, ejecútelo `Import-Module AWSPowerShell` para cargar los cmdlets actualizados en su PowerShell sesión. O bien, ejecute la AWS Tools for PowerShell consola personalizada desde el menú Inicio.

Instalación AWS Tools for PowerShell en Linux o macOS

En este tema se proporcionan instrucciones sobre cómo instalarlo AWS Tools for PowerShell en Linux o macOS.

Información general de la configuración

Para instalarlo AWS Tools for PowerShell en un ordenador Linux o macOS, puede elegir entre dos opciones de paquetes:

- **`AWS.Tools`**— La versión modularizada de. AWS Tools for PowerShell Cada AWS servicio está respaldado por su propio módulo pequeño e individual, con módulos de soporte compartidos. `AWS.Tools.Common`
- AWSPowerShell. NetCore— La versión única de módulos grandes de AWS Tools for PowerShell. Todos los AWS servicios son compatibles con este módulo único y grande.

> **Note**
>
> Tenga en cuenta que el módulo individual puede ser demasiado grande para usarlo con funciones de AWS Lambda. En su lugar, utilice la versión modularizada que se muestra arriba.

La configuración de cualquiera de estas versiones en un equipo con Linux o macOS implica las siguientes tareas, que se describen en detalle más adelante en este tema:

1. Instale PowerShell Core 6.0 o una versión posterior en un sistema compatible.
2. Tras instalar PowerShell Core, empiece PowerShell por ejecutarlo `pwsh` en el shell del sistema.
3. Instale una de las dos `AWS.Tools` opciones AWSPowerShell. NetCore.
4. Ejecute el `Import-Module` cmdlet correspondiente para importar el módulo a la sesión PowerShell.
5. Ejecute el AWSDefaultConfiguration cmdlet Initialize- para proporcionar sus credenciales. AWS

Requisitos previos

Para ejecutarlo AWS Tools for PowerShell Core, su equipo debe ejecutar PowerShell Core 6.0 o una versión posterior.

- Para obtener una lista de las versiones de la plataforma Linux compatibles y obtener información sobre cómo instalar la última versión de PowerShell en un equipo basado en Linux, consulte Instalación PowerShell en Linux en el sitio web de Microsoft. Algunos sistemas operativos basados en Linux, como Arch, Kali y Raspbian, no se admiten oficialmente, pero reciben diversos grados de soporte de la comunidad.
- Para obtener información sobre las versiones de macOS compatibles y sobre cómo instalar la última versión de PowerShell en macOS, consulta Instalación PowerShell en macOS en el sitio web de Microsoft.

Instalación de **AWS.Tools** en Linux o macOS

Puede instalar la versión modularizada de AWS Tools for PowerShell en ordenadores que ejecuten PowerShell Core 6.0 o una versión posterior. Para obtener información acerca de cómo instalar PowerShell Core, consulte Instalación de varias versiones de PowerShell en el PowerShell sitio web de Microsoft.

Puede instalar `AWS.Tools` de tres maneras:

- Utilizando los cmdlets del módulo `AWS.Tools.Installer`. Este módulo simplifica la instalación y actualización de otros `AWS.Tools` módulos. `AWS.Tools.Installer`requiere `PowerShellGet` y descarga e instala automáticamente una versión actualizada del mismo. `AWS.Tools.Installer`mantiene sincronizadas automáticamente las versiones de sus módulos. Al instalar o actualizar a una versión más reciente de un módulo, los cmdlets `AWS.Tools.Installer` actualizan automáticamente todos los demás `AWS.Tools` módulos a la misma versión.

  Este método se describe en el procedimiento siguiente.
- Descargando los módulos de `AWS.Tools.zip` y extrayéndolos en uno de los directorios del módulo. Para saber cuáles son los directorios del módulo, puede imprimir el valor de la variable `$Env:PSModulePath`.
- Instalar cada módulo de servicio de la PowerShell Galería mediante el `Install-Module` cmdlet.

Para instalar **AWS.Tools** en Linux o macOS mediante el **AWS.Tools.Installer** módulo

1. Inicie una sesión PowerShell básica ejecutando el siguiente comando.

```
$ pwsh
```

> **Note**
>
> Te recomendamos que no te postules PowerShell como administrador con permisos elevados, excepto cuando lo exija la tarea en cuestión. Esto puede suponer un riesgo para la seguridad y no se atiene al principio de privilegios mínimos.

2. Para instalar el paquete de varios módulos de `AWS.Tools` con el módulo `AWS.Tools.Installer`, ejecute el siguiente comando.

```
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
```

   Si se le notifica que el repositorio no es de confianza, se le preguntará si desea realizar la instalación de todos modos. Introduzca **y** para PowerShell permitir la instalación del módulo. Para evitar que aparezca el mensaje e instalar el módulo sin confiar en el repositorio, puede ejecutar el siguiente comando.

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

3. Ahora puede instalar el módulo para cada servicio que desee utilizar. Por ejemplo, el siguiente comando instala los módulos de Amazon EC2 y Amazon S3. Este comando también instala los módulos dependientes necesarios para que el módulo especificado funcione. Por ejemplo, cuando instala el primer módulo de servicio `AWS.Tools`, también se instala `AWS.Tools.Common`. Se trata de un módulo compartido que requieren todos los módulos AWS de servicio. También elimina las versiones anteriores de los módulos y actualiza otros módulos a la misma versión.

```
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**
>
> El cmdlet `Install-AWSToolsModule` descarga todos los módulos solicitados de un repositorio de `PSRepository` llamado `PSGallery` (https://www.powershellgallery.com/) y considera este repositorio como un origen de confianza. Utilice el comando `Get-PSRepository -Name PSGallery` para obtener más información sobre este repositorio de `PSRepository`.

El comando anterior instala los módulos en los directorios predeterminados del sistema. Los directorios reales dependen de la distribución y la versión del sistema operativo y de la versión PowerShell que tenga instalada. Por ejemplo, si instaló PowerShell 7 en un sistema similar a RHEL, lo más probable es que los módulos predeterminados estén ubicados en `/opt/microsoft/powershell/7/Modules` (o`$PSHOME/Modules`) y los módulos de usuario probablemente estén ubicados en. `~/.local/share/powershell/Modules` Para obtener más información, consulte Instalar PowerShell en Linux en el PowerShell sitio web de Microsoft. Para ver dónde están instalados los módulos, ejecute el siguiente comando:

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

Para instalar otros módulos, ejecute comandos similares con los nombres de módulo correspondientes, tal y como se encuentra en la PowerShell Galería.

Instala AWSPowerShell. NetCore en Linux o macOS

Para actualizar a una versión más reciente de AWSPowerShell. NetCore, siga las instrucciones que se indican enActualización del AWS Tools for PowerShell en Linux o macOS. Desinstale las versiones anteriores de AWSPowerShell. NetCore primero.

Puedes instalarlo AWSPowerShell. NetCore de una de las dos maneras siguientes:

- Descargando el módulo de `AWSPowerShell.NetCore.zip` y extrayéndolo en uno de los directorios del módulo. Para saber cuáles son los directorios del módulo, puede imprimir el valor de la variable `$Env:PSModulePath`.
- Instalación desde la PowerShell Galería mediante el `Install-Module` cmdlet tal y como se describe en el siguiente procedimiento.

Para instalar. AWSPowerShell NetCore en Linux o macOS mediante el cmdlet Install-Module

Inicie una sesión de PowerShell Core ejecutando el siguiente comando.

```
$ pwsh
```

> **Note**
>
> Te recomendamos que no empieces PowerShell por correr `sudo pwsh` para correr PowerShell con derechos de administrador elevados. Esto puede suponer un riesgo para la seguridad y no se atiene al principio de privilegios mínimos.

Para instalar el AWSPowerShell. NetCore paquete de un solo módulo de la PowerShell Galería, ejecute el siguiente comando.

```
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
```

Si se le notifica que el repositorio no es de confianza, se le preguntará si desea realizar la instalación de todos modos. Introduzca **y** para PowerShell permitir la instalación del módulo. Para evitar que aparezca el mensaje que indica que el repositorio no es de confianza, puede ejecutar el siguiente comando.

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

No tiene que ejecutar este comando como root, a menos que desee instalarlo AWS Tools for PowerShell para todos los usuarios de un equipo. Para ello, ejecute el siguiente comando en una PowerShell sesión con la que haya empezado`sudo pwsh`.

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

Ejecución de scripts

El comando `Set-ExecutionPolicy` no está disponible en los sistemas que no son Windows. Puede ejecutar`Get-ExecutionPolicy`, lo que demuestra que la configuración de política de ejecución predeterminada en PowerShell Core que se ejecuta en sistemas que no son Windows es`Unrestricted`. Para obtener más información, vea Acerca de las directivas de ejecución en el sitio web de Microsoft Technet.

Como `PSModulePath` incluye la ubicación del directorio del AWS módulo, el `Get-Module -ListAvailable` cmdlet muestra el módulo que ha instalado.

`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
```

Configure una PowerShell consola para usar el AWS Tools for PowerShell Core (. AWSPowerShell NetCore (Solo)

PowerShell Por lo general, Core carga los módulos automáticamente cada vez que se ejecuta un cmdlet en el módulo. Pero esto no funciona para. AWSPowerShell NetCore debido a su gran tamaño. Para empezar a correr AWSPowerShell. NetCore cmdlets, primero debe ejecutar el `Import-Module AWSPowerShell.NetCore` comando. Esto no es necesario en los cmdlets de los módulos `AWS.Tools`.

Inicialice su sesión PowerShell

Cuando inicie PowerShell en un sistema basado en Linux o macOS después de haber instalado el AWS Tools for PowerShell, debe ejecutar Initialize- AWSDefaultConfiguration para especificar qué clave de acceso utilizar. AWS Para obtener más información acerca de `Initialize-AWSDefaultConfiguration`, consulte Uso de credenciales de AWS.

> **Note**
>
> En versiones anteriores (anteriores a la 3.3.96.0) del, este cmdlet recibía el nombre. AWS Tools for PowerShell`Initialize-AWSDefaults`

Control de versiones

AWS publica nuevas versiones del AWS Tools for PowerShell periódicamente para admitir nuevos servicios y características. AWS Para determinar la versión AWS Tools for PowerShell que ha instalado, ejecute el AWSPowerShellVersion cmdlet Get-.

```
PS > Get-AWSPowerShellVersion

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

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

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

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

Para ver una lista de los AWS servicios compatibles en la versión actual de las herramientas, agregue el `-ListServiceVersionInfo` parámetro a un cmdlet Get- AWSPowerShellVersion.

Para determinar la versión PowerShell que está ejecutando, escriba `$PSVersionTable` para ver el contenido de la variable `$PSVersionTable` automática.

```
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
```

Actualización del AWS Tools for PowerShell en Linux o macOS

Periódicamente, a medida que AWS Tools for PowerShell se publiquen versiones actualizadas del, debes actualizar la versión que estás ejecutando localmente.

Actualice los módulos modularizados **`AWS.Tools`**

Para actualizar `AWS.Tools` los módulos a la última versión, ejecute el siguiente comando:

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

Este comando actualiza todos los módulos `AWS.Tools` que hay instalados actualmente y, en los módulos que se actualizaron correctamente, elimina las versiones anteriores.

> **Note**
>
> El cmdlet `Update-AWSToolsModule` descarga todos los módulos de un repositorio de `PSRepository` llamado `PSGallery` (https://www.powershellgallery.com/) y considera este repositorio como un origen de confianza. Utilice el comando `Get-PSRepository -Name PSGallery` para obtener más información sobre este repositorio de `PSRepository`.

Actualice las herramientas de PowerShell Core

Ejecute el `Get-AWSPowerShellVersion` cmdlet para determinar la versión que está ejecutando y compárela con la versión de Tools para Windows PowerShell que está disponible en el sitio web de PowerShell Gallery. Le sugerimos que revise cada dos o tres semanas. Support para nuevos comandos y AWS servicios solo está disponible después de actualizar a una versión con ese soporte.

Antes de instalar una versión más reciente de AWSPowerShell. NetCore, desinstale el módulo existente. Cierre todas PowerShell las sesiones abiertas antes de desinstalar el paquete existente. Ejecute el siguiente comando para desinstalar el paquete.

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

Cuando se haya completado la desinstalación del paquete, instale el módulo actualizado ejecutando el siguiente comando.

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

Tras la instalación, ejecute el comando `Import-Module AWSPowerShell.NetCore` para cargar los cmdlets actualizados en la sesión PowerShell .

Información relacionada

- Comenzar a utilizar la AWS Tools for Windows PowerShell
- Trabaje con AWS los servicios del AWS Tools for PowerShell

Migración de la AWS Tools for PowerShell versión 3.3 a la versión 4

AWS Tools for PowerShell la versión 4 es una actualización compatible con versiones anteriores de la versión 3.3. AWS Tools for PowerShell Agrega mejoras significativas a la vez que mantiene el comportamiento existente del cmdlet.

Los scripts existentes deberían seguir funcionando después de actualizar a la nueva versión, pero recomendamos que los pruebe a fondo antes de actualizar los entornos de producción.

En esta sección, se describen los cambios y se explica cómo pueden afectar a los scripts.

Nueva versión de **`AWS.Tools`** dividida completamente en módulos

AWSPowerShellEl. NetCore y AWSPowerShell los paquetes eran «monolíticos». Esto significaba que todos los AWS servicios eran compatibles con el mismo módulo, lo que lo hacía muy grande y crecía a medida que se añadían nuevos AWS servicios y funciones. El nuevo `AWS.Tools` paquete está dividido en módulos más pequeños que le ofrecen la flexibilidad de descargar e instalar solo los que necesite para los AWS servicios que utilice. El paquete incluye un módulo `AWS.Tools.Common` compartido que todos los demás módulos necesitan y un módulo `AWS.Tools.Installer` que simplifica la instalación, actualización y eliminación de módulos, en función de las necesidades.

Esto también permite importar automáticamente cmdlets en la primera llamada sin tener que invocar primero a `Import-module`. Sin embargo, para interactuar con lo asociado. NETantes de llamar a un cmdlet, debe seguir llamando `Import-Module` para PowerShell informar sobre lo relevante. NETtipos.

Por ejemplo, el comando siguiente contiene una referencia a `Amazon.EC2.Model.Filter`. Este tipo de referencia no puede desencadenar la importación automática, por lo que será necesario llamar primero a `Import-Module` o el comando no se ejecutará correctamente.

```
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
```

Nuevo cmdlet **`Get-AWSService`**

Para ayudarle a descubrir los nombres de los módulos de cada AWS servicio del `AWS.Tools` conjunto de módulos, puede usar el `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
  ...
```

Nuevo parámetro **`-Select`** para controlar el objeto devuelto por un cmdlet

La mayoría de los cmdlets de la versión 4 admiten el nuevo parámetro `-Select`. Cada cmdlet llama al AWS servicio APIs por usted mediante. AWS SDK for .NET A continuación, el AWS Tools for PowerShell cliente convierte la respuesta en un objeto que puede utilizar en sus PowerShell scripts y canalizarla a otros comandos. A veces, el PowerShell objeto final tiene más campos o propiedades en la respuesta original de los que necesita y, en otras ocasiones, es posible que desee que el objeto

incluya campos o propiedades de la respuesta que no aparecen de forma predeterminada. El `-Select` parámetro le permite especificar lo que se incluye en. NETobjeto devuelto por el cmdlet.

Por ejemplo, el Get-S3Objectcmdlet invoca la operación Amazon S3. SDK ListObjects Esa operación devuelve un objeto. ListObjectsResponse Sin embargo, de forma predeterminada, el `Get-S3Object` cmdlet devuelve al usuario únicamente el `S3Objects` elemento de la SDK PowerShell respuesta. En el siguiente ejemplo, ese objeto es una matriz con dos elementos.

```
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
```

En la AWS Tools for PowerShell versión 4, puede especificar que se devuelva `-Select *` el archivo completo. NETobjeto de respuesta devuelto por la SDK API llamada.

```
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      :
```

También puede especificar la ruta de acceso a una propiedad anidada específica que desee. En el ejemplo siguiente, solo se devuelve la propiedad `Key` de cada elemento de la matriz `S3Objects`.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select S3Objects.Key
file1.txt
file2.txt
```

En determinadas situaciones, puede ser útil devolver un parámetro de cmdlet. Puede hacerlo con `-Select ^ParameterName`. Esta función suplanta al parámetro `-PassThru`, que, aunque sigue estando disponible, se ha quedado obsoleto.

```
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
```

En el tema de referencia de cada cmdlet, se indica si se admite el parámetro `-Select`.

Limitación más coherente del número de elementos de la salida

Las versiones anteriores de AWS Tools for PowerShell permitían utilizar el `-MaxItems` parámetro para especificar el número máximo de objetos devueltos en la salida final.

Este comportamiento se ha eliminado en `AWS.Tools`.

Este comportamiento está obsoleto en AWSPowerShell. NetCore y AWSPowerShell, y se eliminará de esas versiones en una versión futura.

Si el servicio subyacente API admite un `MaxItems` parámetro, sigue disponible y funciona según lo API especificado. Sin embargo, no dispondrá del comportamiento adicional que permite limitar el número de elementos devueltos en el resultado del cmdlet.

Para limitar el número de elementos devueltos en la salida final, canalice la salida al `Select-Object` cmdlet y especifique el `-First` *n* parámetro, donde *n* es el número máximo de elementos que se van a incluir en el resultado final.

```
PS > Get-S3ObjectV2 -BucketName amzn-s3-demo-bucket -Select S3Objects.Key | select -
first 2
file1.txt
file2.txt
```

No todos los AWS servicios son compatibles `-MaxItems` de la misma manera, por lo que se elimina esa incoherencia y los resultados inesperados que a veces se producían. Además, `-MaxItems` combinado con el nuevo parámetro `-Select` podría dar lugar en ocasiones a resultados confusos.

Parámetros de flujo más fáciles de usar

Los parámetros de tipo `Stream` o `byte[]` ahora pueden aceptar valores `string`, `string[]` o `FileInfo`.

Por ejemplo, puede utilizar cualquiera de los siguientes ejemplos.

```
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 convierte todas las cadenas para que `byte[]` utilicen la codificación UTF-8.

Ampliación de la canalización por nombre de propiedad

Para que la experiencia del usuario sea más coherente, ahora puede pasar la entrada de la canalización especificando el nombre de propiedad de cualquier parámetro.

En el ejemplo siguiente, creamos un objeto personalizado con propiedades cuyos nombres coinciden con los nombres de parámetro del cmdlet de destino. Cuando se ejecuta el cmdlet, automáticamente consume esas propiedades como parámetros.

```
PS > [pscustomobject] @{ BucketName='amzn-s3-demo-bucket'; Key='file1.txt';
 PartNumber=1 } | Get-S3ObjectMetadata
```

> **Note**
>
> Algunas propiedades lo admitían en versiones anteriores de AWS Tools for PowerShell. La versión 4 hace que este comportamiento sea más coherente, ya que está habilitado en todos los parámetros.

Parámetros comunes estáticos

Para mejorar la coherencia en la versión 4.0 de AWS Tools for PowerShell, todos los parámetros son estáticos.

En versiones anteriores de AWS Tools for PowerShell, algunos parámetros comunes, como`AccessKey`,, o`SecretKey`,`ProfileName`,`Region`, eran dinámicos, mientras que todos los demás parámetros eran estáticos. Esto podría crear problemas porque PowerShell vincula los parámetros estáticos antes que los dinámicos. Por ejemplo, supongamos que antes ejecutaba el siguiente comando.

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

Las versiones anteriores de PowerShell vinculaban el valor `us-west-2` al parámetro `-RegionName` estático en lugar del `-Region` dinámico. Probablemente, esto podría confundir a los usuarios.

`AWS.Tools` declara y aplica parámetros obligatorios

Todos los módulos de `AWS.Tools.*` ahora declaran y aplican parámetros de cmdlet obligatorios. Cuando un AWS servicio declara que uno de los parámetros de an API es obligatorio, PowerShell le pide el parámetro del cmdlet correspondiente si no lo ha especificado. Esto solo es aplicable a `AWS.Tools`. Para garantizar la compatibilidad con versiones anteriores, esto no se aplica a. AWSPowerShell NetCore o AWSPowerShell.

Todos los parámetros pueden ser nulos

Ahora puede asignar `$null` a los parámetros de tipo de valor (números y fechas). Este cambio no debería afectar a los scripts existentes. Esto le permitirá omitir el mensaje sobre la obligatoriedad de un parámetro. Los parámetros obligatorios solo se aplican forzosamente en `AWS.Tools`.

Si ejecuta el siguiente ejemplo utilizando la versión 4, se omitirá eficazmente la validación del lado del cliente porque se proporcionará un «valor» para cada parámetro obligatorio. Sin embargo, la llamada EC2 API de servicio de Amazon falla porque el AWS servicio aún requiere esa información.

```
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
```

Eliminación de características que ya estaban obsoletas

Las siguientes funciones quedaron obsoletas en las versiones anteriores de la versión 4 AWS Tools for PowerShell y se eliminaron en la versión 4:

- El parámetro `-Terminate` se ha eliminado del cmdlet `Stop-EC2Instance`. En su lugar, use `Remove-EC2Instance`.
- Se quitó el `-ProfileName` parámetro del AWSCredential cmdlet Clear-. En su lugar, use `Remove-AWSCredentialProfile`.
- Los cmdlets `Import-EC2Instance` y `Import-EC2Volume` se han eliminado.

Comenzar a utilizar la AWS Tools for Windows PowerShell

Algunos de los temas de esta sección describen aspectos básicos del uso de las Herramientas para Windows PowerShell después de que haya instalado las herramientas. Por ejemplo, se explica cómo especificar qué credenciales y región de AWS deben usar las Herramientas para Windows PowerShell cuando interactúan con AWS.

En otros temas de esta sección se proporciona información sobre las formas avanzadas de configurar las herramientas, el entorno y los proyectos.

Temas

- Configure la autenticación de herramientas con AWS
- Especificar AWS regiones
- Configure la identidad federada con AWS Tools for PowerShell
- Detección y alias de cmdlet
- Canalización y $AWSHistory
- Resolución de credencial y perfil
- Información adicional acerca de los usuarios y los roles
- Uso de credenciales heredadas

Configure la autenticación de herramientas con AWS

Debes establecer cómo se autentica tu código AWS al desarrollar con. Servicios de AWS Existen diferentes formas de configurar el acceso programático a AWS los recursos, según el entorno y el AWS acceso del que dispongas.

Para ver los distintos métodos de autenticación de las Herramientas PowerShell, consulte Autenticación y acceso en la Guía de referencia de AWS los SDK y las herramientas.

En este tema se presupone que un nuevo usuario se está desarrollando a nivel local, que su empresa no le ha proporcionado un método de autenticación y que lo utilizará AWS IAM Identity Center para obtener credenciales temporales. Si el entorno no se basa en estos supuestos, es posible que parte de la información de este tema no se aplique a su caso o que ya se le haya proporcionado parte de la información.

La configuración de este entorno requiere varios pasos, que se resumen de la siguiente manera:

1. Habilitar y configurar el Centro de identidades de IAM
2. Configure las herramientas PowerShell para utilizar el IAM Identity Center.
3. Inicie una sesión en el portal de AWS acceso

Habilitar y configurar el Centro de identidades de IAM

Para usarlo AWS IAM Identity Center, primero debe estar habilitado y configurado. Para obtener más información sobre cómo hacerlo PowerShell, consulte el paso 1 del tema sobre la autenticación del IAM Identity Center de la Guía de referencia de herramientas y AWS SDK. En concreto, siga las instrucciones necesarias en No he establecido el acceso a través del Centro de identidades de IAM.

Configure las herramientas PowerShell para utilizar el IAM Identity Center.

> **Note**
>
> A partir de la versión 4.1.538 de Tools for PowerShell, el método recomendado para configurar las credenciales de SSO e iniciar una sesión en el portal de AWS acceso consiste en utilizar los `Invoke-AWSSSOLogin`cmdlets y, tal `Initialize-AWSSSOConfiguration`y como se describe en este tema. Si no tiene acceso a esa versión de Tools for PowerShell (o posterior) o no puede usar esos cmdlets, puede seguir realizando estas tareas mediante el. AWS CLI Para obtener más información sobre cómo hacerlo, consulte. Utilice el AWS CLI para iniciar sesión en el portal

El siguiente procedimiento actualiza el AWS `config` archivo compartido con la información de inicio de sesión único que la herramienta PowerShell utiliza para obtener credenciales temporales. Como consecuencia de este procedimiento, también se inicia una sesión en el portal de AWS acceso. Si el `config` archivo compartido ya contiene información sobre el inicio de sesión único y solo quiere saber cómo iniciar una sesión en el portal de acceso mediante las Herramientas PowerShell, consulte la siguiente sección de este tema. Inicie una sesión en el portal de AWS acceso

1. Si aún no lo ha hecho, ábralo PowerShell e instálelo según AWS Tools for PowerShell corresponda a su sistema operativo y entorno, incluidos los cmdlets más comunes. Para obtener información acerca de cómo hacerlo, consulte Instalación de AWS Tools for PowerShell.

   Por ejemplo, si instala la versión modularizada de las Herramientas para PowerShell Windows, lo más probable es que ejecute comandos similares a los siguientes:

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

2. Ejecute el siguiente comando de la . Sustituya los valores de propiedad del ejemplo por valores de la configuración del Centro de identidades de IAM. Para obtener información sobre estas propiedades y cómo encontrarlas, consulte la configuración del proveedor de credenciales del IAM Identity Center en la Guía de referencia de herramientas y AWS SDK.

```
$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
```

   Como alternativa, puede utilizar el cmdlet por sí solo y la herramienta de herramientas PowerShell le solicitará los valores de las propiedades. `Initialize-AWSSSOConfiguration`

   Consideraciones sobre determinados valores de propiedad:

   - Si simplemente ha seguido las instrucciones para activar y configurar IAM Identity Center, el valor de `-RoleName` podría ser`PowerUserAccess`. Sin embargo, si ha creado un conjunto de permisos del Centro de Identidad de IAM específicamente para PowerShell trabajar, utilícelo en su lugar.
   - Asegúrese de usar el Región de AWS lugar donde configuró el Centro de identidades de IAM.

3. En este punto, el AWS `config` archivo compartido contiene un perfil llamado `my-sso-profile` con un conjunto de valores de configuración a los que se puede hacer referencia en las Herramientas para PowerShell. Para encontrar la ubicación de este archivo, consulte Ubicación de los archivos compartidos en la Guía de referencia de las herramientas y los SDK de AWS.

   The Tools for PowerShell utiliza el proveedor del token de inicio de sesión único del perfil para adquirir las credenciales antes de enviar las solicitudes a AWSél. El `sso_role_name` valor, que es un rol de IAM conectado a un conjunto de permisos del Centro de Identidad de IAM, debería permitir el acceso a los Servicios de AWS utilizados en la aplicación.

En el siguiente ejemplo, se muestra el perfil que se creó mediante el comando que se muestra arriba. Es posible que algunos de los valores de las propiedades y su orden difieran en su perfil real. La `sso-session` propiedad del perfil hace referencia a la sección denominada`my-sso-session`, que contiene la configuración para iniciar una sesión en el portal de AWS acceso.

```
[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. Si ya tiene una sesión activa en el portal de AWS acceso, las Herramientas PowerShell le informarán de que ya ha iniciado sesión.

   Si ese no es el caso, la herramienta de herramientas PowerShell intentará abrir automáticamente la página de autorización del SSO en el navegador web predeterminado. Sigue las instrucciones del navegador, que pueden incluir un código de autorización del SSO, un nombre de usuario y una contraseña, y permiso para acceder a AWS IAM Identity Center las cuentas y conjuntos de permisos.

   Las herramientas de PowerShell le informan de que el inicio de sesión único se ha realizado correctamente.

Inicie una sesión en el portal de AWS acceso

Antes de ejecutar los comandos de acceso Servicios de AWS, necesita una sesión activa en el portal de AWS acceso para que las herramientas PowerShell puedan utilizar la autenticación del IAM Identity Center para resolver las credenciales. Para iniciar sesión en el portal de AWS acceso, ejecute el siguiente comando PowerShell, donde `-ProfileName my-sso-profile` aparece el nombre del perfil que se creó en el `config` archivo compartido al seguir el procedimiento de la sección anterior de este tema.

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

Si ya tiene una sesión activa en el portal de AWS acceso, las Herramientas de PowerShell le informarán de que ya ha iniciado sesión.

Si ese no es el caso, la herramienta de herramientas PowerShell intentará abrir automáticamente la página de autorización del SSO en el navegador web predeterminado. Sigue las instrucciones del navegador, que pueden incluir un código de autorización del SSO, un nombre de usuario y una contraseña, y permiso para acceder a AWS IAM Identity Center las cuentas y conjuntos de permisos.

Las herramientas de PowerShell le informan de que el inicio de sesión único se ha realizado correctamente.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente comando después de instalar o importar el `AWS.Tools.SecurityToken` módulo, según sea necesario.

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

La respuesta al `Get-STSCallerIdentity` cmdlet indica la cuenta y el conjunto de permisos del IAM Identity Center configurados en el archivo compartido. `config`

Ejemplo

A continuación se muestra un ejemplo de cómo utilizar el Centro de identidades de IAM con las herramientas para. PowerShell Asume lo siguiente:

- Ha habilitado el Centro de identidades de IAM y lo ha configurado como se ha descrito anteriormente en este tema. Las propiedades del SSO se encuentran en el `my-sso-profile` perfil, que se configuró anteriormente en este tema.
- Al iniciar sesión a través de los `Invoke-AWSSSOLogin` cmdlets `Initialize-AWSSSOConfiguration` o, el usuario tiene al menos permisos de solo lectura para Amazon S3.
- Algunos buckets de S3 están disponibles para que los vea ese usuario.

Instale o importe el `AWS.Tools.S3` módulo según sea necesario y, a continuación, utilice el siguiente PowerShell comando para mostrar una lista de los buckets de S3.

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

Información adicional

- Para obtener más opciones de autenticación para las herramientas PowerShell, como el uso de perfiles y variables de entorno, consulte el capítulo de configuración de la Guía de referencia de herramientas y AWS SDK.
- Algunos comandos requieren que se especifique una AWS región. Hay varias formas de hacerlo, incluidas la opción de `-Region` cmdlet, el `[default]` perfil y la variable de `AWS_REGION` entorno. Para obtener más información, consulte Especificar AWS regiones esta guía y AWS la región en la Guía de referencia de AWS SDK y herramientas.
- Para obtener más información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.
- Para crear AWS credenciales de corta duración, consulte Credenciales de seguridad temporales en la Guía del usuario de IAM.
- Para obtener más información sobre otros proveedores de credenciales, consulte los proveedores de credenciales estandarizados en la Guía de referencia de herramientas y AWS SDK.

Temas

- Utilice el AWS CLI para iniciar sesión en el portal

Utilice el AWS CLI para iniciar sesión en el portal

A partir de la versión 4.1.538 de Tools for PowerShell, el método recomendado para configurar las credenciales de SSO e iniciar una sesión en el portal de AWS acceso consiste en utilizar los `Invoke-AWSSSOLogin`cmdlets y, tal `Initialize-AWSSSOConfiguration`y como se describe en. Configure la autenticación de herramientas con AWS Si no tiene acceso a esa versión de las Herramientas PowerShell (o posterior) o no puede usar esos cmdlets, puede seguir realizando estas tareas mediante el. AWS CLI

Configure las herramientas PowerShell para utilizar el Centro de identidades de IAM a través del. AWS CLI

Si aún no lo ha hecho, asegúrese de habilitar y configurar el Centro de identidades de IAM antes de continuar.

La información sobre cómo configurar las herramientas PowerShell para utilizar el Centro de Identidad de IAM AWS CLI se encuentra en el paso 2 del tema sobre la autenticación del Centro

de Identidad de IAM de la Guía de referencia de herramientas y AWS SDK. Tras completar esta configuración, el sistema debe contener los siguientes elementos:

- El AWS CLI, que se utiliza para iniciar una sesión en el portal de AWS acceso antes de ejecutar la aplicación.
- El AWS `config` archivo compartido que contiene un `[default]`perfil con un conjunto de valores de configuración a los que se puede hacer referencia en las Herramientas para PowerShell. Para encontrar la ubicación de este archivo, consulte Ubicación de los archivos compartidos en la Guía de referencia de las herramientas y los SDK de AWS. The Tools for PowerShell utiliza el proveedor del token de inicio de sesión único del perfil para adquirir las credenciales antes de enviar las solicitudes a AWSél. El `sso_role_name` valor, que es un rol de IAM conectado a un conjunto de permisos del Centro de Identidad de IAM, debería permitir el acceso a los Servicios de AWS utilizados en la aplicación.

  En el siguiente `config` archivo de ejemplo, se muestra un `[default]` perfil configurado con un proveedor de tokens de SSO. La configuración `sso_session` del perfil hace referencia a la sección `sso-session` nombrada. La `sso-session` sección contiene la configuración para iniciar una sesión en el portal de AWS acceso.

```
[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**
>
> La PowerShell sesión debe tener los siguientes módulos instalados e importados para que la resolución del SSO pueda funcionar:
>
> - `AWS.Tools.SSO`
> - `AWS.Tools.SSOOIDC`

Si utilizas una versión anterior de Tools for PowerShell y no dispones de estos módulos, aparecerá un error similar al siguiente: «No se ha podido encontrar el archivo Assembly AWSSDK .SSOOIDC...».

Inicie una sesión en el portal de acceso AWS

Antes de ejecutar los comandos de acceso Servicios de AWS, necesita una sesión activa en el portal de AWS acceso para que las herramientas de Windows PowerShell puedan utilizar la autenticación del IAM Identity Center para resolver las credenciales. En función de la duración de las sesiones configuradas, el acceso eventualmente caducará y las Herramientas para Windows PowerShell detectarán un error de autenticación. Para iniciar sesión en el portal de AWS acceso, ejecute el siguiente comando en AWS CLI.

```
aws sso login
```

Como está utilizando el `[default]` perfil, no necesita llamar al comando con la `--profile` opción. Si la configuración de su proveedor de token de SSO utiliza un perfil con nombre, el comando lo hará `aws sso login --profile named-profile` en su lugar. Para obtener más información sobre los perfiles con nombre asignado, consulta la sección Perfiles de la Guía de referencia de AWS SDK y herramientas.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente AWS CLI comando (teniendo en cuenta lo mismo para el perfil nombrado):

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

La respuesta a este comando debe indicar la cuenta y el conjunto de permisos del Centro de identidades de IAM configurados en el archivo compartido `config`.

Note

Si ya tiene una sesión activa en el portal de AWS acceso y la ejecuta`aws sso login`, no tendrá que proporcionar credenciales.

Es posible que el proceso de inicio de sesión le pida que permita el AWS CLI acceso a sus datos. Como AWS CLI se basa en el SDK para Python, los mensajes de permiso pueden contener variaciones del `botocore` nombre.

Ejemplo

A continuación se muestra un ejemplo de cómo utilizar el Centro de identidades de IAM con las herramientas para PowerShell. Asume lo siguiente:

- Ha habilitado el Centro de identidades de IAM y lo ha configurado como se ha descrito anteriormente en este tema. Las propiedades del SSO se encuentran en el perfil `[default]`.
- Al iniciar sesión AWS CLI mediante el uso de`aws sso login`, ese usuario tiene al menos permisos de solo lectura para Amazon S3.
- Algunos buckets de S3 están disponibles para que los vea ese usuario.

Utilice los siguientes PowerShell comandos para mostrar una lista de los buckets de S3:

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule S3
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SSO, SSOOIDC

# 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.SSO
Import-Module AWS.Tools.SSOOIDC

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

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

Como se ha mencionado anteriormente, dado que utiliza el `[default]` perfil, no necesita llamar al `Get-S3Bucket` cmdlet con la opción. `-ProfileName` Si la configuración del proveedor de token de SSO utiliza un perfil con nombre, el comando es `Get-S3Bucket -ProfileName` *named-*

profile. Para obtener más información sobre los perfiles con nombre asignado, consulte la sección Perfiles de la Guía de AWS referencia de SDK y herramientas.

Información adicional

- Para obtener más opciones de autenticación para las herramientas PowerShell, como el uso de perfiles y variables de entorno, consulte el capítulo de configuración de la Guía de referencia de AWS los SDK y las herramientas.
- Algunos comandos requieren que se especifique una AWS región. Hay varias formas de hacerlo, incluidas la opción de `-Region` cmdlet, el `[default]` perfil y la variable de `AWS_REGION` entorno. Para obtener más información, consulte Especificar AWS regiones esta guía y AWS la región en la Guía de referencia de AWS SDK y herramientas.
- Para obtener más información sobre las prácticas recomendadas, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.
- Para crear AWS credenciales de corta duración, consulte Credenciales de seguridad temporales en la Guía del usuario de IAM.
- Para obtener más información sobre otros proveedores de credenciales, consulte los proveedores de credenciales estandarizados en la Guía de referencia de herramientas y AWS SDK.

Especificar AWS regiones

Hay dos formas de especificar la AWS región que se va a utilizar al ejecutar AWS Tools for PowerShell comandos:

- Utilice el parámetro común `-Region` en comandos individuales.
- Utilice el comando `Set-DefaultAWSRegion` para establecer una región predeterminada para todos los comandos.

Muchos AWS cmdlets fallan si las Herramientas de Windows no PowerShell pueden determinar qué región usar. Las excepciones incluyen los cmdlets de Amazon S3, Amazon SES y AWS Identity and Access Management, que se establecen automáticamente en un punto de enlace global de forma predeterminada.

Para especificar la región de un solo comando AWS

Agregue el parámetro `-Region` al comando, como el siguiente.

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

Para establecer una región predeterminada para todos los comandos AWS CLI de la sesión actual

En la PowerShell línea de comandos, escriba el siguiente comando.

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

> **Note**
>
> Este valor persiste únicamente durante la sesión actual. Para aplicar la configuración a todas las PowerShell sesiones, añada este comando a su PowerShell perfil tal como lo hizo con el `Import-Module` comando.

Para ver la región predeterminada actual de todos los comandos AWS CLI

En la PowerShell línea de comandos, escriba el siguiente comando.

```
PS > Get-DefaultAWSRegion

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

Para borrar la región predeterminada actual para todos los comandos AWS CLI

En la PowerShell línea de comandos, escriba el siguiente comando.

```
PS > Clear-DefaultAWSRegion
```

Para ver una lista de todas las AWS regiones disponibles

En la PowerShell línea de comandos, escriba el siguiente comando. La tercera columna del resultado de ejemplo identifica qué región es la predeterminada para su sesión actual.

```
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**
>
> Es posible que se admitan algunas regiones, pero que no aparezcan en los resultados del cmdlet `Get-AWSRegion`. Por ejemplo, esto a veces también es válido para las regiones que aún no son globales. Si no puede especificar una región cuando agrega el parámetro `-Region` a un comando, intente especificar la región en un punto de enlace personalizado, como se muestra en la siguiente sección.

Especificación de un punto de enlace personalizado o que no sea estándar

Especifique un punto final personalizado como URL añadiendo el parámetro `-EndpointUrl` común al PowerShell comando Tools for Windows, en el siguiente formato de ejemplo.

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

A continuación, se muestra un ejemplo con el cmdlet `Get-EC2Instance`. En este ejemplo, el punto de enlace personalizado se encuentra en la región `us-west-2` o EE. UU. Oeste (Oregón), pero puede utilizar cualquier otra región de AWS admitida, incluidas las que no aparecen cuando se ejecuta `Get-AWSRegion`.

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

Información adicional

Para obtener información adicional sobre AWS las regiones, consulte AWS la sección Región en la Guía de referencia de AWS SDK y herramientas.

Configure la identidad federada con AWS Tools for PowerShell

Para permitir que los usuarios de su organización accedan a AWS los recursos, debe configurar un método de autenticación estándar y repetible para garantizar la seguridad, la auditabilidad, el cumplimiento y la capacidad de permitir la separación de funciones y cuentas. Si bien es habitual ofrecer a los usuarios la posibilidad de acceder AWS APIs, sin un API acceso federado, también tendrías que crear AWS Identity and Access Management (IAM) usuarios, lo que va en contra de la finalidad de utilizar la federación. En este tema se describe la compatibilidad SAML (lenguaje de marcado de aserciones de seguridad) con la solución AWS Tools for PowerShell que facilita el acceso federado.

SAMLel soporte en el AWS Tools for PowerShell le permite proporcionar a sus usuarios un acceso federado a los servicios. AWS SAMLes un formato XML basado en estándares abiertos para transmitir datos de autenticación y autorización de usuarios entre servicios; en particular, entre un proveedor de identidad (como los Servicios de federación de Active Directory) y un proveedor de servicios (como AWS). Para obtener más información SAML y cómo funciona, consulte Wikipedia o las especificaciones SAML técnicas SAMLen el sitio web de la Organización para el Avance de los Estándares de Información Estructurada (OASIS). SAMLel soporte en el AWS Tools for PowerShell es compatible con SAML 2.0.

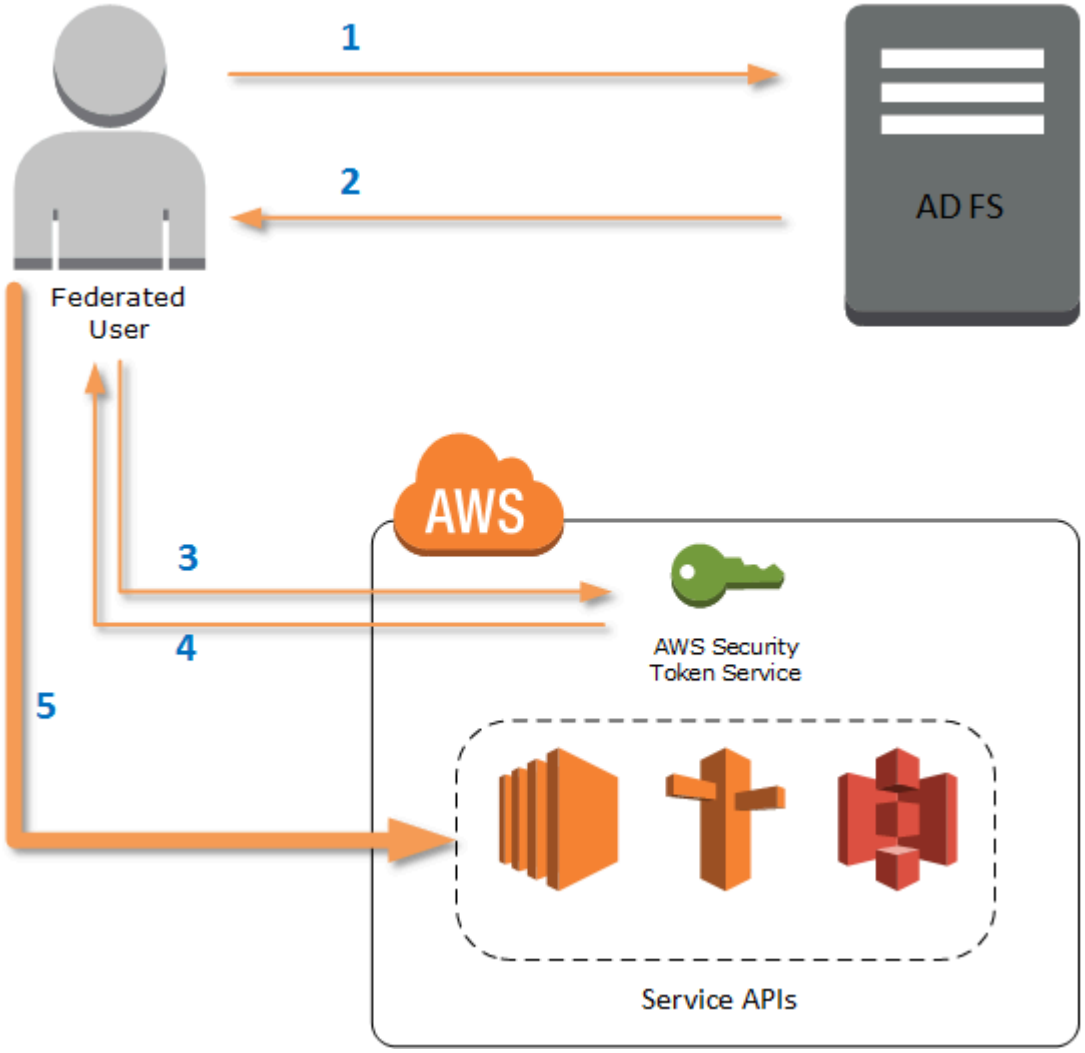
Requisitos previos

Debe disponer de lo siguiente antes de intentar utilizar el SAML soporte por primera vez.

- Una solución de identidad federada que esté integrada de forma correcta a su cuenta de AWS para poder tener acceso a la consola usando solo las credenciales de su organización. Para obtener más información acerca de cómo hacerlo específicamente para los Servicios de federación de Active Directory, consulte Acerca de la federación SAML 2.0 en la Guía del IAM usuario y la entrada del blog titulada Cómo habilitar la federación para AWS usar Windows Active Directory, AD FS y SAML 2.0. Aunque la entrada de blog explica el procedimiento para AD FS 2.0, los pasos son similares si ejecuta AD FS 3.0.
- La versión 3.1.31.0 o posterior de la AWS Tools for PowerShell instalada en la estación de trabajo local.

Cómo obtiene un usuario con identidad federada el acceso federado al servicio AWS APIs

El siguiente proceso describe, de forma general, cómo AD FS federa a un usuario de Active Directory (AD) para obtener acceso a los recursos. AWS

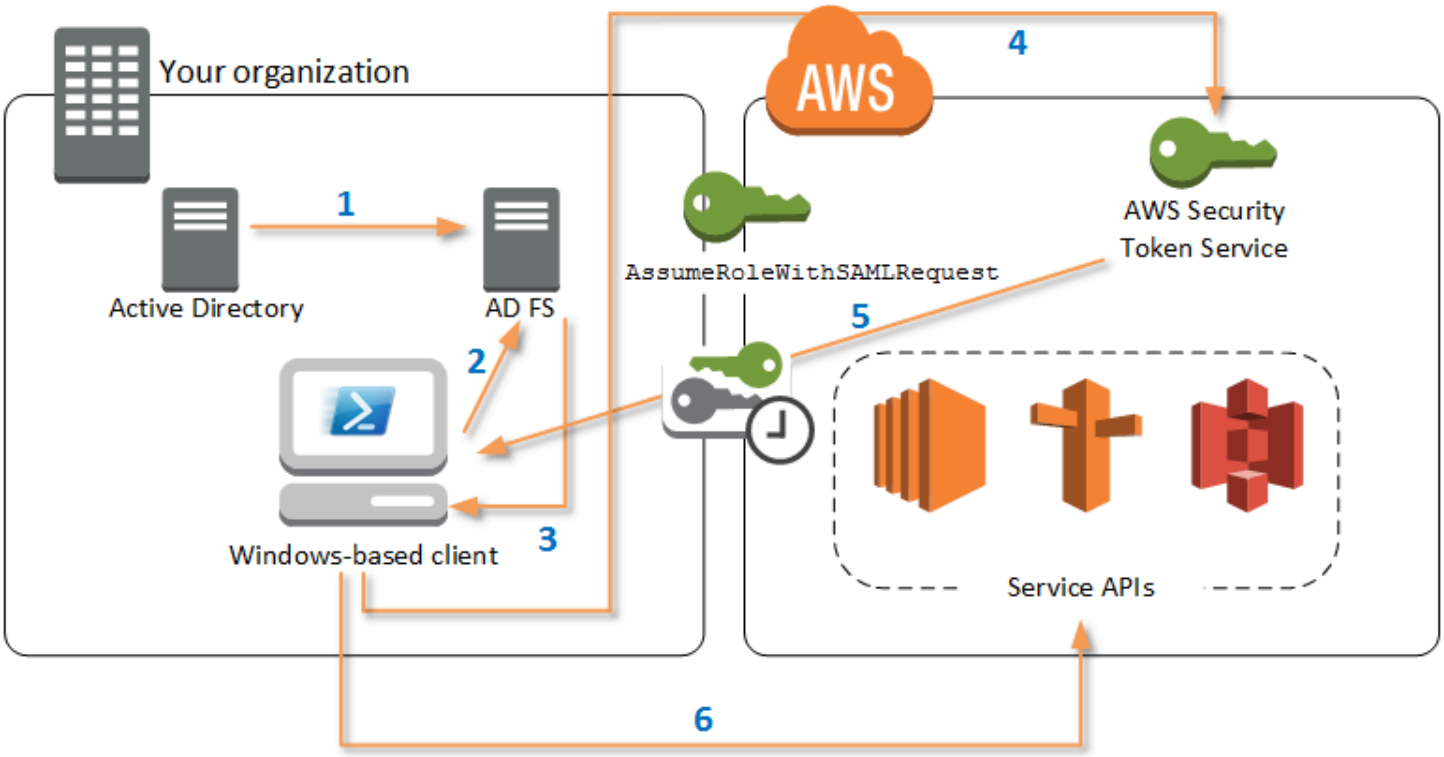


1. El cliente en el equipo del usuario federado se autentica en AD FS.
2. Si la autenticación se realiza correctamente, AD FS envía una SAML afirmación al usuario.
3. El cliente del usuario envía la SAML aserción a AWS Security Token Service (STS) como parte de una SAML solicitud de federación.
4. STSdevuelve una SAML respuesta que contiene credenciales AWS temporales para un rol que el usuario puede asumir.

5. El usuario accede AWS al servicio APIs al incluir esas credenciales temporales en la solicitud realizada por AWS Tools for PowerShell.

Cómo funciona SAML Support en el AWS Tools for PowerShell

En esta sección se describe cómo los AWS Tools for PowerShell cmdlets permiten a los usuarios configurar la federación de identidades SAML basada en datos.



1. AWS Tools for PowerShell se autentica en AD FS con las credenciales actuales del usuario de Windows o, de forma interactiva, cuando el usuario intenta ejecutar un cmdlet que requiere credenciales para realizar llamadas. AWS
2. AD FS autentica al usuario.
3. AD FS genera una respuesta de autenticación SAML 2.0 que incluye una afirmación; el objetivo de la afirmación es identificar al usuario y proporcionar información sobre él. AWS Tools for PowerShell extrae la lista de las funciones autorizadas del usuario de la SAML afirmación.
4. AWS Tools for PowerShell reenvía la SAML solicitud, incluidos los nombres de recursos de Amazon (ARN) del rol solicitado, STS al realizar la `AssumeRoleWithSAMLRequest` API llamada.

5. Si la SAML solicitud es válida, STS devuelve una respuesta que contiene los caracteres AWS `AccessKeyIdSecretAccessKey`, y`SessionToken`. Estas credenciales duran 3 600 segundos (1 hora).
6. El usuario ahora tiene credenciales válidas para trabajar con cualquier AWS servicio al APIs que el rol del usuario esté autorizado a acceder. AWS Tools for PowerShell aplica automáticamente estas credenciales a cualquier AWS API llamada posterior y las renueva automáticamente cuando caducan.

> **Note**
>
> Cuando las credenciales caducan y se necesitan nuevas credenciales, AWS Tools for PowerShell vuelve a autenticarse automáticamente con AD FS y obtiene credenciales nuevas durante la hora siguiente. Para los usuarios de cuentas unidas al dominio, este proceso se produce silenciosamente. En el caso de las cuentas que no están unidas a un dominio, AWS Tools for PowerShell pide a los usuarios que introduzcan sus credenciales antes de poder volver a autenticarse.

Cómo utilizar los cmdlets de configuración PowerShell SAML

AWS Tools for PowerShell incluye dos cmdlets nuevos que proporcionan soporte. SAML

- AD FS`Set-AWSSamlEndpoint` configura el punto de enlace de AD-FS, asigna un nombre descriptivo al punto de enlace y, de forma opcional, describe el tipo de autenticación del punto de enlace.
- `Set-AWSSamlRoleProfile` crea o edita el perfil de cuenta de usuario que desea asociar a un punto de enlace de AD FS, identificado mediante la especificación del nombre descriptivo proporcionado en el cmdlet `Set-AWSSamlEndpoint`. Cada perfil de rol se asigna a un único rol que el usuario tiene permiso para realizar.

  Al igual que con los perfiles de AWS credenciales, se asigna un nombre descriptivo al perfil de rol. Puede usar el mismo nombre descriptivo con el `Set-AWSCredential` cmdlet o como valor del `-ProfileName` parámetro para cualquier cmdlet que invoque el servicio. AWS APIs

Abra una sesión nueva. AWS Tools for PowerShell Si ejecuta la PowerShell versión 3.0 o una versión posterior, el AWS Tools for PowerShell módulo se importa automáticamente al ejecutar

cualquiera de sus cmdlets. Si ejecuta la PowerShell versión 2.0, debe importar el módulo manualmente ejecutando el cmdlet ``Import-Module``, como se muestra en el siguiente ejemplo.

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

Cómo ejecutar los cmdlets **Set-AWSSamlEndpoint** y **Set-AWSSamlRoleProfile**

1. En primer lugar, configure el valor del punto de enlace del sistema AD FS. La forma más sencilla de hacer esto es almacenar el punto de enlace en una variable, tal y como se muestra en este paso. Asegúrese de reemplazar la cuenta de marcador de posición y el nombre de host de AD FS por su IDs propia cuenta y el nombre de host de AD FS. IDs Especifique el nombre de host de AD FS en el parámetro `Endpoint`.

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

2. Para crear el valor del punto de enlace, ejecute el cmdlet `Set-AWSSamlEndpoint` especificando el valor correcto para el parámetro `AuthenticationType`. Los valores válidos son `Basic`, `Digest`, `Kerberos`, `Negotiate` y `NTLM`. Si no especifica este parámetro, el valor predeterminado es `Kerberos`.

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

   El cmdlet devuelve el nombre descriptivo que asignó mediante el parámetro `-StoreAs`, para que pueda utilizarlo cuando ejecute `Set-AWSSamlRoleProfile` en la línea siguiente.

3. Ahora, ejecute el `Set-AWSSamlRoleProfile` cmdlet para autenticarse con el proveedor de identidades de AD FS y obtener el conjunto de funciones (en la SAML afirmación) que el usuario está autorizado a desempeñar.

   El cmdlet `Set-AWSSamlRoleProfile` utiliza el conjunto de roles devueltos para pedir al usuario que seleccione el rol que desea asociar con el perfil especificado o para validar que los datos del rol proporcionados en los parámetros están presentes (si no están presentes, se le pedirá al usuario que los elija). Si el usuario está autorizado para un único rol, el cmdlet asocia el rol con el perfil automáticamente, sin preguntarle al usuario. No es necesario proporcionar credenciales para configurar un perfil para su uso con una cuenta unida al dominio.

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

Como alternativa, en el caso de non-domain-joined las cuentas, puedes proporcionar las credenciales de Active Directory y, a continuación, seleccionar una AWS función a la que el usuario tenga acceso, como se muestra en la siguiente línea. Esto es útil si tiene cuentas de usuario de Active Directory diferentes para diferenciar los roles dentro de su organización (por ejemplo, funciones de administración).

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

4. En cualquier caso, el cmdlet `Set-AWSSamlRoleProfile` le pide que elija el rol que debe almacenarse en el perfil. En el ejemplo siguiente se muestran dos roles disponibles: `ADFS-Dev` y `ADFS-Production`. El administrador de AD FS asocia las IAM funciones a sus credenciales de inicio de sesión de AD.

```
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"):
```

También puede especificar un rol sin la solicitud introduciendo los parámetros `RoleARN`, `PrincipalARN` y, opcionalmente, `NetworkCredential`. Si el rol especificado no aparece en la aserción devuelta por la autenticación, se le pedirá al usuario que elija entre los roles disponibles.

```
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. Puede crear perfiles para todos los roles en un solo comando añadiendo el parámetro `StoreAllRoles`, tal y como se muestra en el siguiente código. Tenga en cuenta que el nombre del rol se utiliza como nombre de perfil.

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

```
ADFS-Production
```

Cómo usar los perfiles de rol para ejecutar cmdlets que requieren credenciales AWS

Para ejecutar cmdlets que requieren AWS credenciales, puede usar los perfiles de rol definidos en el AWS archivo de credenciales compartido. Proporcione el nombre de un perfil de rol `Set-AWSCredential` (o el valor de cualquier `ProfileName` parámetro del AWS Tools for PowerShell) para obtener automáticamente AWS las credenciales temporales para el rol que se describe en el perfil.

Aunque utilice un único perfil de rol cada vez, puede cambiar de un perfil a otro dentro de una sesión del shell. El cmdlet `Set-AWSCredential` no autentica ni obtiene credenciales cuando se ejecuta por sí solo; el cmdlet registra que desea usar un perfil de rol especificado. Hasta que ejecuta un cmdlet que requiera credenciales de AWS , no se produce la autenticación ni la solicitud de credenciales.

Ahora puede usar las AWS credenciales temporales que obtuvo con el `SAMLDemoProfile` perfil para trabajar con el AWS servicioAPIs. En las secciones siguientes se muestran ejemplos de cómo utilizar los perfiles de rol.

Ejemplo 1: Definir un rol predeterminado con **`Set-AWSCredential`**

En este ejemplo se establece un rol predeterminado para una AWS Tools for PowerShell sesión mediante`Set-AWSCredential`. A continuación, puede ejecutar cmdlets que requieran credenciales y que estén autorizados por el rol especificado. En este ejemplo se muestran todas las instancias de Amazon Elastic Compute Cloud en la región EE. UU. Oeste (Oregón) que están asociadas con el perfil especificado con el cmdlet `Set-AWSCredential`.

```
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}
```

Ejemplo 2: cambiar los perfiles de los roles durante una PowerShell sesión

En este ejemplo, se enumeran todos los buckets de Amazon S3 disponibles en la AWS cuenta del rol asociado al `SAMLDemoProfile` perfil. En el ejemplo se muestra que, si bien es posible que haya utilizado otro perfil al principio de la AWS Tools for PowerShell sesión, puede cambiar los perfiles especificando un valor diferente para el `-ProfileName` parámetro con los cmdlets que lo admitan. Esta es una tarea habitual para los administradores que gestionan Amazon S3 desde la línea de PowerShell comandos.

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

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

Tenga en cuenta que el cmdlet `Get-S3Bucket` especifica el nombre del perfil que se creó al ejecutar el cmdlet `Set-AWSSamlRoleProfile`. Este comando puede ser útil si ha definido un perfil de rol anteriormente en su sesión (por ejemplo, ejecutando el cmdlet `Set-AWSCredential`) y desea utilizar un perfil de rol diferente para el cmdlet `Get-S3Bucket`. El administrador de perfiles pone credenciales temporales a disposición del cmdlet `Get-S3Bucket`.

Si bien las credenciales caducan al cabo de 1 hora (un límite obligatorioSTS), las actualiza AWS Tools for PowerShell automáticamente solicitando una nueva SAML confirmación cuando la herramienta detecta que las credenciales actuales han caducado.

Para los usuarios unidos a un dominio, este proceso se produce sin interrupción, porque durante la autenticación se usa la identidad de Windows del usuario actual. En el caso non-domain-joined de las cuentas de usuario, AWS Tools for PowerShell muestra una solicitud de PowerShell credenciales en la que se solicita la contraseña del usuario. El usuario proporciona las credenciales, que se usan para volver a autenticarle y obtener una nueva aserción.

Ejemplo 3: Obtener instancias en una región

En el siguiente ejemplo, se enumeran todas las EC2 instancias de Amazon de la región Asia Pacífico (Sídney) que están asociadas a la cuenta utilizada por el `ADFS-Production` perfil. Este comando es útil para devolver todas las EC2 instancias de Amazon de una región.

```
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
```

Lecturas adicionales

Para obtener información general sobre cómo implementar el API acceso federado, consulte Cómo implementar una solución general para el acceso federado o el CLI acceso mediante SAML la API versión 2.0.

Si tiene preguntas o comentarios de soporte, visite los foros de AWS desarrolladores sobre PowerShell creación de scripts o. NETDesarrollo.

Detección y alias de cmdlet

En esta sección se muestra cómo enumerar los servicios compatibles con las AWS Tools for PowerShell, cómo mostrar el conjunto de cmdlets proporcionados por las AWS Tools for PowerShell que respaldan dichos servicios y cómo encontrar nombres de cmdlet alternativos (también denominados alias) para obtener acceso a dichos servicios.

Detección de cmdlets

Todas las operaciones de servicio (o API) de AWS se documentan en la Guía de referencia de la API de cada servicio. Por ejemplo, consulte la Referencia de la API de IAM. En la mayoría de los

casos, existe una correspondencia uno a uno entre una API de servicio de AWS y un cmdlet de PowerShell de AWS. Para obtener el nombre de cmdlet correspondiente a la API de un servicio de AWS, ejecute el cmdlet de AWS `Get-AWSCmdletName` junto con el parámetro `-ApiOperation` y el nombre de la API del servicio de AWS. Por ejemplo, para obtener todos los nombres posibles de cmdlet basados en cualquier API `DescribeInstances` del servicio de AWS disponible, ejecute el siguiente comando:

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

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

El parámetro `-ApiOperation` es el parámetro predeterminado, por lo que puede omitir el nombre del parámetro. El siguiente ejemplo es equivalente al anterior:

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

Si conoce los nombres de la API y el servicio que desea utilizar, puede incluir el parámetro `-Service` junto con el prefijo del nombre de cmdlet o parte del nombre del servicio de AWS. Por ejemplo, el prefijo del nombre de cmdlet para Amazon EC2 es `EC2`. Para obtener el nombre de cmdlet correspondiente a la API `DescribeInstances` en el servicio de Amazon EC2, ejecute uno de los siguientes comandos. Todos ellos dan como resultado la misma salida:

```
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
```

Los valores de los parámetros en estos comandos no distinguen entre mayúsculas y minúsculas.

Si no conoce el nombre de la API del servicio de AWS o el servicio de AWS que desea usar, puede utilizar el parámetro `-ApiOperation` junto con el patrón de correspondencia y el parámetro `-MatchWithRegex`. Por ejemplo, para obtener todos los nombres de cmdlet disponibles que contienen `SecurityGroup`, ejecute el siguiente comando:

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

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

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

Si conoce el nombre del servicio de AWS que desea usar, pero no la API del servicio de AWS, agregue tanto el parámetro `-MatchWithRegex` como el parámetro `-Service` para establecer el ámbito de la búsqueda en un solo servicio. Por ejemplo, para obtener todos los nombres de cmdlet que contienen `SecurityGroup` solo en el servicio de Amazon EC2, ejecute el siguiente comando:

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

CmdletName                                      ServiceOperation
   ServiceName                  CmdletNounPrefix
----------                                      ----------------
   -----------                  ----------------
Add-EC2SecurityGroupToClientVpnTargetNetwrk
 ApplySecurityGroupsToClientVpnTargetNetwork Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroup                            DescribeSecurityGroups
   Amazon Elastic Compute Cloud EC2
```

```
Get-EC2SecurityGroupReference                 DescribeSecurityGroupReferences
   Amazon Elastic Compute Cloud EC2
Get-EC2StaleSecurityGroup                     DescribeStaleSecurityGroups
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupEgress                  AuthorizeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupIngress                 AuthorizeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
New-EC2SecurityGroup                          CreateSecurityGroup
   Amazon Elastic Compute Cloud EC2
Remove-EC2SecurityGroup                       DeleteSecurityGroup
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupEgress                 RevokeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupIngress                RevokeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleEgressDescription  UpdateSecurityGroupRuleDescriptionsEgress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleIngressDescription
 UpdateSecurityGroupRuleDescriptionsIngress  Amazon Elastic Compute Cloud EC2
```

Si conoce el nombre del comando de AWS Command Line Interface (AWS CLI), puede utilizar el parámetro `-AwsCliCommand` y el nombre de comando de la AWS CLI deseada para obtener el nombre del cmdlet basado en la misma API. Por ejemplo, para obtener el nombre de cmdlet correspondiente a la llamada al comando `authorize-security-group-ingress` de la AWS CLI en el servicio de Amazon EC2, ejecute el siguiente comando:

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

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

El cmdlet `Get-AWSCmdletName` solo necesita una parte del nombre de comando de la AWS CLI para identificar el servicio y la API de AWS.

Para obtener una lista de todos los cmdlets en Tools for PowerShell Core, ejecute el cmdlet `Get-Command` de PowerShell, como se muestra en el siguiente ejemplo.

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

Puede ejecutar el mismo comando con `-Module AWSPowerShell` para ver los cmdlets en las AWS Tools for Windows PowerShell.

El cmdlet `Get-Command` genera la lista de cmdlets en orden alfabético. Tenga en cuenta que, de forma predeterminada, la lista se ordena por verbo de PowerShell en lugar de por nombre de PowerShell.

Para ordenar los resultados por servicio en su lugar, ejecute el siguiente comando:

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

Para filtrar los cmdlets devueltos por el cmdlet `Get-Command`, pase la salida al cmdlet `Select-String` de PowerShell. Por ejemplo, para ver el conjunto de cmdlets que funcionan con regiones de AWS, ejecute el siguiente comando:

```
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
```

También puede encontrar cmdlets para un servicio específico filtrando por el prefijo de servicio de los nombres de cmdlet. Para ver la lista de prefijos de servicio disponibles, ejecute `Get-AWSPowerShellVersion -ListServiceVersionInfo`. El siguiente ejemplo devuelve cmdlets que admiten el servicio de Amazon CloudWatch Events.

```
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
```

Nomenclatura y alias de cmdlets

Los cmdlets en las AWS Tools for PowerShell para cada servicio se basan en los métodos que proporciona el AWS SDK para el servicio. No obstante, debido a las convenciones obligatorias de nomenclatura de PowerShell, el nombre de un cmdlet puede ser diferente del nombre de la llamada a la API o del método en el que se basa. Por ejemplo, el cmdlet `Get-EC2Instance` se basa en el método `DescribeInstances` de Amazon EC2.

En algunos casos, el nombre de cmdlet puede ser similar a un nombre de método, pero realizar una función diferente. Por ejemplo, el método `GetObject` de Amazon S3 recupera un objeto de Amazon S3. Sin embargo, el cmdlet `Get-S3Object` devuelve información sobre un objeto de Amazon S3 en lugar del propio objeto.

```
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
```

Para obtener un objeto de S3 con las AWS Tools for PowerShell, ejecute el cmdlet `Read-S3Object`:

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

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

> **Note**
>
> La ayuda de un cmdlet de AWS proporciona el nombre de la API de AWS SDK sobre el que se basa el cmdlet.
> Para obtener más información acerca de los verbos de PowerShell estándares y sus significados, consulte Verbos aprobados para comandos de PowerShell.

Todos los cmdlets de AWS que utilizan el verbo `Remove`, y el cmdlet `Stop-EC2Instance` cuando se agrega el parámetro `-Terminate`, solicitan confirmación antes de continuar. Para eludir la confirmación, añada el parámetro `-Force` a su comando.

> **Important**
>
> Los cmdlets de AWS no admiten el modificador `-WhatIf`.

Alias

La instalación de AWS Tools for PowerShell instala un archivo de alias que contiene alias para muchos de los cmdlets de AWS. Tal vez estos alias le resulten más intuitivos que los nombres de cmdlet. Por ejemplo, los nombres de los servicios y los nombres de los métodos de AWS SDK sustituyen a los verbos y los nombres de PowerShell en algunos alias. Un ejemplo es el alias `EC2-DescribeInstances`.

Otros alias usan verbos que, aunque no siguen las convenciones de PowerShell estándar, pueden describir mejor la operación real. Por ejemplo, el archivo de alias asocia el alias `Get-S3Content` al cmdlet `Read-S3Object`.

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

El archivo de alias se encuentra en el directorio de instalación de las AWS Tools for PowerShell. Para cargar los alias en su entorno, use la notación dot-source en el archivo. A continuación se muestra un ejemplo basado en Windows.

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

Para un shell de Linux o macOS, podría ser así:

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

Para mostrar todos los alias de las AWS Tools for PowerShell, ejecute el siguiente comando. Este comando utiliza el alias ? para el cmdlet de PowerShell `Where-Object` y la propiedad `Source` para filtrar solo los alias que provengan solo del módulo `AWSPowerShell.NetCore`.

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

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           Add-ASInstances                                    3.3.343.0
 AWSPowerShell
Alias           Add-CTTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-DPTags                                         3.3.343.0
 AWSPowerShell
Alias           Add-DSIpRoutes                                     3.3.343.0
 AWSPowerShell
Alias           Add-ELBTags                                        3.3.343.0
 AWSPowerShell
Alias           Add-EMRTag                                         3.3.343.0
 AWSPowerShell
Alias           Add-ESTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-MLTag                                          3.3.343.0
 AWSPowerShell
Alias           Clear-AWSCredentials                               3.3.343.0
 AWSPowerShell
Alias           Clear-AWSDefaults                                  3.3.343.0
 AWSPowerShell
Alias           Dismount-ASInstances                               3.3.343.0
 AWSPowerShell
Alias           Edit-EC2Hosts                                      3.3.343.0
 AWSPowerShell
```

```
Alias           Edit-RSClusterIamRoles                          3.3.343.0
 AWSPowerShell
Alias           Enable-ORGAllFeatures                           3.3.343.0
 AWSPowerShell
Alias           Find-CTEvents                                   3.3.343.0
 AWSPowerShell
Alias           Get-ASACases                                    3.3.343.0
 AWSPowerShell
Alias           Get-ASAccountLimits                             3.3.343.0
 AWSPowerShell
Alias           Get-ASACommunications                           3.3.343.0
 AWSPowerShell
Alias           Get-ASAServices                                 3.3.343.0
 AWSPowerShell
Alias           Get-ASASeverityLevels                           3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckRefreshStatuses       3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorChecks                     3.3.343.0
 AWSPowerShell
Alias           Get-ASATrustedAdvisorCheckSummaries             3.3.343.0
 AWSPowerShell
Alias           Get-ASLifecycleHooks                            3.3.343.0
 AWSPowerShell
Alias           Get-ASLifecycleHookTypes                        3.3.343.0
 AWSPowerShell
Alias           Get-AWSCredentials                              3.3.343.0
 AWSPowerShell
Alias           Get-CDApplications                              3.3.343.0
 AWSPowerShell
Alias           Get-CDDeployments                               3.3.343.0
 AWSPowerShell
Alias           Get-CFCloudFrontOriginAccessIdentities          3.3.343.0
 AWSPowerShell
Alias           Get-CFDistributions                             3.3.343.0
 AWSPowerShell
Alias           Get-CFGConfigRules                              3.3.343.0
 AWSPowerShell
Alias           Get-CFGConfigurationRecorders                   3.3.343.0
 AWSPowerShell
Alias           Get-CFGDeliveryChannels                         3.3.343.0
 AWSPowerShell
Alias           Get-CFInvalidations                             3.3.343.0
 AWSPowerShell
```

```
Alias           Get-CFNAccountLimits                                3.3.343.0
 AWSPowerShell
Alias           Get-CFNStackEvents                                  3.3.343.0
 AWSPowerShell

...
```

Para añadir su propio alias a este archivo, es posible que tenga que aumentar el valor de la variable de la preferencia `$MaximumAliasCount` de PowerShell a un valor mayor que 5500. El valor predeterminado es 4096 y puede elevarlo a un máximo de 32 768. Para ello, haga lo siguiente.

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

Para verificar que el cambio se ha realizado correctamente, escriba el nombre de la variable para que muestre su valor actual.

```
PS > $MaximumAliasCount
32768
```

Canalización y $AWSHistory

Para las llamadas de servicio de AWS que devuelven colecciones, los objetos de la colección ahora se enumeran siempre en la canalización. Los objetos resultantes que contienen campos adicionales además de la colección y que no son campos de control de paginación tienen estos campos añadidos como propiedades Note para las llamadas. Estas propiedades Note se registran en la nueva variable de sesión `$AWSHistory`, por si necesitara tener acceso a estos datos. La variable `$AWSHistory` se describe en la siguiente sección.

> **Note**
>
> En versiones de Tools for Windows PowerShell anteriores a v1.1, se emitía el propio objeto de colección, que era necesario para usar el operador foreach {$_.getenumerator()} para continuar con la canalización.

Ejemplos

En el siguiente ejemplo se devuelve una lista de regiones de AWS y las Amazon EC2 machine images (AMI) de cada región.

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

En el siguiente ejemplo se detienen todas las instancias de Amazon EC2 de la región predeterminada actual.

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

Como las colecciones se enumeran en la canalización, la salida de un determinado cmdlet puede ser `$null`, un único objeto o una colección. Si es una colección, puede utilizar la propiedad `.Count` para determinar el tamaño de la colección. Sin embargo, la propiedad `.Count` no está presente cuando se emite un único objeto. Si el script debe determinar, de forma coherente, la cantidad de objetos que se han emitido, puede comprobar la propiedad `EmittedObjectsCount` del último valor del comando en `$AWSHistory`.

$AWSHistory

Para admitir una mejor canalización, la salida de los cmdlets de AWS no se reestructura para incluir la respuesta del servicio y las instancias resultantes como propiedades Note en el objeto de colección emitido. En lugar de ello, para aquellas llamadas que emiten una única colección como salida, la colección ahora se enumera en la canalización de PowerShell. Esto significa que la repuesta del AWS SDK y los datos resultantes no pueden existir en la canalización, porque no hay ningún objeto de colección contenedor con el que se puedan asociar.

Aunque la mayoría de los usuarios probablemente no necesiten estos datos, pueden resultar útiles para fines de diagnóstico, ya que es posible ver exactamente lo que se ha enviado y recibido de las llamadas del servicio de AWS subyacente realizadas por el cmdlet.

A partir de la versión 1.1, estos datos y otros están ahora disponibles en una variable del shell llamada `$AWSHistory`. Esta variable mantiene un registro de las invocaciones de cmdlets de AWS y las respuestas del servicio que se recibieron para cada invocación. Opcionalmente, este historial también se puede configurar para registrar las solicitudes de servicio realizadas por cada cmdlet. Otros datos útiles, como el tiempo total de ejecución del cmdlet, también se puede obtener de cada entrada. Por motivos de seguridad, las solicitudes y respuestas que contienen datos confidenciales no se registran de forma predeterminada. Sin embargo, el historial se puede configurar para anular este comportamiento si es necesario. Para obtener más información, consulte el cmdlet `Set-AWSHistoryConfiguration` que se muestra a continuación.

Cada entrada de la lista `$AWSHistory.Commands` es de tipo `AWSCmdletHistory`. Este tipo tiene los siguientes miembros útiles:

CmdletName

Nombre del cmdlet.

CmdletStart

Fecha y hora en que se ejecutó el cmdlet.

CmdletEnd

Fecha y hora en que el cmdlet terminó todo el procesamiento.

Solicitudes

Si el registro de solicitudes está habilitado, muestra las últimas solicitudes del servicio.

Respuestas

Lista de las últimas respuestas del servicio recibidas.

LastServiceResponse

Método auxiliar para devolver la respuesta del servicio más reciente.

LastServiceRequest

Método auxiliar para devolver la solicitud de servicio más reciente, si está disponible.

Tenga en cuenta que la variable `$AWSHistory` no se crea hasta que se usa un cmdlet de AWS que realiza una llamada al servicio. Se evalúa como $null hasta ese momento.

> **Note**
>
> Las versiones anteriores de Tools for Windows PowerShell emitían datos relacionados con las respuestas del servicio como propiedades `Note` en el objeto devuelto. Estas ahora se encuentran en las entradas de respuesta que se registran para cada invocación de la lista.

Set-AWSHistoryConfiguration

Una invocación de cmdlet puede contener cero o más solicitudes de servicio y entradas de respuesta. Para limitar el impacto en la memoria, la lista `$AWSHistory` mantiene un registro de solo

las últimas cinco ejecuciones de cmdlets de forma predeterminada; y para cada una de ellas, las últimas cinco respuestas (y si se habilita, las últimas cinco solicitudes de servicio). Puede cambiar estos límites predeterminados ejecutando el cmdlet `Set-AWSHistoryConfiguration`. Le permite controlar el tamaño de la lista y también si se han registrado solicitudes de servicio:

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

Todos los parámetros son opcionales.

El parámetro `MaxCmdletHistory` establece el número máximo de cmdlets que puede controlar en un momento dado. Un valor de 0 desactiva el registro de la actividad de los cmdlets de AWS. El parámetro `MaxServiceCallHistory` establece el número máximo de respuestas de servicio (o solicitudes) que se controlan para cada cmdlet. El parámetro `RecordServiceRequests`, si se especifica, activa el seguimiento de solicitudes de servicio para cada cmdlet. El parámetro `IncludeSensitiveData`, si se especifica, activa el seguimiento de las respuestas y solicitudes de servicio (si se realiza un seguimiento) que contienen datos confidenciales para cada cmdlet.

Si se ejecuta sin parámetros, `Set-AWSHistoryConfiguration` simplemente desactiva el registro de todas las solicitudes anteriores, dejando el tamaño de la lista actual sin cambios.

Para borrar todas las entradas de la lista del historial actual, ejecute el cmdlet `Clear-AWSHistory`.

`$AWSHistory` Ejemplos

Enumerar los detalles de los cmdlets de AWS que se encuentran en la lista de la canalización.

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

Tener acceso a los detalles del último cmdlet de AWS que se ejecutó:

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

Tener acceso a los detalles de la última respuesta del servicio recibida por el último cmdlet de AWS que se ejecutó. Si la salida de un cmdlet de AWS está paginada, se podrían realizar varias llamadas de servicio para obtener todos los datos o la cantidad máxima de datos (determinada por los parámetros del cmdlet).

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

Obtener acceso a los detalles de la última solicitud realizada (de nuevo, un cmdlet puede realizar más de una solicitud si la salida está paginada en nombre del usuario). Produce $null a menos que el seguimiento de solicitudes de servicio esté habilitado.

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

Paginación automática hasta el final para las operaciones que devuelven varias páginas

Para las API de servicio que imponen un número máximo de objetos devueltos para una determinada llamada o que admiten conjuntos de resultados paginables, todos los cmdlets "paginan hasta el final" de forma predeterminada. Cada cmdlet realiza todas las llamadas necesarias en su nombre para devolver todo el conjunto de datos a la canalización.

En el siguiente ejemplo, que utiliza `Get-S3Object`, la variable `$c` contiene instancias `S3Object` para cada clave en el bucket `test`, lo que puede dar lugar a un gran conjunto de datos.

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

Si desea conservar el control de la cantidad de datos devueltos, puede utilizar los parámetros en los distintos cmdlets (por ejemplo `MaxKey` en `Get-S3Object`) o puede controlar de forma explícita la paginación usted mismo usando una combinación de parámetros de paginación en los cmdlets y los datos incluidos en la variable `$AWSHistory` para obtener los siguientes datos de token del servicio. El siguiente ejemplo utiliza el parámetro MaxKeys para limitar el número de instancias `S3Object` que se devuelven a las primeras 500 encontradas en el bucket como máximo.

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

Para saber si hay más datos disponibles que no se han devuelto, use la entrada de la variable de sesión `$AWSHistory` que registró las llamadas del servicio realizadas por el cmdlet.

Si la siguiente expresión se evalúa como $true, puede encontrar el marcador `next` para el siguiente conjunto de resultados utilizando `$AWSHistory.LastServiceResponse.NextMarker`:

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

Para controlar manualmente la paginación con `Get-S3Object`, utilice una combinación de los parámetros `MaxKey` y `Marker` del cmdlet y las notas `IsTruncated`/`NextMarker` sobre la última

respuesta registrada. En el siguiente ejemplo, la variable `$c` contiene hasta un máximo de 500 instancias `S3Object` para los siguientes 500 objetos que se encuentran en el bucket tras el inicio del marcador de prefijo de clave especificado.

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

Resolución de credencial y perfil

Orden de búsqueda de credenciales

Cuando ejecuta un comando, AWS Tools for PowerShell busca las credenciales en el orden siguiente. Se detiene cuando encuentra credenciales utilizables.

1. Credenciales literales que están incrustadas como parámetros en la línea de comandos.

   Es absolutamente recomendable que utilice perfiles en lugar de incluir credenciales literales en las líneas de comandos.
2. Un nombre de perfil o una ubicación de perfil especificados.
   - Si solo especifica un nombre de perfil, el comando busca el perfil especificado en el almacén de AWS SDK y, en caso de que no exista, el perfil especificado del archivo de credenciales compartidas de AWS en la ubicación predeterminada.
   - Si especifica solo una ubicación de perfil, el comando busca el perfil `default` desde ese archivo de credenciales.
   - Si especifica un nombre y una ubicación, el comando busca el perfil especificado en ese archivo de credenciales.

   Si no se encuentra el perfil o la ubicación especificados, el comando genera una excepción. La búsqueda continúa con los pasos siguientes únicamente si no ha especificado un perfil o una ubicación.
3. Credenciales especificadas por el comando `-Credential`.
4. El perfil de sesión, si existe alguno.
5. El perfil predeterminado, en el orden que se indica a continuación:
   a. El perfil `default` en el almacén de AWS SDK.
   b. El perfil `default` en el archivo de credenciales compartidas de AWS.

c. El perfil `AWS PS Default` en el almacén de AWS SDK.

6. Si el comando se ejecuta en una instancia de Amazon EC2 configurada para utilizar un rol de IAM, se accede a las credenciales temporales de la instancia EC2 desde el perfil de instancias.

   Para obtener más información acerca del uso de roles de IAM para instancias de Amazon EC2, consulte AWS SDK for .NET.

Si la búsqueda no puede encontrar las credenciales especificadas, el comando produce una excepción.

Información adicional acerca de los usuarios y los roles

Para ejecutar los comandos de Herramientas para PowerShell en AWS, debe tener una combinación de usuarios, conjuntos de permisos y roles de servicio adecuada para las tareas.

Los usuarios, conjuntos de permisos y roles de servicio específicos que cree, así como la forma en que los utilice, dependerán de sus necesidades. A continuación, se muestra información adicional sobre por qué se podrían usar y cómo crearlos.

Usuarios y conjuntos de permisos

Aunque es posible utilizar una cuenta de usuario de IAM con credenciales de larga duración para acceder a los servicios de AWS, esta práctica ya no es recomendable y se debe evitar. Incluso durante el desarrollo, se recomienda crear usuarios y conjuntos de permisos en AWS IAM Identity Center y utilizar credenciales temporales proporcionadas por un origen de identidad.

Para el desarrollo, puede usar el usuario que creó o que le dieron en Configurar la autenticación de herramientas. Si tiene los permisos de AWS Management Console adecuados, también puede crear diferentes conjuntos de permisos con los privilegios mínimos para ese usuario o crear nuevos usuarios específicamente para proyectos de desarrollo, lo que proporciona conjuntos de permisos con los privilegios mínimos. El curso de acción que elija, si corresponde, depende de las circunstancias.

Para obtener más información sobre estos usuarios y conjuntos de permisos y sobre cómo crearlos, consulte Autenticación y acceso en la Guía de referencia herramientas y AWS SDK e Introducción en la Guía del usuario de AWS IAM Identity Center.

Roles de servicio

Puede configurar un rol de servicio de AWS para acceder a los servicios de AWS en nombre de los usuarios. Este tipo de acceso es adecuado si varias personas van a ejecutar la aplicación de forma remota; por ejemplo, en una instancia de Amazon EC2 que haya creado para este fin.

El proceso de creación de un rol de servicio varía en función de la situación, pero básicamente es el siguiente.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Elija Roles y después Create Role (Crear rol).
3. Elija el servicio de AWS, busque y seleccione EC2 (por ejemplo) y, a continuación, elija el caso de uso de EC2 (por ejemplo).
4. Elija Siguiente y seleccione las políticas adecuadas para los servicios de AWS que utilizará la aplicación.

   > ⚠ Warning
   >
   > NO elija la política AdministratorAccess porque esa política permite permisos de lectura y escritura en casi todo el contenido de la cuenta.

5. Elija Next (Siguiente). Ingrese un nombre de rol, una descripción y las etiquetas que desee.

   Puede encontrar información sobre etiquetas en Controlar el acceso con etiquetas de recursos de AWS en la Guía del usuario de IAM.
6. Elija Create role (Crear rol).

Puede encontrar información de alto nivel acerca de los roles de IAM en Identidades de IAM (usuarios, grupos de usuarios y roles) en la Guía del usuario de IAM. Encuentre información detallada sobre los roles en el tema Roles de IAM.

Uso de credenciales heredadas

En los temas de esta sección se proporciona información sobre el uso de credenciales a corto o largo plazo sin usar AWS IAM Identity Center.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

> **Note**
>
> La información de estos temas se refiere a las circunstancias en las que necesita obtener y administrar credenciales a corto o largo plazo de forma manual. Para obtener información adicional sobre las credenciales a corto y largo plazo, consulte Otras formas de autenticarse en la Guía de referencia de herramientas y AWS SDK.
> Para conocer las prácticas recomendadas de seguridad, use AWS IAM Identity Center, como se describe en Configurar la autenticación de herramientas.

Advertencias y directrices importantes para las credenciales

Advertencias para las credenciales

- NO use las credenciales raíz de la cuenta para obtener acceso a los recursos de AWS. Estas credenciales proporcionan acceso ilimitado a la cuenta y son difíciles de revocar.
- NO incluya claves de acceso literales ni información sobre credenciales en los comandos o scripts. Si lo hace, corre el riesgo de exponer accidentalmente sus credenciales.
- Tenga en cuenta que las credenciales almacenadas en el archivo compartido `credentials` de AWS se almacenan en texto no cifrado.

Guía adicional para administrar las credenciales de forma segura

Para obtener información general sobre cómo administrar de forma segura las credenciales de AWS, consulte credenciales de seguridad de AWS en Referencia general de AWS y Prácticas recomendadas de seguridad y casos de uso en la Guía del usuario de IAM. Además de esas conversaciones, tenga en cuenta lo siguiente:

- Cree usuarios adicionales, como los usuarios del Centro de identidades de IAM y utilice sus credenciales en lugar de las credenciales de usuario raíz de AWS. Las credenciales de otros

usuarios se pueden revocar si es necesario o son de naturaleza temporal. Además, puede aplicar una política a cada usuario para acceder solo a determinados recursos y acciones y, por lo tanto, adoptar una política de permisos con privilegios mínimos.

- Use roles de IAM para tareas para tareas de Amazon Elastic Container Service (Amazon ECS).
- Use roles de IAM para aplicaciones que se ejecutan en instancias de Amazon EC2.

Temas

Uso de credenciales de AWS

Cada comando de las AWS Tools for PowerShell debe incluir un conjunto de credenciales de AWS, que se utilizan para firmar criptográficamente la solicitud del servicio web correspondiente. Puede especificar las credenciales por comando, por sesión o para todas las sesiones.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

> **Note**
>
> La información de este tema se refiere a las circunstancias en las que necesita obtener y administrar credenciales a corto o largo plazo de forma manual. Para obtener información adicional sobre las credenciales a corto y largo plazo, consulte Otras formas de autenticarse en la Guía de referencia de herramientas y AWS SDK.
> Para conocer las prácticas recomendadas de seguridad, use AWS IAM Identity Center, como se describe en Configurar la autenticación de herramientas.

Es recomendable que procure no exponer sus credenciales: no incluya credenciales literales en un comando. En lugar de ello, cree un perfil para cada conjunto de credenciales que desee utilizar y almacene el perfil en alguno de los dos almacenes de credenciales. Especifique el nombre de perfil correcto en el comando y las AWS Tools for PowerShell recuperarán las credenciales asociadas. Para obtener una descripción general de cómo administrar de forma segura las credenciales de AWS, consulte Prácticas recomendadas para administrar las claves de acceso de AWS en la Referencia general de Amazon Web Services.

> **Note**
>
> Necesita una cuenta de AWS para obtener credenciales y usar las AWS Tools for PowerShell. Para crear una cuenta de AWS, consulte Introducción: ¿es la primera vez que usa AWS? en la Guía de referencia de AWS Account Management.

Temas

Ubicaciones de almacén de credenciales

Las AWS Tools for PowerShell pueden utilizar uno de los dos almacenes de credenciales:

- El almacén de AWS SDK, que cifra las credenciales y las almacena en su carpeta principal. En Windows, este almacén se encuentra en: `C:\Users\username\AppData\Local\AWSToolkit\RegisteredAccounts.json`.

  AWS SDK for .NET y Toolkit for Visual Studio también pueden utilizar el almacén de AWS SDK.
- El archivo de credenciales compartidas, que también se encuentra en su carpeta principal, pero que almacena las credenciales como texto sin formato.

  De forma predeterminada, el archivo de credenciales se almacena aquí:

  - En Windows: `C:\Users\username\.aws\credentials`

- En Mac/Linux: `~/.aws/credentials`

Los AWS SDK y la AWS Command Line Interface también pueden utilizar el archivo de credenciales. Si ejecuta un script fuera del contexto de usuario de AWS, asegúrese de que el archivo que contiene sus credenciales se copia en una ubicación en la que todas las cuentas de usuario (sistema local y usuario) pueden tener acceso a sus credenciales.

Administración de perfiles

Los perfiles le permiten hacer referencia a diferentes conjuntos de credenciales con las AWS Tools for PowerShell. Puede usar cmdlets de AWS Tools for PowerShell para administrar sus perfiles en el almacén de AWS SDK. También puede administrar los perfiles en el almacén de AWS SDK mediante Toolkit for Visual Studio o mediante programación a través de AWS SDK for .NET. Para obtener instrucciones sobre cómo administrar los perfiles en el archivo de credenciales, consulte Prácticas recomendadas para administrar las claves de acceso de AWS.

Añadir un nuevo perfil

Para agregar un nuevo perfil al almacén de AWS SDK, ejecute el comando `Set-AWSCredential`. Almacena la clave de acceso y la clave secreta en el archivo de credenciales predeterminado bajo el nombre de perfil que especifique.

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

- `-AccessKey`: el ID de clave de acceso.
- `-SecretKey`: la clave secreta.
- `-StoreAs`: el nombre de perfil, que debe ser único. Para especificar el perfil predeterminado, utilice el nombre `default`.

Actualizar un perfil

El almacén de AWS SDK debe mantenerse de forma manual. Si posteriormente cambia las credenciales en el servicio (por ejemplo, mediante la consola de IAM), la ejecución de un comando con las credenciales almacenadas localmente producirá el siguiente mensaje de error:

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

Puede actualizar un perfil volviendo a ejecutar el comando `Set-AWSCredential` para el perfil y pasándole las nuevas claves de acceso y clave secreta.

Mostrar perfiles

Puede comprobar la lista actual de nombres con el siguiente comando. En este ejemplo, un usuario llamado Shirley tiene acceso a tres perfiles que están todos almacenados en el archivo de credenciales compartidas (`~/.aws/credentials`).

```
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
```

Eliminar un perfil

Para eliminar un perfil que ya no necesite, utilice el siguiente comando.

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

El parámetro `-ProfileName` especifica el perfil que desea eliminar.

El comando obsoleto Clear-AWScredential todavía está disponible por motivos de compatibilidad con versiones anteriores, pero es preferible `Remove-AWSCredentialProfile`.

Especificación de credenciales

Hay varias formas de especificar credenciales. La forma preferida es identificar un perfil en lugar de incorporar credenciales literales en la línea de comandos. AWS Tools for PowerShell localiza el perfil utilizando un orden de búsqueda que se describe en Orden de búsqueda de credenciales.

En Windows, las credenciales de AWS almacenadas en el almacén de AWS SDK se cifran con la identidad de usuario de Windows que ha iniciado sesión. No se pueden descifrar usando otra cuenta, ni usar en un dispositivo que sea diferente del dispositivo en el que se crearon originalmente. Para

realizar tareas que requieren credenciales de otro usuario, como una cuenta de usuario en la que se va a ejecutar una tarea programada, configure un perfil de credenciales, tal y como se ha descrito en la sección anterior, que pueda utilizar cuando inicie sesión en el equipo como ese usuario. Inicie sesión como el usuario que realiza tareas para completar los pasos de configuración de credenciales y crear un perfil que funcione para ese usuario. A continuación, cierre la sesión y vuelva a iniciar sesión con sus propias credenciales para configurar la tarea programada.

> **Note**
>
> Utilice el parámetro `-ProfileName` común para especificar un perfil. Este parámetro es similar al parámetro `-StoredCredentials` de versiones anteriores de AWS Tools for PowerShell. Aún se admite `-StoredCredentials` para la compatibilidad con versiones anteriores.

Perfil predeterminado (recomendado)

Todos los AWS SDK y las herramientas de administración permiten buscar las credenciales automáticamente en el equipo local si las credenciales están almacenadas en un perfil denominado `default`. Por ejemplo, si tiene un perfil denominado `default` en el equipo local, no es necesario que ejecute el cmdlet `Initialize-AWSDefaultConfiguration` ni el cmdlet `Set-AWSCredential`. Las herramientas utilizan automáticamente los datos de acceso y la clave secreta almacenados en ese perfil. Para utilizar una región de AWS que no sea su región predeterminada (los resultados de `Get-DefaultAWSRegion`), puede ejecutar `Set-DefaultAWSRegion` y especificar una región.

Si su perfil no se llama `default`, pero desea utilizarlo como perfil predeterminado para la sesión actual, ejecute `Set-AWSCredential` para definirlo como perfil predeterminado.

Aunque la ejecución de `Initialize-AWSDefaultConfiguration` le permite especificar un perfil predeterminado para cada sesión de PowerShell, el cmdlet carga credenciales de su perfil con nombre personalizado, pero sobrescribe el perfil `default` con el perfil con nombre.

Le recomendamos que no ejecute `Initialize-AWSDefaultConfiguration` a menos que esté ejecutando una sesión de PowerShell en una instancia de Amazon EC2 que no se ha lanzado con un perfil de instancias y desee configurar el perfil de credenciales de forma manual. Tenga en cuenta que el perfil de credenciales en este caso no contendría credenciales. El perfil de credenciales que resulta de la ejecución de `Initialize-AWSDefaultConfiguration` en

una instancia EC2 no almacena directamente las credenciales, sino que apunta a metadatos de instancia (que proporcionan credenciales temporales que cambian automáticamente). Sin embargo, almacena la región de la instancia. Otra situación que podría requerir la ejecución de `Initialize-AWSDefaultConfiguration` se da si desea ejecutar una llamada en una región que no sea la región en la que se está ejecutando la instancia. Al ejecutar este comando se invalida de forma permanente la región almacenada en los metadatos de la instancia.

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

> **Note**
>
> Las credenciales predeterminadas se incluyen en el almacén de AWS SDK con el nombre de perfil `default`. El comando sobrescribe cualquier perfil existente con ese nombre.

Si la instancia EC2 se lanzó con un perfil de instancia, PowerShell obtiene automáticamente la información de las credenciales y de la región de AWS a partir del perfil de instancia. No hay necesidad de ejecutar `Initialize-AWSDefaultConfiguration`. La ejecución del cmdlet `Initialize-AWSDefaultConfiguration` en una instancia EC2 lanzada con un perfil de instancia no es necesaria, porque utiliza los mismos datos de perfil de instancia que ya utiliza PowerShell de forma predeterminada.

Perfil de sesión

Utilice `Set-AWSCredential` para especificar un perfil predeterminado para una determinada sesión. Este perfil invalida cualquier perfil predeterminado durante la sesión. Se recomienda su uso si desea usar un perfil con nombre personalizado en la sesión en lugar del perfil `default` actual.

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

> **Note**
>
> En las versiones de Tools for Windows PowerShell anteriores a la 1.1, el cmdlet `Set-AWSCredential` no funcionaba de forma adecuada y sobrescribía el perfil especificado por “MyProfileName”. Le recomendamos que utilice una versión más reciente de Tools for Windows PowerShell.

Perfil de comando

En comandos individuales, puede agregar el parámetro `-ProfileName` para especificar un perfil que se aplique solo a ese comando. Este perfil invalida todos los perfiles predeterminados o de sesión, como se muestra en el ejemplo.

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

> **Note**
>
> Cuando especifica un perfil de sesión o predeterminado, también puede añadir un parámetro `-Region` para invalidar una región de sesión o predeterminada. Para obtener más información, consulte Especificar AWS regiones. En el siguiente ejemplo se especifica un perfil y una región predeterminados.
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-
> west-2
> ```

De forma predeterminada, se presupone que el archivo de credenciales compartidas de AWS está en la carpeta principal del usuario (`C:\Users\username\.aws` en Windows o `~/.aws` en Linux). Para especificar un archivo de credenciales en otra ubicación, incluya el parámetro `-ProfileLocation` y especifique la ruta del archivo de credenciales. En el siguiente ejemplo se especifica un archivo de credenciales no predeterminado para un comando específico.

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

> **Note**
>
> Si ejecuta un script de PowerShell en un momento en el que no suele iniciar sesión en AWS (por ejemplo, ejecuta un script de PowerShell como una tarea programada fuera del horario laboral normal), agregue el parámetro `-ProfileLocation` cuando especifique el perfil que desea utilizar y establezca el valor en la ruta del archivo que contiene sus credenciales. Para cerciorarse de que su script de AWS Tools for PowerShell se ejecuta con las credenciales de la cuenta correctas, debe añadir el parámetro `-ProfileLocation` siempre que el script se ejecute en un contexto o proceso que no utilice una cuenta de AWS. También puede copiar

su archivo de credenciales en una ubicación que esté accesible al sistema local o en otra cuenta que usen sus scripts para realizar tareas.

Orden de búsqueda de credenciales

Cuando ejecuta un comando, AWS Tools for PowerShell busca las credenciales en el orden siguiente. Se detiene cuando encuentra credenciales utilizables.

1. Credenciales literales que están incrustadas como parámetros en la línea de comandos.

   Es absolutamente recomendable que utilice perfiles en lugar de incluir credenciales literales en las líneas de comandos.
2. Un nombre de perfil o una ubicación de perfil especificados.
   - Si solo especifica un nombre de perfil, el comando busca el perfil especificado en el almacén de AWS SDK y, en caso de que no exista, el perfil especificado del archivo de credenciales compartidas de AWS en la ubicación predeterminada.
   - Si especifica solo una ubicación de perfil, el comando busca el perfil `default` desde ese archivo de credenciales.
   - Si especifica un nombre y una ubicación, el comando busca el perfil especificado en ese archivo de credenciales.

   Si no se encuentra el perfil o la ubicación especificados, el comando genera una excepción. La búsqueda continúa con los pasos siguientes únicamente si no ha especificado un perfil o una ubicación.
3. Credenciales especificadas por el comando `-Credential`.
4. El perfil de sesión, si existe alguno.
5. El perfil predeterminado, en el orden que se indica a continuación:
   a. El perfil `default` en el almacén de AWS SDK.
   b. El perfil `default` en el archivo de credenciales compartidas de AWS.
   c. El perfil `AWS PS Default` en el almacén de AWS SDK.
6. Si el comando se ejecuta en una instancia de Amazon EC2 configurada para utilizar un rol de IAM, se accede a las credenciales temporales de la instancia EC2 desde el perfil de instancias.

   Para obtener más información acerca del uso de roles de IAM para instancias de Amazon EC2, consulte AWS SDK for .NET.

Si la búsqueda no puede encontrar las credenciales especificadas, el comando produce una excepción.

Gestión de credenciales en AWS Tools for PowerShell Core

Los cmdlets de AWS Tools for PowerShell Core aceptan claves de acceso y secretas de AWS o los nombres de perfiles de credenciales cuando se ejecutan, al igual que AWS Tools for Windows PowerShell. Cuando se ejecutan en Windows, ambos módulos tienen acceso al archivo del almacén de credenciales de AWS SDK for .NET (almacenado en el archivo `AppData\Local\AWSToolkit\RegisteredAccounts.json` de cada usuario).

Este archivo almacena las claves en formato cifrado y no se puede usar en otro equipo. Es el primer archivo en el que AWS Tools for PowerShell busca un perfil de credenciales y también es el archivo en el que AWS Tools for PowerShell almacena los perfiles de credenciales. Para obtener más información acerca del archivo de almacén de credenciales de AWS SDK for .NET, consulte Configuración de credenciales de AWS. En la actualidad, el módulo de Tools for Windows PowerShell no permite que se escriban credenciales en otros archivos o ubicaciones.

Ambos módulos pueden leer perfiles del archivo de credenciales compartidas de AWS usado por otros AWS SDK y por la AWS CLI. En Windows, la ubicación predeterminada de este archivo es `C:\Users\<userid>\.aws\credentials`. En plataformas distintas de Windows, este archivo se almacena en `~/.aws/credentials`. Se puede usar el parámetro `-ProfileLocation` para apuntar a un nombre de archivo o ubicación de archivo distintos de los predeterminados.

El almacén de credenciales del SDK almacena las credenciales en formato cifrado usando las API criptográficas de Windows. Estas API no están disponibles en otras plataformas, por lo que el módulo AWS Tools for PowerShell Core usa exclusivamente el archivo de credenciales compartidas de AWS y permite escribir nuevos perfiles de credenciales en este archivo.

Los siguientes script de ejemplo que utilizan el cmdlet `Set-AWSCredential` muestran las opciones para gestionar los perfiles de credenciales en Windows con los módulos AWSPowerShell o AWSPowerShell.NetCore.

```
# 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
```

Los siguientes ejemplos muestran el comportamiento del módulo AWSPowerShell.NetCore en los sistemas operativos Linux o macOS.

```
# 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
```

Credenciales compartidas en las AWS Tools for PowerShell

Las Tools for Windows PowerShell son compatibles con el uso del archivo de credenciales compartidas de AWS de forma similar a la AWS CLI y otros SDK de AWS. Las Tools for Windows PowerShell ahora permiten la lectura y la escritura de los perfiles de credenciales `basic`, `session` y `assume role` en el archivo de credenciales de .NET y el archivo de credenciales compartidas de AWS. Esta funcionalidad se permite gracias a un nuevo espacio de nombres `Amazon.Runtime.CredentialManagement`.

> **Warning**
>
> Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como AWS IAM Identity Center.

> **Note**
>
> La información de este tema se refiere a las circunstancias en las que necesita obtener y administrar credenciales a corto o largo plazo de forma manual. Para obtener información adicional sobre las credenciales a corto y largo plazo, consulte Otras formas de autenticarse en la Guía de referencia de herramientas y AWS SDK.
> Para conocer las prácticas recomendadas de seguridad, use AWS IAM Identity Center, como se describe en Configurar la autenticación de herramientas.

Los nuevos tipos de perfiles y el acceso al archivo de credenciales compartidas de AWS se admiten a través de los siguientes parámetros, que se han añadido a los cmdlets relacionados con credenciales, Initialize-AWSDefaultConfiguration, New-AWSCredential y Set-AWSCredential. En los cmdlets del servicio, puede hacer referencia a sus perfiles añadiendo el parámetro común `-ProfileName`.

Uso de un rol de IAM con las AWS Tools for PowerShell

El archivo de credenciales compartidas de AWS facilita tipos adicionales de acceso. Por ejemplo, puede acceder a los recursos de AWS mediante un rol de IAM en lugar de las credenciales a largo plazo de un usuario de IAM. Para ello, debe tener un perfil estándar que tenga permisos para asumir el rol. Cuando se indica a las AWS Tools for PowerShell que usen un perfil que especificó un rol, las AWS Tools for PowerShell buscan el perfil identificado por el parámetro `SourceProfile`. Estas credenciales se utilizan para solicitar credenciales temporales para el rol especificado por el parámetro `RoleArn`. Opcionalmente, puede requerir el uso de un dispositivo de autenticación multifactor (MFA) o un código `ExternalId` cuando un tercero asuma el rol.

Nombre del parámetro	Descripción
ExternalId	El ID externo definido por el usuario que se utilizará al asumir un rol, si así lo requiere

Nombre del parámetro	Descripción
	el rol. Normalmente, esto solo es necesario cuando delega el acceso a su cuenta a un tercero. El tercero debe incluir el ID externo como parámetro al asumir el rol asignado. Para obtener más información, consulte Cómo utilizar un ID externo cuando se otorga acceso a los recursos de AWS a terceros en la Guía del usuario de IAM.
MfaSerial	El número de serie de MFA que se utilizará al asumir un rol, si así lo requiere el rol. Para obtener más información, consulte Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.
RoleArn	El ARN del rol que se va a asumir al asumir las credenciales del rol. Para obtener más información acerca de la creación y el uso de roles, consulte Roles de IAM en la Guía del usuario de IAM.
SourceProfile	El nombre del perfil de origen que se va a usar al asumir las credenciales del rol. Las credenciales encontradas en este perfil se utilizan para asumir el rol especificado por el parámetro `RoleArn`.

Configuración de perfiles para asumir un rol

El siguiente es un ejemplo que muestra cómo configurar un perfil de origen que permite asumir directamente un rol de IAM.

El primer comando crea un perfil de origen al que hace referencia el perfil de rol. El segundo comando crea el perfil de rol que el rol debe asumir. El tercer comando muestra las credenciales del perfil de rol.

```
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
```

Para utilizar este perfil de rol con los cmdlets del servicio de Tools for Windows PowerShell, agregue el parámetro común `-ProfileName` al comando para hacer referencia al perfil de rol. En el ejemplo siguiente se utiliza el perfil de rol definido en el ejemplo anterior para tener acceso al cmdlet Get-S3Bucket. AWS Tools for PowerShell busca las credenciales en `my_source_profile`, usa esas credenciales para llamar a `AssumeRole` en nombre del usuario y, a continuación, utiliza esas credenciales de rol temporales para llamar a `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
```

Uso de tipos de perfiles de credenciales

Para establecer un tipo de perfil de credenciales, debe conocer qué parámetros proporcionan la información requerida por el tipo de perfil.

Tipo de credenciales	Parámetros que debe utilizar

Básica
 	
Estas son las credenciales a largo plazo para un usuario de IAM.
	`-AccessKey`
	
	`-SecretKey`

Tipo de credenciales	Parámetros que debe utilizar

Sesión:
 	
Estas son las credenciales a corto plazo para un rol de IAM que se recupera de forma manual, por ejemplo, llamando directamente al cmdlet Use-STSRole.
	`-AccessKey`
	
	`-SecretKey`
	
	`-SessionToken`

Rol:
 	
Estas son credenciales a corto plazo para un rol de IAM que las AWS Tools for PowerShell recuperan para usted.
	`-SourceProfile`
	
	`-RoleArn`
	
	opcional: `-ExternalId`
	
	opcional: `-MfaSerial`

El parámetro común **`ProfilesLocation`**

Puede utilizar `-ProfileLocation` para escribir en el archivo de credenciales compartidas, así como para indicar a un cmdlet que lea el archivo de credenciales. La incorporación del parámetro `-ProfileLocation` permite controlar si Tools for Windows PowerShell utiliza el archivo de credenciales compartidas o el archivo de credenciales de .NET. En la siguiente tabla, se describe el funcionamiento del parámetro en Tools for Windows PowerShell.

Valor de ubicación del perfil	Comportamiento de resolución del perfil
null (no establecido) o vacío	En primer lugar, busca en el archivo de credenciales de .NET un perfil con el nombre especificado. Si no se encuentra el perfil, busque en el archivo de credenciales compartid as de AWS en *(user's home directory)* `\.aws\credentials` .
La ruta a un archivo en el formato del archivo de credenciales compartidas de AWS	Busca solo en el archivo especificado el perfil con el nombre designado.

Guardar las credenciales en un archivo de credenciales

Para escribir y guardar credenciales en uno de los dos archivos de credenciales, ejecute el cmdlet `Set-AWSCredential`. El siguiente ejemplo le muestra cómo hacerlo. El primer comando utiliza `Set-AWSCredential` con `-ProfileLocation` para agregar claves de acceso y secretas a un perfil especificado por el parámetro `-ProfileName`. En la segunda línea, ejecute el cmdlet Get-Content para mostrar el contenido del archivo de credenciales.

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

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

Visualización de los perfiles de credenciales

Ejecute el cmdlet Get-AWSCredential y añada el parámetro `-ListProfileDetail` para devolver los tipos de archivos de credenciales y las ubicaciones, y una lista de nombres de perfil.

```
PS > Get-AWSCredential -ListProfileDetail

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

Eliminar perfiles de credenciales

Para eliminar perfiles de credenciales, ejecute el nuevo cmdlet Remove-AWSCredentialProfile. Clear-awsCredential está obsoleto, pero aún está disponible por motivos de compatibilidad con versiones anteriores.

Notas importantes

Solo Initialize-AWSDefaultConfiguration, New-AWSCredential y Set-AWSCredential admiten los parámetros para perfiles de rol. No se pueden especificar los parámetros de rol directamente en un comando como `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`. Eso no funciona porque los cmdlets del

servicio no admiten directamente los parámetros `SourceProfile` o `RoleArn`. En su lugar, debe almacenar esos parámetros en un perfil y, a continuación, llamar al comando con el parámetro `-ProfileName`.

Características del AWS Tools for Windows PowerShell

En algunos temas de esta sección se proporciona información sobre las características de las Herramientas para Windows PowerShell que quizás deba tener en cuenta al crear sus proyectos y scripts. En otros temas de esta sección se proporciona información sobre las formas avanzadas de configurar las herramientas, el entorno y los proyectos. Asegúrese de haber instalado y configurado primero las herramientas.

Para obtener información sobre el desarrollo de software para AWS servicios específicos junto con ejemplos de código, consulteTrabaje con AWS los servicios. Para ver otros ejemplos de código, consulte Herramientas para ejemplos PowerShell de código.

Temas

Observabilidad

> Esta es una documentación preliminar para un servicio en versión preliminar. Está sujeta a cambios.

La observabilidad es la medida en que se puede deducir el estado actual de un sistema a partir de los datos que emite. Los datos emitidos se denominan comúnmente telemetría. Para obtener información adicional sobre la telemetría al utilizar AWS los servicios, consulte Observabilidad en la Guía para desarrolladores.AWS SDK for .NET

El siguiente código muestra un ejemplo de cómo se puede habilitar la observabilidad en. AWS Tools for PowerShell

```
<#
    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 --prerelease

$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'
```

Trabaje con AWS los servicios del AWS Tools for PowerShell

En esta sección se proporcionan ejemplos del uso de los AWS Tools for PowerShell para acceder a AWS los servicios. Estos ejemplos ayudan a demostrar cómo usar los cmdlets para realizar tareas reales AWS . Estos ejemplos se basan en los cmdlets que proporciona Tools for. PowerShell Para ver qué cmdlets están disponibles, consulte la Referencia de cmdlets de AWS Tools for PowerShell.

PowerShell Codificación por concatenación de archivos

Algunos cmdlets del archivo AWS Tools for PowerShell editan los archivos o registros existentes en los que se encuentra. AWS Un ejemplo es `Edit-R53ResourceRecordSet` que llama a ChangeResourceRecordSetsAPIAmazon Route 53.

Al editar o concatenar archivos en versiones PowerShell 5.1 o anteriores, PowerShell codifica la salida en UTF -16, no en -8. UTF Esto puede añadir caracteres no deseados y crear resultados no válidos. Un editor hexadecimal pueden mostrar los caracteres no deseados.

Para evitar convertir la salida del archivo a UTF -16, puede canalizar el comando al `Out-File` cmdlet y especificar PowerShell la codificación UTF -8, como se muestra en el siguiente ejemplo:

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

Si ejecuta AWS CLI comandos desde la PowerShell consola, se aplica el mismo comportamiento. Puede canalizar el resultado de un AWS CLI comando a `Out-File` la PowerShell consola. Otros cmdlets, como `Export-Csv` o `Export-Clixml`, también tienen un parámetro `Encoding`. Para obtener una lista completa de cmdlets que tienen un parámetro `Encoding` y que permiten corregir la codificación de la salida de un archivo concatenado, ejecute el siguiente comando:

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

> **Note**
>
> PowerShell La versión 6.0 y versiones posteriores, incluida PowerShell Core, conservan automáticamente la codificación UTF -8 para la salida de archivos concatenados.

Objetos devueltos para las herramientas PowerShell

Para que sea AWS Tools for PowerShell más útil en un PowerShell entorno nativo, el objeto devuelto por un AWS Tools for PowerShell cmdlet es un. NETobjeto, no el objeto de JSON texto que normalmente se devuelve desde el correspondiente API en. AWS SDK Por ejemplo, `Get-S3Bucket` emite una `Buckets` colección, no un objeto de JSON respuesta de Amazon S3. La `Buckets` colección se puede colocar en la PowerShell canalización y se puede interactuar con ella de forma adecuada. Del mismo modo, `Get-EC2Instance` emite un. `Reservation` NETcolección de objetos, no un objeto de `DescribeEC2Instances` JSON resultado. Este comportamiento se debe a un diseño y permite que la AWS Tools for PowerShell experiencia sea más coherente con la idiomática PowerShell.

Las respuestas de servicio reales están disponibles para usted si las necesita. Se almacenan como propiedades `note` en los objetos devueltos. En el API caso de las acciones que permiten la paginación mediante `NextToken` campos, también se adjuntan como propiedades. `note`

Amazon EC2

En esta sección, se explican los pasos necesarios para lanzar una EC2 instancia de Amazon, y se incluye cómo:

- Recupera una lista de Amazon Machine Images (AMIs).
- Cree un key pair para la SSH autenticación.
- Cree y configure un grupo de EC2 seguridad de Amazon.
- Lanzar la instancia y recuperar información sobre ella

Amazon S3

En esta sección se describen los pasos necesarios para crear un sitio web estático alojado en Amazon S3. Muestra cómo:

- crear y eliminar buckets de Amazon S3
- cargar archivos en un bucket de Amazon S3 como objetos
- eliminar objetos de un bucket de Amazon S3
- designar un bucket de Amazon S3 como un sitio web

AWS Lambda y AWS Tools for PowerShell

En esta sección se proporciona una breve descripción general del PowerShell módulo AWS Lambda Tools for y se describen los pasos necesarios para configurar el módulo.

Amazon SNS y Amazon SQS

En esta sección se explican los pasos necesarios para suscribir una SQS cola de Amazon a un SNS tema de Amazon. Muestra cómo:

- Crea un SNS tema de Amazon.
- Crea una SQS cola de Amazon.
- Suscriba la cola al tema de .
- Enviar un mensaje al tema
- Recibir el mensaje de la cola

CloudWatch

En esta sección se proporciona un ejemplo de cómo publicar datos personalizados en. CloudWatch

- Publique una métrica personalizada en su CloudWatch panel de control.

Véase también

- Comenzar a utilizar la AWS Tools for Windows PowerShell

Temas

- Amazon S3 y Tools for Windows PowerShell
- Amazon EC2 y Tools for Windows PowerShell
- AWS Lambda y AWS Tools for PowerShell
- Amazon SQS, Amazon SNS y Tools for Windows PowerShell
- CloudWatch desde AWS Tools for Windows PowerShell

- Uso del parámetro ClientConfig en los cmdlets

Amazon S3 y Tools for Windows PowerShell

En esta sección, crearemos un sitio web estático mediante el uso de AWS Tools for Windows PowerShell con Amazon S3 y CloudFront. En el proceso, mostraremos una serie de tareas comunes con estos servicios. Esta explicación sigue la Guía de introducción para Alojar un sitio web estático, que describe un proceso similar con el uso de la consola de administración de AWS.

Los comandos que se muestran aquí presuponen que ya ha definido credenciales predeterminadas y una región predeterminada para su sesión de PowerShell. Por lo tanto, las credenciales y las regiones no se incluyen en la invocación de los cmdlets.

> **Note**
>
> En la actualidad, no existe ninguna API de Amazon S3 para cambiar el nombre de los buckets o los objetos y, por tanto, ningún cmdlet de Tools for Windows PowerShell para realizar esta tarea. Para cambiar el nombre de los objetos de S3, le recomendamos que copie el objeto con un nuevo nombre ejecutando el cmdlet Copy-S3Object y que, a continuación, elimine el objeto original ejecutando el cmdlet Remove-S3Object.

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- Alojamiento de un sitio web estático en Amazon S3
- Consola de Amazon S3

Temas

- Creación de un bucket de Amazon S3, verificación de su región y eliminación de este (opcional)
- Configuración de un bucket de Amazon S3 como un sitio web y habilitación del registro
- Carga de objetos en un bucket de Amazon S3
- Eliminación de objetos y buckets de Amazon S3
- Carga de contenido de texto insertado en Amazon S3

Creación de un bucket de Amazon S3, verificación de su región y eliminación de este (opcional)

Utilice el cmdlet `New-S3Bucket` para crear un nuevo bucket de Amazon S3. En los ejemplos siguientes se crea un bucket denominado `website-example`. El nombre del bucket debe ser único en todas las regiones. En el ejemplo el bucket se crea en la región `us-west-1`.

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

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

Puede verificar la región en la que se encuentra el bucket con el cmdlet `Get-S3BucketLocation`.

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

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

Cuando haya terminado este tutorial, puede utilizar la siguiente línea para eliminar este bucket. Le recomendamos que mantenga este bucket ya que lo usaremos en los siguientes ejemplos.

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

Tenga en cuenta que el proceso de eliminación del bucket puede tardar algún tiempo en completarse. Si intenta volver a crear un bucket inmediatamente con el mismo nombre, el cmdlet `New-S3Bucket` puede producir un error hasta que el bucket anterior haya desaparecido por completo.

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- Put Bucket (Referencia del servicio de Amazon S3)
- AWSRegiones de PowerShell para Amazon S3

Configuración de un bucket de Amazon S3 como un sitio web y habilitación del registro

Utilice el cmdlet `Write-S3BucketWebsite` para configurar un bucket de Amazon S3 como un sitio web estático. El siguiente ejemplo especifica un nombre de `index.html` para la página web de contenido predeterminada y un nombre de `error.html` para la página web de error predeterminada. Tenga en cuenta que este cmdlet no crea esas páginas. Deben ser cargados como objetos de Amazon S3.

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

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- Put Bucket Website (Referencia de la API de Amazon S)
- Put Bucket ACL (Referencia de la API de Amazon S)

Carga de objetos en un bucket de Amazon S3

Utilice el cmdlet `Write-S3Object` para cargar archivos del sistema de archivos local en un bucket de Amazon S3 como objetos. En el siguiente ejemplo se crean y cargan dos archivos HTML sencillos en un bucket de Amazon S3 y se verifica la existencia de los objetos cargados. El parámetro `-File` de `Write-S3Object` especifica el nombre del archivo en el sistema de archivos local. El parámetro `-Key` especifica el nombre que el objeto correspondiente tendrá en Amazon S3.

Amazon determina el tipo de contenido de los objetos de las extensiones de archivo (en este caso, ".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
```

Opciones de ACL empaquetadas

Los valores para especificar ACL predefinidas con Tools for Windows PowerShell son los mismos que aquellos que utiliza AWS SDK for .NET. Tenga en cuenta, sin embargo, que son diferentes de los valores utilizados por la acción `Put Object` de Amazon S3. Tools for Windows PowerShell admite las siguientes ACL predefinidas:

- NoACL

- private
- public-read
- public-read-write
- aws-exec-read
- authenticated-read
- bucket-owner-read
- bucket-owner-full-control
- log-delivery-write

Para obtener más información acerca de estos ajustes de listas de control de acceso empaquetadas, consulte Información general de las Access Control Lists (ACL, Listas de control de acceso).

Nota relativa a la carga multiparte

Si utiliza la API de Amazon S3 para cargar un archivo que sobrepasa los 5 GB de tamaño, debe utilizar la carga multiparte. Sin embargo, el cmdlet `Write-S3Object` proporcionado por Tools for Windows PowerShell puede encargarse de las cargas de archivos con un tamaño superior a 5 GB de manera transparente.

Probar el sitio web

En este punto, puede probar el sitio web visitándolo desde un navegador. Las URL de los sitios web estáticos alojados en Amazon S3 siguen un formato estándar.

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

Por ejemplo:

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

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- Put Object (Referencia de la API de Amazon S)
- Canned ACLs (Referencia de la API de Amazon S)

Eliminación de objetos y buckets de Amazon S3

En esta sección se describe cómo eliminar el sitio web que creó en las secciones anteriores. Puede simplemente eliminar los objetos de los archivos HTML y, a continuación, eliminar el bucket de Amazon S3 del sitio.

En primer lugar, ejecute el cmdlet `Remove-S3Object` para eliminar los objetos de los archivos HTML del bucket de Amazon S3.

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

La respuesta `False` es un artefacto esperado de la forma en la que Amazon S3 procesa la solicitud. En este contexto, no indica un problema.

Ahora, ejecute el cmdlet `Remove-S3Bucket` para eliminar el bucket de Amazon S3 que ahora está vacío del sitio.

```
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    :
```

En 1.1 y versiones más recientes de las AWS Tools for PowerShell, puede añadir el parámetro `-DeleteBucketContent` a `Remove-S3Bucket`, que primero elimina todos los objetos y versiones de objeto del bucket especificado antes de intentar eliminar el propio bucket. En función del número de objetos o versiones de objeto del bucket, esta operación puede tardar una cantidad de tiempo considerable. En las versiones de Tools for Windows PowerShell anteriores a la 1.1, el bucket tenía que estar vacío para que `Remove-S3Bucket` pudiera eliminarlo.

> **Note**
>
> A menos que agregue el parámetro `-Force`, las AWS Tools for PowerShell solicitan la confirmación antes de que se ejecute el cmdlet.

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- Delete Object (Referencia de la API de Amazon S)
- DeleteBucket (Referencia de la API de Amazon S)

Carga de contenido de texto insertado en Amazon S3

El cmdlet `Write-S3Object` permite cargar contenido de texto insertado en Amazon S3. Con el parámetro `-Content` (alias `-Text`), puede especificar el contenido de texto que debe cargarse en Amazon S3 sin necesidad de incluirlo primero en un archivo. El parámetro acepta cadenas sencillas de una sola línea, así como cadenas de varias líneas.

```
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
>> "@
>>
PS > # Specifying content from a variable: (note final newline needed to end in-line
 here-string)
PS > $x = @"
>> line 1
>> line 2
>> line 3
>> "@
>>
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content $x
```

Amazon EC2 y Tools for Windows PowerShell

Puede realizar tareas comunes relacionadas con Amazon EC2 mediante AWS Tools for PowerShell.

Los comandos de ejemplo que se muestran aquí presuponen que ya ha definido credenciales predeterminadas y una región predeterminada para su sesión de PowerShell. Por lo tanto, no se incluyen las credenciales ni la región al invocar los cmdlets. Para obtener más información, consulte Comenzar a utilizar la AWS Tools for Windows PowerShell.

Temas

Creación de un par de claves

En el ejemplo de `New-EC2KeyPair` siguiente se crea un par de claves y se almacena en la variable `$myPSKeyPair` de PowerShell

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

Pase el objeto de par de claves al cmdlet `Get-Member` para ver la estructura del objeto.

```
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;}
```

Pase el objeto de par de claves al cmdlet `Format-List` para ver los valores de los miembros `KeyName`, `KeyFingerprint` y `KeyMaterial`. (El resultado se ha truncado para facilitar su lectura).

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

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
  KeyMaterial    : ----BEGIN RSA PRIVATE KEY----
                   MIIEogIBAAKCAQEAkK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                   Mz6btoxPcE7EMeH1wySUp8nouAS9xbl9l7+VkD74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5il/MpkrRogHqOB0rigeTeV5Yc3lvO0RFFPu0Kz4kcm...
                   w3Jg8dKsWn0plOpX7V3sRC02KgJIbejQUvBFGi5OQK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvgWkcFQkLmRHRoDpPb+OdFsZtjHZDpMVFmA9tT8EdbkEF...
                   3SrNeqZPsxJJIxOodb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                   GGlLfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZWa7KFNRZuCuX7jssC...
                   xO98abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwfLtwhmJEy...
                   1BX9X8WFX/A8VLHrT1elrKmLkNECgYEAwltkV1pOJAFhz9p7ZFEv...
                   vvVsPaF0Ev9bk9pqhx269PB5Ox2KokwCagDMMaYvasWobuLmNu/1...
                   lmwRx7KTeQ7W1J3OLgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                   63g6N6rk2FkHZX1E62BgbewUd3eZOS05Ip4VUdvtGcuc8/qa+e5C...
                   KXgyt9nl64pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                   gYALEI7m1jJKpHWAEs0hiemw5VmKyIZpzGstSJsFStERlAjiETDH...
                   YAtnI4J8dRyP9I7BOVOn3wNfIjk85gi1/0Oc+j8S65giLAfndWGR...
                   9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvrUkepIbUDTD...
                   AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hkOSmWhlZGWikJ5VizBf...
                   drkBr/vTKVRMTi3lVFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                   TTld5N6opgOXAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                   x3O2duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzcOajwieNrblr7c...
                   -----END RSA PRIVATE KEY-----
```

El miembro `KeyMaterial` almacena la clave privada del par de claves. La clave pública se almacena en AWS. No puede recuperar la clave pública de AWS, pero puede verificarla comparando la `KeyFingerprint` de la clave privada con la huella devuelta por AWS para la clave pública.

Ver la huella del par de claves

Puede utilizar el cmdlet `Get-EC2KeyPair` para ver la huella del par de claves.

```
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
```

Almacenar la clave privada

Para almacenar la clave privada en un archivo, pase el miembro `KeyFingerMaterial` al cmdlet `Out-File`.

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

Debe especificar `-Encoding ascii` cuando escriba la clave privada en un archivo. De lo contrario, herramientas tales como `openssl` no podrán leer el archivo correctamente. Puede verificar que el formato del archivo resultante es correcto mediante un comando como el siguiente:

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

(La herramienta `openssl` no se incluye con AWS Tools for PowerShell ni con AWS SDK for .NET).

Eliminar el par de claves

Necesita el par de claves para lanzar una instancia y conectarse a ella. Cuando haya terminado de usar un par de claves, puede eliminarlo. Para eliminar la clave pública de AWS, utilice el cmdlet `Remove-EC2KeyPair`. Cuando se le solicite, pulse `Enter` para eliminar el par de claves.

```
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"):
```

La variable `$myPSKeyPair` sigue existiendo en la sesión de PowerShell actual y todavía contiene la información del par de claves. El archivo `myPSKeyPair.pem` también existe. Sin embargo, la clave privada ya no es válida porque la clave pública del par de claves ya no se almacena en AWS.

Creación de un grupo de seguridad mediante Windows PowerShell

Puede utilizar el AWS Tools for PowerShell para crear y configurar un grupo de seguridad. La respuesta es el ID del grupo de seguridad.

Si necesita conectarse a la instancia, debe configurar el grupo de seguridad para permitir el SSH tráfico (Linux) o el RDP tráfico (Windows).

Temas

Requisitos previos

Necesita la dirección IP pública de su equipo, en CIDR notación. Puede obtener la dirección IP pública de su equipo local usando un servicio. Por ejemplo, Amazon proporciona el siguiente servicio: http://checkip.amazonaws.com/ o https://checkip.amazonaws.com. Para buscar otro servicio que le brinde su dirección IP, utilice la frase de búsqueda "what is my IP address" (cuál es mi dirección IP). Si se conecta a través de un firewall ISP o desde detrás de él sin una dirección IP estática, necesitará encontrar el rango de direcciones IP que pueden utilizar los ordenadores cliente.

> **Warning**
>
> Si especifica `0.0.0.0/0`, habilitará el tráfico desde cualquier dirección IP del mundo. En el caso de RDP los protocolos SSH y, podría considerarlo aceptable durante un breve período de tiempo en un entorno de prueba, pero no es seguro para los entornos de producción. En producción, asegúrese de autorizar el acceso solo desde la dirección IP individual o el rango de direcciones adecuadas.

Crear un grupo de seguridad para EC2 - VPC

> **Warning**
>
> EC2-Classic se retiró el 15 de agosto de 2022. Le recomendamos que migre de EC2 - Classic a un. VPC Para obtener más información, consulte la entrada del blog EC2-Classic Networking is Retiring: aquí le mostramos cómo prepararse.

El siguiente `New-EC2SecurityGroup` ejemplo agrega el `-VpcId` parámetro para crear un grupo de seguridad para el especificado. VPC

```
PS > $groupid = New-EC2SecurityGroup `
    -VpcId "vpc-da0013b3" `
    -GroupName "myPSSecurityGroup" `
```

```
    -GroupDescription "EC2-VPC from PowerShell"
```

Para ver la configuración inicial del grupo de seguridad, use el cmdlet `Get-EC2SecurityGroup`. De forma predeterminada, el grupo de seguridad de a VPC contiene una regla que permite todo el tráfico saliente. Tenga en cuenta que no puede hacer referencia a un grupo de seguridad paraEC2... VPC por su nombre.

```
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                : {}
```

Para definir los permisos del tráfico entrante en los TCP puertos 22 (SSH) y TCP 3389, utilice el `New-Object` cmdlet. El siguiente script de ejemplo define los permisos para los TCP puertos 22 y 3389 desde una única dirección IP,. `203.0.113.25/32`

```
$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 )
```

Para verificar el grupo de seguridad que se ha actualizado, use de nuevo el cmdlet `Get-EC2SecurityGroup`.

```
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                : {}
```

Para ver las reglas de entrada, puede recuperar la propiedad `IpPermissions` del objeto de colección devuelto por el comando anterior.

```
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}
```

Buscar una imagen de máquina de Amazon mediante Windows PowerShell

Cuando se lanza una instancia de Amazon EC2, debe especificar una Amazon Machine Image(AMI) que sirva como plantilla para la instancia. Sin embargo, los ID de las AMI de Windows de AWS cambian con frecuencia porque AWS ofrece AMI nuevas con las últimas actualizaciones y mejoras de seguridad. Puede usar los cmdlets Get-EC2Image y Get-EC2ImageByName para buscar las AMI de Windows actuales y obtener sus identificadores.

Temas

- Get-EC2Image
- Get-EC2ImageByName

Get-EC2Image

El cmdlet `Get-EC2Image` recupera una lista de las AMI que puede utilizar.

Utilice el parámetro `-Owner` con el valor de matriz `amazon, self` de modo que `Get-EC2Image` recupere solo las AMI que pertenecen a Amazon o a usted. En este contexto, hará referencia al usuario cuyas credenciales utilizó para invocar el cmdlet.

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

Puede definir el alcance de los resultados mediante el parámetro `-Filter`. Para especificar el filtro, cree un objeto de tipo `Amazon.EC2.Model.Filter`. Por ejemplo, utilice el siguiente filtro para mostrar solo las AMI de Windows.

```
$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
```

El siguiente es un ejemplo de una de las AMI devueltas por el cmdlet; la salida real del comando anterior proporciona información sobre muchas AMI.

```
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

El cmdlet `Get-EC2ImageByName` le permite filtrar la lista de AMI de Windows de AWS en función del tipo de configuración del servidor que desee.

Cuando se ejecuta sin parámetros, como se muestra a continuación, el cmdlet emite el conjunto completo de nombres de filtro actuales:

```
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
```

Para reducir el conjunto de imágenes devueltas, especifique uno o varios nombres de filtro mediante el parámetro `Names`.

```
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
```

Lance una EC2 instancia de Amazon con Windows PowerShell

Para lanzar una EC2 instancia de Amazon, necesitas el key pair y el grupo de seguridad que creaste en las secciones anteriores. También necesitas el ID de una Amazon Machine Image (AMI). Para obtener más información, consulte la siguiente documentación sobre :

- Creación de un par de claves
- Cree un grupo de seguridad con Windows PowerShell
- Encuentra una imagen de máquina de Amazon con Windows PowerShell

> **Important**
>
> Si lanza una instancia que no figura en la capa gratuita, se le facturará en cuanto la lance y se le cobrará el tiempo en que la instancia esté funcionando, aunque permanezca inactiva.

Temas

- Lanzamiento de una instancia en un VPC
- Lanzamiento de una instancia puntual en un VPC

Lanzamiento de una instancia en un VPC

> **Warning**
>
> EC2-Classic se retiró el 15 de agosto de 2022. Le recomendamos que migre de EC2 - Classic a un. VPC Para obtener más información, consulte la entrada del blog EC2-Classic Networking is Retiring: aquí le mostramos cómo prepararse.

El siguiente comando crea una sola instancia `m1.small` en la subred privada especificada. El grupo de seguridad debe ser válido para la subred especificada.

```
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       : {}
```

Al principio, la instancia tiene el estado `pending`, pero cambia al estado `running` en unos minutos. Para ver información sobre la instancia, use el cmdlet `Get-EC2Instance`. Si tiene varias instancias, puede filtrar los resultados por ID de reserva mediante el parámetro `Filter`. En primer lugar, cree un objeto de tipo `Amazon.EC2.Model.Filter`. A continuación, llame a `Get-EC2Instance` que utiliza el filtro y, a continuación, muestra la propiedad `Instances`.

```
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
```

Lanzamiento de una instancia puntual en un VPC

El script de ejemplo siguiente solicita una instancia de spot en la subred especificada. El grupo de seguridad debe ser uno que haya creado para él y VPC que contenga la subred especificada.

```
$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 y AWS Tools for PowerShell

Mediante el módulo AWSLambdaPSCore puede desarrollar funciones AWS Lambda en PowerShell Core 6.0 utilizando el tiempo de ejecución de .NET Core 2.1. Los desarrolladores de PowerShell pueden administrar los recursos de AWS y escribir scripts de automatización en el entorno de

PowerShell mediante el uso de Lambda. La compatibilidad de PowerShell en Lambda le permite ejecutar scripts o funciones de PowerShell en respuesta a cualquier evento de Lambda, como, por ejemplo, un evento de Amazon S3 o un evento de Amazon CloudWatch programado. El módulo AWSLambdaPSCore es un módulo separado de AWS para PowerShell; no forma parte de las AWS Tools for PowerShell, ni cuando se instala el módulo AWSLambdaPSCore se instalan las AWS Tools for PowerShell.

Luego de instalar el módulo AWSLambdaPSCore, puede usar cualquier cmdlet de PowerShell disponible, o desarrollar los suyos propios, para crear funciones sin servidor. El módulo AWS Lambda Tools for PowerShell incluye plantillas de proyecto para aplicaciones sin servidor basadas en PowerShell y herramientas para publicar proyectos en AWS.

La compatibilidad con el módulo AWSLambdaPSCore se encuentra disponible en todas las regiones que admiten Lambda. Para obtener más información acerca de las regiones admitidas, consulte la Tabla de regiones de AWS.

Requisitos previos

Los siguientes pasos son necesarios para poder instalar y utilizar el módulo AWSLambdaPSCore. Para obtener más información acerca de estos pasos, consulte Configuración de un entorno de desarrollo de PowerShell en la Guía para desarrolladores de AWS Lambda.

- Instale la versión correcta de PowerShell - la compatibilidad de Lambda para PowerShell se basa en la versión PowerShell Core 6.0 multiplataforma. Puede desarrollar funciones de Lambda de PowerShell en Windows, Linux o Mac. Si no dispone de al menos esta versión de PowerShell instalada, puede consultar las instrucciones en el sitio web de documentación de Microsoft PowerShell.
- Instale el SDK de .NET Core 2.1: dado que PowerShell Core se basa en .NET Core, la compatibilidad de Lambda con PowerShell utiliza el mismo tiempo de ejecución de Lambda de .NET Core 2.1 tanto para las funciones de Lambda de .NET Core como para las de PowerShell. Los cmdlets de publicación de Lambda de PowerShell utilizan el SDK de .NET Core 2.1 para crear el paquete de implementación de Lambda. El SDK de .NET Core 2.1 está disponible desde el Centro de descargas de Microsoft. Asegúrese de instalar el SDK, no el runtime.

Instale el módulo AWSLambdaPSCore.

Después de completar los requisitos previos, estará listo para instalar el módulo AWSLambdaPSCore. Ejecute el siguiente comando en una sesión de PowerShell Core.

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

Ya está preparado para comenzar a desarrollar funciones de Lambda en PowerShell. Para obtener más información acerca de cómo comenzar, consulte Modelo de programación para crear funciones Lambda en PowerShell en la Guía para desarrolladores de AWS Lambda.

Véase también

- Announcing Lambda Support for PowerShell Core en el Blog para desarrolladores de AWS
- Módulo AWSLambdaPSCore en el sitio web de PowerShell Gallery
- Configuración del entorno de desarrollo de PowerShell
- AWS Lambda Tools para Powershell en GitHub
- Consola de AWS Lambda

Amazon SQS, Amazon SNS y Tools for Windows PowerShell

Esta sección proporciona ejemplos que muestran cómo:

- crear una cola de Amazon SQS y obtener el ARN (Nombre de recurso de Amazon) de la cola
- Cree un tema de Amazon SNS.
- Conceder permisos al tema de SNS para que pueda enviar mensajes a la cola
- Suscribir la cola al tema de SNS
- conceder a usuarios de IAM o cuentas de AWS permisos para publicar en el tema de SNS y leer mensajes de la cola de SQS
- Verificar los resultados publicando un mensaje en el tema y leyendo el mensaje de la cola

crear una cola de Amazon SQS y obtener el ARN de la cola

El siguiente comando crea una cola de SQS en su región predeterminada. La salida muestra la URL de la nueva cola.

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

El siguiente comando recupera el ARN de la cola.

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

Cree un tema de Amazon SNS.

El siguiente comando crea un tema de SNS en la región predeterminada y devuelve el ARN del nuevo tema.

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

Conceder permisos al tema de SNS

El siguiente script de ejemplo crea una cola de SQS y un tema de SNS, y concede permisos al tema de SNS para que pueda enviar mensajes a la cola de SQS:

```
# 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 }
```

Suscribir la cola al tema de SNS

El siguiente comando suscribe la cola `myQueue` al tema de SNS `myTopic` y devuelve el ID de suscripción:

```
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
```

Conceder permisos

El siguiente comando concede permiso para realizar la acción `sns:Publish` en el tema `myTopic`.

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

El siguiente comando concede permiso para realizar las acciones `sqs:ReceiveMessage` y `sqs:DeleteMessage` en la cola `myQueue`.

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

Verificar los resultados

El siguiente comando prueba la nueva cola y el tema publicando un mensaje en el tema de SNS `myTopic` y devuelve el `MessageId`.

```
PS > Publish-SNSMessage `
```

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

El siguiente comando recupera el mensaje de la cola de SQS myQueue y lo muestra.

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

Attributes             : {}
Body                   : {
                           "Type" : "Notification",
                           "MessageId" : "491c687d-b78d-5c48-b7a0-3d8d769ee91b",
                           "TopicArn" : "arn:aws:sns:us-west-2:123456789012:myTopic",
                           "Message" : "Have A Nice Day!",
                           "Timestamp" : "2019-09-09T21:06:27.201Z",
                           "SignatureVersion" : "1",
                           "Signature" :
 "llE17A2+XOuJZnw3TlgcXz4C4KPLXZxbxoEMIirelhl3u/oxkWmz5+9tJKFMns1ZOqQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1bO3hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
                         vyuGkJReV7lOJWPHnCN
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yo
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
                         kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvr3WbaSvg==",
                           "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-6aad65c2f9911b05cd53efda11f913f9.pem",
                           "UnsubscribeURL" :
                         "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:myTopic:22b77de7-
a216-4000-9a23-bf465744ca84"
                         }
MD5OfBody              : 5b5ee4f073e9c618eda3718b594fa257
MD5OfMessageAttributes :
MessageAttributes      : {}
MessageId              : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
ReceiptHandle          :
 AQEB2vvk1e5cOKFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJbl7F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHnl8SlbQa0R/kD+Saqa4OIvfj8x3M4Oh1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
                         +HmXdkax2Wd+9AxrHlQZV5ur1MoByKWWbDbsqoYJTJquCclOgWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
                         sHN12776axknhg3j9K/Xwj54DixdsegnrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2MseyOeM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==
```

CloudWatch desde AWS Tools for Windows PowerShell

En esta sección se proporciona un ejemplo sobre cómo usar Tools for Windows PowerShell para publicar datos de métricas personalizadas en CloudWatch.

En este ejemplo se presupone que ya ha definido credenciales predeterminadas y una región predeterminada para su sesión de PowerShell.

Publicación de una métrica personalizada en el panel de CloudWatch

El siguiente código de PowerShell inicializa un objeto `MetricDatum` de CloudWatch y lo publica en el servicio. Puede ver el resultado de esta operación en la consola de CloudWatch.

```
$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
```

Tenga en cuenta lo siguiente:

- La información de fecha y hora que usa para inicializar `$dat.Timestamp` debe estar en formato UTC (hora universal).
- El valor que utiliza para inicializar `$dat.Value` puede ser un valor de cadena incluido entre comillas o un valor numérico (sin comillas). El ejemplo muestra un valor de cadena.

Véase también

- Trabaje con AWS los servicios del AWS Tools for PowerShell
- AmazonCloudWatchClient.PutMetricData (Referencia del SDK de .NET)
- MetricDatum (Referencia de la API del servicio)
- Consola de Amazon CloudWatch

Uso del parámetro ClientConfig en los cmdlets

El parámetro `ClientConfig` se puede usar para especificar ciertos parámetros de configuración cuando se conecta a un servicio. La mayoría de las propiedades posibles de este parámetro están definidas en la clase `Amazon.Runtime.ClientConfig`, que se hereda en las API de los servicios de AWS. Para ver un ejemplo de herencia simple, consulte la clase `Amazon.Keyspaces.AmazonKeyspacesConfig`. Además, algunos servicios definen propiedades adicionales que solo son apropiadas para ese servicio. Para ver un ejemplo de las propiedades adicionales que se han definido, consulte la clase `Amazon.S3.AmazonS3Config`, específicamente la propiedad `ForcePathStyle`.

Uso del parámetro **`ClientConfig`**

Para usar el parámetro `ClientConfig`, puede especificarlo en la línea de comandos como un objeto `ClientConfig` o usar el método splatting de PowerShell para pasar una colección de valores de parámetros a un comando como una unidad. Estos métodos se muestran en los siguientes ejemplos. En los ejemplos se supone que el módulo `AWS.Tools.S3` se ha instalado e importado y que tiene un perfil de credenciales `[default]` con los permisos adecuados.

Definición de un objeto **`ClientConfig`**

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

Cómo agregar propiedades **`ClientConfig`** mediante el uso del método splatting de PowerShell

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

Get-S3Object @params
```

Uso de una propiedad indefinida

Al utilizar PowerShell Splatting, si especifica una propiedad `ClientConfig` que no existe, AWS Tools for PowerShell no detectará el error hasta el tiempo de ejecución, momento en el que devolverá una excepción. Modificación del ejemplo anterior:

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

Get-S3Object @params
```

En este ejemplo se produce una excepción similar a la siguiente:

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

Especificación de la Región de AWS

Puede usar el parámetro `ClientConfig` para establecer la Región de AWS para el comando. La región se establece mediante la propiedad `RegionEndpoint`. AWS Tools for PowerShell calcula la región que se utilizará de acuerdo con la siguiente prioridad:

1. Parámetro `-Region`
2. Región incluida en el parámetro `ClientConfig`
3. Estado de sesión de PowerShell
4. Archivo `config` de AWS compartido
5. Variables de entorno
6. Metadatos de la instancia de Amazon EC2, si están activados.

Herramientas para ejemplos PowerShell de código

Los ejemplos de código de este tema muestran cómo usar el AWS Tools for PowerShell with AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Servicios

- ACMejemplos de uso de herramientas para PowerShell
- Ejemplos de Application Auto Scaling utilizando herramientas para PowerShell
- AppStream Ejemplos de la versión 2.0 que utilizan herramientas para PowerShell
- Ejemplos de Aurora con herramientas para PowerShell
- Ejemplos de Auto Scaling utilizando herramientas para PowerShell
- AWS Budgets ejemplos de uso de herramientas para PowerShell
- AWS Cloud9 ejemplos de uso de herramientas para PowerShell
- AWS CloudFormation ejemplos de uso de herramientas para PowerShell
- CloudFront ejemplos de uso de herramientas para PowerShell
- CloudTrail ejemplos de uso de herramientas para PowerShell
- CloudWatch ejemplos de uso de herramientas para PowerShell
- CodeCommit ejemplos de uso de herramientas para PowerShell
- CodeDeploy ejemplos de uso de herramientas para PowerShell
- CodePipeline ejemplos de uso de herramientas para PowerShell
- Ejemplos de Amazon Cognito Identity que utilizan herramientas para PowerShell
- AWS Config ejemplos de uso de herramientas para PowerShell
- Ejemplos de Device Farm que utilizan herramientas para PowerShell
- AWS Directory Service ejemplos de uso de herramientas para PowerShell
- AWS DMS ejemplos de uso de herramientas para PowerShell

- Ejemplos de DynamoDB que utilizan herramientas para PowerShell
- EC2Ejemplos de Amazon que utilizan herramientas para PowerShell
- ECREjemplos de Amazon que utilizan herramientas para PowerShell
- ECSEjemplos de Amazon que utilizan herramientas para PowerShell
- EFSEjemplos de Amazon que utilizan herramientas para PowerShell
- EKSEjemplos de Amazon que utilizan herramientas para PowerShell
- Ejemplos de Elastic Load Balancing: versión 1 con herramientas para PowerShell
- Ejemplos de Elastic Load Balancing: versión 2 con herramientas para PowerShell
- FSxEjemplos de Amazon que utilizan herramientas para PowerShell
- AWS Glue ejemplos de uso de herramientas para PowerShell
- AWS Health ejemplos de uso de herramientas para PowerShell
- IAMejemplos de uso de herramientas para PowerShell
- Ejemplos de Kinesis con herramientas para PowerShell
- Ejemplos de Lambda con herramientas para PowerShell
- Ejemplos de Amazon ML que utilizan herramientas para PowerShell
- Ejemplos de Macie que utilizan herramientas para PowerShell
- AWS OpsWorks ejemplos de uso de herramientas para PowerShell
- Lista de precios de AWS ejemplos de uso de herramientas para PowerShell
- Ejemplos de Resource Groups que utilizan herramientas para PowerShell
- Resource Groups: API ejemplos de etiquetado mediante herramientas para PowerShell
- Ejemplos de Route 53 con herramientas para PowerShell
- Ejemplos de Amazon S3 que utilizan herramientas para PowerShell
- Ejemplos de S3 Glacier que utilizan herramientas para PowerShell
- SESEjemplos de Amazon que utilizan herramientas para PowerShell
- SNSEjemplos de Amazon que utilizan herramientas para PowerShell
- SQSEjemplos de Amazon que utilizan herramientas para PowerShell
- AWS STS ejemplos de uso de herramientas para PowerShell
- AWS Support ejemplos de uso de herramientas para PowerShell
- Ejemplos de Systems Manager que utilizan herramientas para PowerShell
- Ejemplos de Amazon Translate que utilizan herramientas para PowerShell

- AWS WAFV2 ejemplos de uso de herramientas para PowerShell
- WorkSpaces ejemplos de uso de herramientas para PowerShell

ACMejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell withACM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-ACMCertificate`

El siguiente ejemplo de código muestra cómo usarlo`Get-ACMCertificate`.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra cómo devolver un certificado y su cadena utilizando ARN el certificado.

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

- Para API obtener más información, consulte GetCertificatela referencia del AWS Tools for PowerShell cmdlet.

`Get-ACMCertificateDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ACMCertificateDetail`

Herramientas para PowerShell

Ejemplo 1: devuelve los detalles del certificado especificado.

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

Salida:

```
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}
```

- Para API obtener más información, consulte DescribeCertificate AWS Tools for PowerShellCmdlet Reference.

`Get-ACMCertificateList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ACMCertificateList`

Herramientas para PowerShell

Ejemplo 1: recupera una lista de todos sus certificados ARNs y el nombre de dominio de cada uno. El cmdlet se paginará automáticamente para recuperar todos los. ARNs Para controlar manualmente la paginación, utilice el MaxItem parámetro - para controlar cuántos certificados

ARNs se devuelven por cada llamada de servicio y el NextToken parámetro - para indicar el punto de partida de cada llamada.

```
Get-ACMCertificateList
```

Salida:

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

Ejemplo 2: Recupera una lista de todos los certificados en los ARNs que el estado del certificado coincide con los estados proporcionados.

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

Ejemplo 3: Este ejemplo devuelve una lista de todos los certificados de la región us-east-1 que tienen un tipo RSA de clave de _2048 y un uso o propósito de clave extendido de _. CODE SIGNING Puede encontrar los valores de estos parámetros de filtrado en el tema de API referencia de ListCertificates filtros: https://docs.aws.amazon.com/acm/ latest//ilters.html. APIReference API_F

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

Salida:

```
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-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-a714-2d277d3eezzz
 *.route53docs.com
```

- Para API obtener más información, consulte la referencia de ListCertificates AWS Tools for PowerShellcmdlets.

`New-ACMCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ACMCertificate`

Herramientas para PowerShell

Ejemplo 1: crea un certificado nuevo. El servicio devuelve ARN el certificado nuevo.

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

Salida:

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

Ejemplo 2: crea un certificado nuevo. El servicio devuelve ARN el certificado nuevo.

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

Salida:

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

- Para API obtener más información, consulte RequestCertificatela referencia del AWS Tools for PowerShell cmdlet.

`Remove-ACMCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ACMCertificate`

Herramientas para PowerShell

Ejemplo 1: Elimina el certificado identificado por la clave privada proporcionada ARN y la asociada. El cmdlet solicitará la confirmación antes de continuar; añada el modificador -Force para suprimir la confirmación.

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

- Para API obtener más información, consulte la referencia del DeleteCertificatecmdlet AWS Tools for PowerShell .

`Send-ACMValidationEmail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Send-ACMValidationEmail`

Herramientas para PowerShell

Ejemplo 1: Solicita que se envíe el correo electrónico para validar la propiedad del dominio de «www.example.com». Si el valor $ del shell ConfirmPreference está configurado en «Medio» o en un valor inferior, el cmdlet solicitará una confirmación antes de continuar. Añada el modificador -Force para suprimir las solicitudes de confirmación.

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ResendValidationEmailReference.

Ejemplos de Application Auto Scaling utilizando herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Application Auto Scaling. AWS Tools for PowerShell

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-AASScalableTarget`

El siguiente ejemplo de código muestra cómo usarlo`Add-AASScalableTarget`.

Herramientas para PowerShell

Ejemplo 1: Este cmdlet registra o actualiza un objetivo escalable. Un objetivo escalable es un recurso que Application Auto Scaling puede escalar de manera horizontal y horizontal.

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

- Para API obtener más información, consulte RegisterScalableTargetla referencia de AWS Tools for PowerShell cmdlets.

`Get-AASScalableTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-AASScalableTarget`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo proporcionará información sobre los objetivos escalables de la aplicación de escalado automático en el espacio de nombres especificado.

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

Salida:

```
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
```

- Para API obtener más información, consulte la referencia de DescribeScalableTargetscmdlets AWS Tools for PowerShell .

`Get-AASScalingActivity`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-AASScalingActivity`

Herramientas para PowerShell

Ejemplo 1: proporciona información descriptiva sobre las actividades de escalado en el espacio de nombres especificado de las seis semanas anteriores.

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

Salida:

```
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.
```

- Para API obtener más información, consulte la referencia del DescribeScalingActivities AWS Tools for PowerShellcmdlet.

`Get-AASScalingPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-AASScalingPolicy`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet describe las políticas de escalado de Application Auto Scaling para el espacio de nombres del servicio especificado.

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

Salida:

```
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 :
```

- Para obtener API más información, consulte DescribeScalingPolicieslа referencia del cmdlet.AWS Tools for PowerShell

`Get-AASScheduledAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-AASScheduledAction`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet muestra las acciones programadas para su grupo de Auto Scaling que no se han ejecutado o que no han llegado a su hora de finalización.

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

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DescribeScheduledActions AWS Tools for PowerShellcmdlet.

`Remove-AASScalableTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-AASScalableTarget`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet anula el registro de un objetivo escalable de Application Auto Scaling. Al anular el registro de un destino escalable, se eliminan las políticas de escalado asociadas a él.

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

Salida:

```
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
```

- Para DeregisterScalableTargetobtener AWS Tools for PowerShell más información, consulte la referencia del cmdlet. API

`Remove-AASScalingPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-AASScalingPolicy`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina la política de escalado especificada para un destino escalable de Application Auto Scaling.

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

- Para API obtener más información, consulte la referencia del DeleteScalingPolicycmdlet AWS Tools for PowerShell .

`Remove-AASScheduledAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-AASScheduledAction`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina la acción programada especificada para un objetivo escalable de Application Auto Scaling.

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

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DeleteScheduledActioncmdlet AWS Tools for PowerShell .

`Set-AASScalingPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-AASScalingPolicy`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet crea o actualiza una política para un destino escalable de Application Auto Scaling. Cada objetivo escalable se identifica mediante un espacio de nombres de servicio, un identificador de recurso y una dimensión escalable.

```
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}
```

Salida:

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

- Para API obtener más información, consulte la referencia de PutScalingPolicy AWS Tools for PowerShellcmdlets.

`Set-AASScheduledAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-AASScheduledAction`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet crea o actualiza una acción programada para un objetivo escalable de Application Auto Scaling. Cada objetivo escalable se identifica mediante un espacio de nombres de servicio, un identificador de recurso y una dimensión escalable.

```
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
```

- Para API obtener más información, consulte la referencia de PutScheduledAction AWS Tools for PowerShellcmdlets.

AppStream Ejemplos de la versión 2.0 que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS Tools for PowerShell AppStream versión 2.0.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-APSResourceTag`

El siguiente ejemplo de código muestra cómo usarloAdd-APSResourceTag.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega una etiqueta de recurso al AppStream recurso

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

Salida:

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

- Para API obtener más información, consulte TagResource AWS Tools for PowerShellCmdlet Reference.

`Copy-APSImage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Copy-APSImage`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se copia una imagen a otra región

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

Salida:

```
TestImageCopy
```

- Para API obtener más información, consulte CopyImagela referencia del AWS Tools for PowerShell cmdlet.

`Disable-APSUSer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-APSUSer`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo desactiva a un usuario en USERPOOL

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

- Para API obtener más información, consulte DisableUser AWS Tools for PowerShellCmdlet Reference.

`Enable-APSUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-APSUser`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite a un usuario discapacitado USERPOOL

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

- Para API obtener más información, consulte EnableUser AWS Tools for PowerShellCmdlet Reference.

`Get-APSAssociatedFleetList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSAssociatedFleetList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra la flota asociada a una pila

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

Salida:

```
PowershellFleet
```

- Para API obtener más información, consulte ListAssociatedFleets AWS Tools for PowerShellCmdlet Reference.

`Get-APSAssociatedStackList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSAssociatedStackList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra la pila asociada a una flota

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

Salida:

```
PowershellStack
```

- Para API obtener más información, consulte ListAssociatedStacks AWS Tools for PowerShellCmdlet Reference.

`Get-APSDirectoryConfigList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSDirectoryConfigList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran las configuraciones de directorio creadas en AppStream

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeDirectoryConfigsla referencia del AWS Tools for PowerShell cmdlet.

`Get-APSFleetList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSFleetList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran los detalles de una flota

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeFleets AWS Tools for PowerShellCmdlet Reference.

`Get-APSImageBuilderList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSImageBuilderList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran los detalles de un ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeImageBuildersla referencia del AWS Tools for PowerShell cmdlet.

`Get-APSImageList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSImageList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra AppStream imágenes privadas

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeImages AWS Tools for PowerShellCmdlet Reference.

`Get-APSImagePermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSImagePermission`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran los permisos de imagen en una AppStream imagen compartida

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

Salida:

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

- Para API obtener más información, consulte DescribeImagePermissions AWS Tools for PowerShellCmdlet Reference.

`Get-APSSessionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSSessionList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra la lista de sesiones de una flota

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeSessions AWS Tools for PowerShellCmdlet Reference.

`Get-APSStackList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSStackList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra la lista de AppStream Stack

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeStacks AWS Tools for PowerShellCmdlet Reference.

`Get-APSTagsForResourceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSTagsForResourceList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran las etiquetas de un AppStream recurso

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

Salida:

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

- Para API obtener más información, consulte ListTagsForResource AWS Tools for PowerShellCmdlet Reference.

`Get-APSUsageReportSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSUsageReportSubscription`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran los detalles AppStreamUsageReport de configuración

```
Get-APSUsageReportSubscription
```

Salida:

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

- Para API obtener más información, consulte DescribeUsageReportSubscriptions AWS Tools for PowerShellCmdlet Reference.

`Get-APSUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSUser`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra una lista de usuarios con el estado habilitado

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

Salida:

```
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
```

- Para API obtener más información, consulte DescribeUsersla referencia de AWS Tools for PowerShell cmdlets.

`Get-APSUserStackAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-APSUserStackAssociation`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra una lista de usuarios asignados a una pila

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Salida:

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

- Para API obtener más información, consulte DescribeUserStackAssociationsla referencia de AWS Tools for PowerShell cmdlets.

`New-APSDirectoryConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSDirectoryConfig`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una configuración de directorio en AppStream

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

Salida:

```
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
```

- Para API obtener más información, consulte CreateDirectoryConfig AWS Tools for PowerShellCmdlet Reference.

`New-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva AppStream flota

```
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
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateFleet AWS Tools for PowerShellCmdlet Reference.

`New-APSImageBuilder`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSImageBuilder`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un Image Builder en 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
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateImageBuilder AWS Tools for PowerShellCmdlet Reference.

`New-APSImageBuilderStreamingURL`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSImageBuilderStreamingURL`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una ImageBuilder transmisión URL con una validez de 2 horas

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

Salida:

```
Expires               StreamingURL
-------               ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- Para API obtener más información, consulte CreateImageBuilderStreamingURL AWS Tools for PowerShellCmdlet Reference.

`New-APSStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSStack`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una AppStream pila nueva

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

Salida:

```
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}
```

- Para API obtener más información, consulte CreateStack AWS Tools for PowerShellCmdlet Reference.

`New-APSStreamingURL`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSStreamingURL`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una transmisión URL de Stack

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

Salida:

```
Expires                StreamingURL
-------                ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- Para API obtener más información, consulte CreateStreamingURL AWS Tools for PowerShellCmdlet Reference.

`New-APSUsageReportSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSUsageReportSubscription`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita los informes AppStream de uso

```
New-APSUsageReportSubscription
```

Salida:

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

- Para API obtener más información, consulte CreateUsageReportSubscriptionla referencia del AWS Tools for PowerShell cmdlet.

`New-APSUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-APSUser`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un usuario en USERPOOL

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

Salida:

```
Test@lab.com
```

- Para API obtener más información, consulte CreateUser AWS Tools for PowerShellCmdlet Reference.

`Register-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-APSFleet`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra la flota con una pila

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

- Para API obtener más información, consulte AssociateFleet AWS Tools for PowerShellCmdlet Reference.

`Register-APSUserStackBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-APSUserStackBatch`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asigna una pila a un usuario en USERPOOL

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

- Para API obtener más información, consulte la referencia BatchAssociateUserStackdel AWS Tools for PowerShell cmdlet.

`Remove-APSDirectoryConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSDirectoryConfig`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la configuración del AppStream directorio

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

Salida:

```
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
```

- Para API obtener más información, consulte DeleteDirectoryConfigla referencia del AWS Tools for PowerShell cmdlet.

`Remove-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se elimina una flota AppStream

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

Salida:

```
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
```

- Para API obtener más información, consulte DeleteFleet AWS Tools for PowerShellCmdlet Reference.

`Remove-APSImage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSImage`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina una imagen

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

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImage (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
```

- Para API obtener más información, consulte DeleteImage AWS Tools for PowerShellCmdlet Reference.

`Remove-APSImageBuilder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSImageBuilder`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina un ImageBuilder

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

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImageBuilder (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
```

- Para API obtener más información, consulte la referencia DeleteImageBuilderdel AWS Tools for PowerShell cmdlet.

`Remove-APSImagePermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSImagePermission`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina los permisos de una imagen

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

Salida:

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

- Para API obtener más información, consulte DeleteImagePermissions AWS Tools for PowerShellCmdlet Reference.

`Remove-APSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSResourceTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina una etiqueta de recurso del AppStream recurso

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

Salida:

```
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
```

- Para API obtener más información, consulte UntagResource AWS Tools for PowerShellCmdlet Reference.

`Remove-APSStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSStack`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina una pila

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

Salida:

```
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
```

- Para API obtener más información, consulte DeleteStack AWS Tools for PowerShellCmdlet Reference.

`Remove-APSUsageReportSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSUsageReportSubscription`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo desactiva la suscripción a los informes AppStream de uso

```
Remove-APSUsageReportSubscription
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteUsageReportSubscription AWS Tools for PowerShellCmdlet Reference.

`Remove-APSUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-APSUser`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina un usuario de USERPOOL

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

Salida:

```
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
```

- Para API obtener más información, consulte DeleteUser AWS Tools for PowerShellCmdlet Reference.

`Revoke-APSSession`

En el siguiente ejemplo de código se muestra cómo usarlo. `Revoke-APSSession`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo revoca una sesión con la flota AppStream

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

- Para API obtener más información, consulte ExpireSession AWS Tools for PowerShellCmdlet Reference.

`Start-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-APSFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se inicia una flota

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

- Para API obtener más información, consulte StartFleet AWS Tools for PowerShellCmdlet Reference.

`Start-APSImageBuilder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-APSImageBuilder`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo inicia un ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

Salida:

```
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
```

- Para API obtener más información, consulte StartImageBuilderla referencia del AWS Tools for PowerShell cmdlet.

`Stop-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-APSFleet`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo detiene una flota

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

- Para API obtener más información, consulte StopFleetla referencia del AWS Tools for PowerShell cmdlet.

`Stop-APSImageBuilder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-APSImageBuilder`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo detiene un ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

Salida:

```
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
```

- Para API obtener más información, consulte StopImageBuilderla referencia del AWS Tools for PowerShell cmdlet.

`Unregister-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-APSFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se anula el registro de una flota de la pila

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

- Para API obtener más información, consulte DisassociateFleet AWS Tools for PowerShellCmdlet Reference.

`Unregister-APSUserStackBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-APSUserStackBatch`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina un usuario de una pila asignada

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

- Para API obtener más información, consulte BatchDisassociateUserStack AWS Tools for PowerShellCmdlet Reference.

`Update-APSDirectoryConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-APSDirectoryConfig`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo actualiza la configuración del directorio creada en AppStream

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

Salida:

```
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
```

- Para API obtener más información, consulte UpdateDirectoryConfigla referencia del AWS Tools for PowerShell cmdlet.

`Update-APSFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-APSFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualizan las propiedades de una flota

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

Salida:

```
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
```

- Para API obtener más información, consulte UpdateFleet AWS Tools for PowerShellCmdlet Reference.

`Update-APSImagePermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-APSImagePermission`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se comparte una AppStream imagen con otra cuenta

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

- Para API obtener más información, consulte UpdateImagePermissions AWS Tools for PowerShellCmdlet Reference.

`Update-APSStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-APSStack`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo actualiza (habilita) la persistencia de la configuración de la aplicación y las carpetas de inicio de una pila

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

Salida:

```
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}
```

- Para API obtener más información, consulte UpdateStackla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Aurora con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS Tools for PowerShell de Aurora.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-RDSOrderableDBInstanceOption`

El siguiente ejemplo de código muestra cómo usarlo`Get-RDSOrderableDBInstanceOption`.

Herramientas para PowerShell

Ejemplo 1: este ejemplo enumera las versiones del motor de base de datos que admiten una clase de instancia de base de datos concreta en una Región de AWS.

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

Ejemplo 2: este ejemplo enumera las clases de instancia de base de datos que se admiten para una versión de motor de base de datos concreta en una Región de AWS.

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

- Para API obtener más información, consulte DescribeOrderableDBInstanceOptionsla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Auto Scaling utilizando herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS Tools for PowerShell uso de Auto Scaling.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-ASLoadBalancer`

El siguiente ejemplo de código muestra cómo usarloAdd-ASLoadBalancer.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se adjunta el balanceador de cargas especificado al grupo de Auto Scaling especificado.

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

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet AttachLoadBalancersReference.

`Complete-ASLifecycleAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Complete-ASLifecycleAction`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo completa la acción del ciclo de vida especificada.

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

- Para API obtener más información, consulte CompleteLifecycleActionla referencia del AWS Tools for PowerShell cmdlet.

`Disable-ASMetricsCollection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-ASMetricsCollection`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo deshabilita la supervisión de las métricas especificadas para el grupo de Auto Scaling especificado.

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

Ejemplo 2: Este ejemplo deshabilita la supervisión de todas las métricas del grupo de Auto Scaling especificado.

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

- Para API obtener más información, consulte DisableMetricsCollection AWS Tools for PowerShellCmdlet Reference.

`Dismount-ASInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-ASInstance`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo separa la instancia especificada del grupo de Auto Scaling especificado y reduce la capacidad deseada para que Auto Scaling no lance una instancia de reemplazo.

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

Salida:

```
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        :
```

Ejemplo 2: Este ejemplo separa la instancia especificada del grupo de Auto Scaling especificado sin reducir la capacidad deseada. Auto Scaling lanza una instancia de reemplazo.

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

Salida:

```
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        :
```

- Para API obtener más información, consulte DetachInstances AWS Tools for PowerShellCmdlet Reference.

`Dismount-ASLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-ASLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo separa el balanceador de cargas especificado del grupo de Auto Scaling especificado.

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

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DetachLoadBalancersReference.

`Enable-ASMetricsCollection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-ASMetricsCollection`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite monitorear las métricas especificadas para el grupo de Auto Scaling especificado.

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

Ejemplo 2: Este ejemplo permite monitorear todas las métricas del grupo de Auto Scaling especificado.

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

- Para API obtener más información, consulte EnableMetricsCollection AWS Tools for PowerShellCmdlet Reference.

`Enter-ASStandby`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enter-ASStandby`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo pone la instancia especificada en modo de espera y reduce la capacidad deseada para que Auto Scaling no lance una instancia de reemplazo.

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

Salida:

```
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        :
```

Ejemplo 2: Este ejemplo pone la instancia especificada en modo de espera sin reducir la capacidad deseada. Auto Scaling lanza una instancia de reemplazo.

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

Salida:

```
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        :
```

- Para API obtener más información, consulte EnterStandby AWS Tools for PowerShellCmdlet Reference.

`Exit-ASStandby`

En el siguiente ejemplo de código se muestra cómo usarlo. `Exit-ASStandby`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo saca la instancia especificada del modo de espera.

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

Salida:

```
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        :
```

- Para API obtener más información, consulte ExitStandbyla referencia de AWS Tools for PowerShell cmdlets.

`Get-ASAccountLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAccountLimit`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe los límites de recursos de Auto Scaling para su AWS cuenta.

```
Get-ASAccountLimit
```

Salida:

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

- Para API obtener más información, consulte DescribeAccountLimits AWS Tools for PowerShellCmdlet Reference.

`Get-ASAdjustmentType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAdjustmentType`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe los tipos de ajustes que admite Auto Scaling.

```
Get-ASAdjustmentType
```

Salida:

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

- Para API obtener más información, consulte DescribeAdjustmentTypes AWS Tools for PowerShellCmdlet Reference.

`Get-ASAutoScalingGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAutoScalingGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra los nombres de los grupos de Auto Scaling.

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

Salida:

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

Ejemplo 2: Este ejemplo describe el grupo de Auto Scaling especificado.

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

Salida:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
```

```
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Ejemplo 3: Este ejemplo describe los dos grupos de Auto Scaling especificados.

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

Ejemplo 4: Este ejemplo describe las instancias de Auto Scaling del grupo de Auto Scaling especificado.

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

Ejemplo 5: Este ejemplo describe todos los grupos de Auto Scaling.

```
Get-ASAutoScalingGroup
```

Ejemplo 6: Este ejemplo describe todos los grupos de Auto Scaling, en lotes de 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

Ejemplo 7: Este LaunchTemplate ejemplo describe el grupo de Auto Scaling especificado. En este ejemplo se supone que las «Opciones de compra de instancias» están configuradas en «Adherirse a la plantilla de lanzamiento». En caso de que esta opción esté configurada como «Combinar opciones de compra y tipos de instancias», se LaunchTemplate puede acceder a ella mediante»MixedInstancesPolicy. LaunchTemplate«propiedad.

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

Salida:

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

- Para API obtener más información, consulte DescribeAutoScalingGroups AWS Tools for PowerShellCmdlet Reference.

`Get-ASAutoScalingInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAutoScalingInstance`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las instancias IDs de Auto Scaling.

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

Salida:

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

Ejemplo 2: Este ejemplo describe la instancia de Auto Scaling especificada.

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

Salida:

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

Ejemplo 3: Este ejemplo describe las dos instancias de Auto Scaling especificadas.

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

Ejemplo 4: Este ejemplo describe las instancias de Auto Scaling del grupo de Auto Scaling especificado.

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

Ejemplo 5: Este ejemplo describe todas las instancias de Auto Scaling.

```
Get-ASAutoScalingInstance
```

Ejemplo 6: Este ejemplo describe todas las instancias de Auto Scaling, en lotes de 10.

```
$nextToken = $null
do {
  Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte la referencia DescribeAutoScalingInstancesde AWS Tools for PowerShell cmdlets.

`Get-ASAutoScalingNotificationType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAutoScalingNotificationType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los tipos de notificaciones que admite Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Salida:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
```

```
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Para API obtener más información, consulte DescribeAutoScalingNotificationTypes AWS Tools for PowerShellCmdlet Reference.

`Get-ASLaunchConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASLaunchConfiguration`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los nombres de las configuraciones de lanzamiento.

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

Salida:

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

Ejemplo 2: en este ejemplo se describe la configuración de lanzamiento especificada.

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

Salida:

```
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                     :
```

Ejemplo 3: en este ejemplo se describen las dos configuraciones de lanzamiento especificadas.

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

Ejemplo 4: En este ejemplo se describen todas las configuraciones de lanzamiento.

```
Get-ASLaunchConfiguration
```

Ejemplo 5: En este ejemplo se describen todas las configuraciones de lanzamiento, en lotes de 10.

```
$nextToken = $null
do {
  Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte la referencia DescribeLaunchConfigurationsde AWS Tools for PowerShell cmdlets.

`Get-ASLifecycleHook`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASLifecycleHook`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el enlace del ciclo de vida especificado.

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

Salida:

```
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
```

Ejemplo 2: Este ejemplo describe todos los enlaces del ciclo de vida del grupo de Auto Scaling especificado.

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

Ejemplo 3: Este ejemplo describe todos los enlaces del ciclo de vida de todos los grupos de Auto Scaling.

```
Get-ASLifecycleHook
```

- Para API obtener más información, consulte DescribeLifecycleHooks AWS Tools for PowerShellCmdlet Reference.

`Get-ASLifecycleHookType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASLifecycleHookType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los tipos de enlaces del ciclo de vida compatibles con Auto Scaling.

```
Get-ASLifecycleHookType
```

Salida:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Para API obtener más información, consulte DescribeLifecycleHookTypes AWS Tools for PowerShellCmdlet Reference.

`Get-ASLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe los balanceadores de carga para el grupo de Auto Scaling especificado.

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

Salida:

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

- Para API obtener más información, consulte DescribeLoadBalancers AWS Tools for PowerShellCmdlet Reference.

`Get-ASMetricCollectionType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASMetricCollectionType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los tipos de recopilación de métricas que admite Auto Scaling.

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

Salida:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Ejemplo 2: En este ejemplo se enumeran las granularidades correspondientes.

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

Salida:

```
Granularity
-----------
1Minute
```

- Para API obtener más información, consulte DescribeMetricCollectionTypes AWS Tools for PowerShellCmdlet Reference.

`Get-ASNotificationConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASNotificationConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe las acciones de notificación asociadas al grupo de Auto Scaling especificado.

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

Salida:

```
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
```

Ejemplo 2: Este ejemplo describe las acciones de notificación asociadas a todos los grupos de Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Para API obtener más información, consulte DescribeNotificationConfigurations AWS Tools for PowerShellCmdlet Reference.

`Get-ASPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe todas las políticas del grupo de Auto Scaling especificado.

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

Salida:

```
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         : {}
```

Ejemplo 2: Este ejemplo describe las políticas especificadas para el grupo de Auto Scaling especificado.

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

Ejemplo 3: Este ejemplo describe todas las políticas de todos los grupos de Auto Scaling.

```
Get-ASPolicy
```

- Para API obtener más información, consulte DescribePolicies AWS Tools for PowerShellCmdlet Reference.

`Get-ASScalingActivity`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASScalingActivity`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe las actividades de escalado de las últimas seis semanas para el grupo de Auto Scaling especificado.

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

Salida:

```
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        :
```

Ejemplo 2: En este ejemplo se describe la actividad de escalado especificada.

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

Ejemplo 3: Este ejemplo describe las actividades de escalado de las últimas seis semanas para todos los grupos de Auto Scaling.

```
Get-ASScalingActivity
```

- Para API obtener más información, consulte DescribeScalingActivitiesla referencia de AWS Tools for PowerShell cmdlets.

`Get-ASScalingProcessType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASScalingProcessType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los tipos de procesos que admite Auto Scaling.

```
Get-ASScalingProcessType
```

Salida:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Para API obtener más información, consulte DescribeScalingProcessTypes AWS Tools for PowerShellCmdlet Reference.

`Get-ASScheduledAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASScheduledAction`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe las acciones de escalado programadas para el grupo de Auto Scaling especificado.

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

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen las acciones de escalado programadas especificadas.

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

Ejemplo 3: En este ejemplo se describen las acciones de escalado programadas que comienzan a la hora especificada.

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

Ejemplo 4: En este ejemplo se describen las acciones de escalado programadas que finalizan a la hora especificada.

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

Ejemplo 5: Este ejemplo describe las acciones de escalado programadas para todos los grupos de Auto Scaling.

```
Get-ASScheduledAction
```

- Para API obtener más información, consulte DescribeScheduledActions AWS Tools for PowerShellCmdlet Reference.

`Get-ASTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen las etiquetas con un valor clave de «myTag» o «myTag2». Los valores posibles para el nombre del filtro son auto-scaling-group «», «clave», «valor» y «propagate-at-launch». La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

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

Salida:

```
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
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear el filtro para el parámetro Filter.

```
$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 )
```

Ejemplo 3: Este ejemplo describe todas las etiquetas de todos los grupos de Auto Scaling.

```
Get-ASTag
```

- Para API obtener más información, consulte DescribeTags AWS Tools for PowerShellCmdlet Reference.

`Get-ASTerminationPolicyType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASTerminationPolicyType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las políticas de terminación compatibles con Auto Scaling.

```
Get-ASTerminationPolicyType
```

Salida:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Para API obtener más información, consulte DescribeTerminationPolicyTypes AWS Tools for PowerShellCmdlet Reference.

`Mount-ASInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Mount-ASInstance`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se adjunta la instancia especificada al grupo de Auto Scaling especificado. Auto Scaling aumenta automáticamente la capacidad deseada del grupo Auto Scaling.

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

- Para API obtener más información, consulte AttachInstances AWS Tools for PowerShellCmdlet Reference.

`New-ASAutoScalingGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ASAutoScalingGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un grupo de Auto Scaling con el nombre y los atributos especificados. La capacidad deseada por defecto es el tamaño mínimo. Por lo tanto, este grupo de Auto Scaling lanza dos instancias, una en cada una de las dos zonas de disponibilidad especificadas.

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

- Para API obtener más información, consulte CreateAutoScalingGroupla referencia de AWS Tools for PowerShell cmdlets.

`New-ASLaunchConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ASLaunchConfiguration`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una configuración de inicio denominada «my-lc». Las EC2 instancias lanzadas por los grupos de Auto Scaling que utilizan esta configuración de lanzamiento utilizan el tipo de instanciaAMI, el grupo de seguridad y el IAM rol especificados.

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

- Para API obtener más información, consulte CreateLaunchConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-ASAutoScalingGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASAutoScalingGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el grupo de Auto Scaling especificado si no tiene instancias en ejecución. Se le solicitará una confirmación antes de continuar con la operación.

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

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

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

Ejemplo 3: En este ejemplo se elimina el grupo de Auto Scaling especificado y se finalizan todas las instancias en ejecución que contenga.

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

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteAutoScalingGroupReference.

`Remove-ASLaunchConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASLaunchConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la configuración de lanzamiento especificada si no está asociada a un grupo de Auto Scaling. Se le solicitará una confirmación antes de continuar con la operación.

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

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

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

- Para API obtener más información, consulte DeleteLaunchConfiguration AWS Tools for PowerShellCmdlet Reference.

`Remove-ASLifecycleHook`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASLifecycleHook`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el enlace del ciclo de vida especificado para el grupo de Auto Scaling especificado. Se le solicitará una confirmación antes de continuar con la operación.

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

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

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

- Para API obtener más información, consulte DeleteLifecycleHook AWS Tools for PowerShellCmdlet Reference.

`Remove-ASNotificationConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASNotificationConfiguration`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la acción de notificación especificada. Se le solicitará una confirmación antes de continuar con la operación.

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

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

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

- Para API obtener más información, consulte DeleteNotificationConfiguration AWS Tools for PowerShellCmdlet Reference.

`Remove-ASPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la política especificada para el grupo de Auto Scaling especificado. Se le solicitará una confirmación antes de continuar con la operación.

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

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

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

- Para API obtener más información, consulte DeletePolicy AWS Tools for PowerShellCmdlet Reference.

`Remove-ASScheduledAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASScheduledAction`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la acción programada especificada para el grupo de Auto Scaling especificado. Se le solicitará una confirmación antes de continuar con la operación.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le solicitará la confirmación antes de continuar con la operación.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Para API obtener más información, consulte DeleteScheduledAction AWS Tools for PowerShellCmdlet Reference.

`Remove-ASTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ASTag`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la etiqueta especificada del grupo de Auto Scaling especificado. Se le solicitará una confirmación antes de continuar con la operación. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Salida:

```
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"):
```

Ejemplo 2: Si especifica el parámetro Force, no se le pedirá confirmación antes de continuar con la operación.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Ejemplo 3: Con la versión 2 de Powershell, debe usar New-Object para crear la etiqueta del parámetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Para API obtener más información, consulte Cmdlet DeleteTagsReference AWS Tools for PowerShell .

`Resume-ASProcess`

En el siguiente ejemplo de código se muestra cómo usarlo. `Resume-ASProcess`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo reanuda el proceso de Auto Scaling especificado para el grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Ejemplo 2: Este ejemplo reanuda todos los procesos de Auto Scaling suspendidos para el grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Para API obtener más información, consulte ResumeProcesses AWS Tools for PowerShellCmdlet Reference.

`Set-ASDesiredCapacity`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ASDesiredCapacity`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo establece el tamaño del grupo de Auto Scaling especificado.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Ejemplo 2: Este ejemplo establece el tamaño del grupo de Auto Scaling especificado y espera a que finalice el período de enfriamiento antes de escalar al nuevo tamaño.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown
 $true
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet SetDesiredCapacityReference.

`Set-ASInstanceHealth`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ASInstanceHealth`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se establece el estado de la instancia especificada en «En mal estado», lo que la deja fuera de servicio. Auto Scaling finaliza y reemplaza la instancia.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Ejemplo 2: en este ejemplo se establece el estado de la instancia especificada en «En buen estado» y se mantiene en servicio. No se respeta ningún período de gracia para comprobar el estado del grupo Auto Scaling.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Para API obtener más información, consulte SetInstanceHealth AWS Tools for PowerShellCmdlet Reference.

`Set-ASInstanceProtection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ASInstanceProtection`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita la protección de instancias para la instancia especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Ejemplo 2: Este ejemplo inhabilita la protección de instancias para la instancia especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Para API obtener más información, consulte SetInstanceProtection AWS Tools for PowerShellCmdlet Reference.

`Set-ASTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ASTag`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega una sola etiqueta al grupo de Auto Scaling especificado. La clave de la etiqueta es 'myTag' y el valor de la etiqueta es 'myTagValue'. Auto Scaling propaga esta etiqueta a las EC2 instancias posteriores lanzadas por el grupo Auto Scaling. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o posterior.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear la etiqueta para el parámetro Tag.

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateOrUpdateTagsReference.

`Start-ASPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-ASPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo ejecuta la política especificada para el grupo de Auto Scaling especificado.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Ejemplo 2: Este ejemplo ejecuta la política especificada para el grupo de Auto Scaling especificado, después de esperar a que finalice el período de enfriamiento.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ExecutePolicyReference.

`Stop-ASInstanceInAutoScalingGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-ASInstanceInAutoScalingGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo termina la instancia especificada y reduce la capacidad deseada de su grupo de Auto Scaling para que Auto Scaling no lance una instancia de reemplazo.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Salida:

```
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        :
```

Ejemplo 2: Este ejemplo termina la instancia especificada sin disminuir la capacidad deseada de su grupo de Auto Scaling. Auto Scaling lanza una instancia de reemplazo.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Salida:

```
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        :
```

- Para API obtener más información, consulte TerminateInstanceInAutoScalingGroup AWS Tools for PowerShellCmdlet Reference.

`Suspend-ASProcess`

En el siguiente ejemplo de código se muestra cómo usarlo. `Suspend-ASProcess`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo suspende el proceso de Auto Scaling especificado para el grupo de Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Ejemplo 2: Este ejemplo suspende todos los procesos de Auto Scaling del grupo de Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Para API obtener más información, consulte SuspendProcesses AWS Tools for PowerShellCmdlet Reference.

`Update-ASAutoScalingGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-ASAutoScalingGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo actualiza el tamaño mínimo y máximo del grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Ejemplo 2: Este ejemplo actualiza el período de enfriamiento predeterminado del grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Ejemplo 3: Este ejemplo actualiza las zonas de disponibilidad del grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Ejemplo 4: Este ejemplo actualiza el grupo de Auto Scaling especificado para usar las comprobaciones de estado de Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Para API obtener más información, consulte UpdateAutoScalingGroup AWS Tools for PowerShellCmdlet Reference.

`Write-ASLifecycleActionHeartbeat`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-ASLifecycleActionHeartbeat`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra un latido de la acción del ciclo de vida especificada. Esto mantiene la instancia en estado pendiente hasta que completes la acción personalizada.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para API obtener más información, consulte RecordLifecycleActionHeartbeat AWS Tools for PowerShellCmdlet Reference.

`Write-ASLifecycleHook`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-ASLifecycleHook`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega el enlace de ciclo de vida especificado al grupo de Auto Scaling especificado.

```
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"
```

- Para API obtener más información, consulte PutLifecycleHook AWS Tools for PowerShellCmdlet Reference.

`Write-ASNotificationConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-ASNotificationConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo configura el grupo de Auto Scaling especificado para enviar una notificación al SNS tema especificado cuando lance EC2 instancias.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-
topic"
```

Ejemplo 2: Este ejemplo configura el grupo de Auto Scaling especificado para enviar una notificación al SNS tema especificado cuando lance o finalice instanciasEC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet PutNotificationConfigurationReference.

`Write-ASScalingPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-ASScalingPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega la política especificada al grupo de Auto Scaling especificado. El tipo de ajuste especificado determina cómo interpretar el ScalingAdjustment parámetro. Con 'ChangeInCapacity', un valor positivo aumenta la capacidad en el número especificado de instancias y un valor negativo reduce la capacidad en el número especificado de instancias.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Salida:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Para API obtener más información, consulte PutScalingPolicy AWS Tools for PowerShellCmdlet Reference.

`Write-ASScheduledUpdateGroupAction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-ASScheduledUpdateGroupAction`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea o actualiza una acción programada única para cambiar la capacidad deseada a la hora de inicio especificada.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
 "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Para API obtener más información, consulte PutScheduledUpdateGroupAction AWS Tools for PowerShellCmdlet Reference.

AWS Budgets ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Budgets.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`New-BGTBudget`

En el siguiente ejemplo de código se muestra cómo usarlo`New-BGTBudget`.

Ejemplo 1: Crea un nuevo presupuesto con las restricciones presupuestarias y de tiempo especificadas mediante notificaciones por correo electrónico.

```
$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
```

- Para API obtener más información, consulte CreateBudget AWS Tools for PowerShellCmdlet Reference.

AWS Cloud9 ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Cloud9.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-C9EnvironmentData`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-C9EnvironmentData`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre los entornos de desarrollo de AWS Cloud9 especificados.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Salida:

```
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
```

Ejemplo 2: Este ejemplo obtiene información sobre el estado del ciclo de vida del entorno de desarrollo AWS Cloud9 especificado.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Salida:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Para API obtener más información, consulte la referencia DescribeEnvironmentsde AWS Tools for PowerShell cmdlets.

`Get-C9EnvironmentList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-C9EnvironmentList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene una lista de los identificadores del entorno de desarrollo de AWS Cloud9 disponibles.

```
Get-C9EnvironmentList
```

Salida:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Para API obtener más información, consulte la referencia de ListEnvironments AWS Tools for PowerShellcmdlets.

`Get-C9EnvironmentMembershipList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-C9EnvironmentMembershipList`

Ejemplo 1: en este ejemplo se obtiene información sobre los miembros del entorno de desarrollo AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Salida:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Ejemplo 2: Este ejemplo obtiene información sobre el propietario del entorno de desarrollo AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Salida:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Ejemplo 3: Este ejemplo obtiene información sobre el miembro del entorno especificado para varios entornos de desarrollo de AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Salida:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
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        : AIDAJ3LOROMOUXTBSU6EX
```

- Para API obtener más información, consulte la referencia DescribeEnvironmentMembershipsde AWS Tools for PowerShell cmdlets.

`Get-C9EnvironmentStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-C9EnvironmentStatus`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene la información de estado del entorno de desarrollo AWS Cloud9 especificado.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Salida:

```
Message                    Status
-------                    ------
Environment is ready to use ready
```

- Para API obtener más información, consulte la referencia DescribeEnvironmentStatusde AWS Tools for PowerShell cmdlets.

`New-C9EnvironmentEC2`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-C9EnvironmentEC2`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un entorno de desarrollo de AWS Cloud9 con la configuración especificada, se lanza una instancia de Amazon Elastic Compute Cloud EC2 (Amazon) y, a continuación, se conecta desde la instancia al entorno.

```
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
```

Salida:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para API obtener más información, consulte la sección CreateEnvironmentEc2 en la referencia de AWS Tools for PowerShell cmdlets.

`New-C9EnvironmentMembership`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-C9EnvironmentMembership`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega el miembro del entorno especificado al entorno de desarrollo AWS Cloud9 especificado.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

Salida:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para API obtener más información, consulte la referencia CreateEnvironmentMembershipde AWS Tools for PowerShell cmdlets.

`Remove-C9Environment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-C9Environment`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el entorno de desarrollo AWS Cloud9 especificado. Si una EC2 instancia de Amazon está conectada al entorno, también termina la instancia.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para API obtener más información, consulte la referencia DeleteEnvironmentde AWS Tools for PowerShell cmdlets.

`Remove-C9EnvironmentMembership`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-C9EnvironmentMembership`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el miembro del entorno especificado del entorno de desarrollo AWS Cloud9 especificado.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para API obtener más información, consulte la referencia de DeleteEnvironmentMembership AWS Tools for PowerShellcmdlets.

`Update-C9Environment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-C9Environment`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo cambia la configuración especificada del entorno de desarrollo AWS Cloud9 existente especificado.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Para API obtener más información, consulte la referencia UpdateEnvironmentde AWS Tools for PowerShell cmdlets.

`Update-C9EnvironmentMembership`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-C9EnvironmentMembership`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo cambia la configuración del miembro del entorno existente especificado para el entorno de desarrollo AWS Cloud9 especificado.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Salida:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para API obtener más información, consulte la referencia UpdateEnvironmentMembershipde AWS Tools for PowerShell cmdlets.

AWS CloudFormation ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS CloudFormation.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-CFNStack`.

Herramientas para PowerShell

Ejemplo 1: Devolución de una colección de instancias de pila que describen todas las pilas del usuario.

```
Get-CFNStack
```

Ejemplo 2: Devolución de una instancia de pila que describe la pila especificada.

```
Get-CFNStack -StackName "myStack"
```

Ejemplo 3: Devolución de una colección de instancias de pila que describen todas las pilas del usuario con una paginación manual. El token de inicio de la página siguiente se recupera después de cada llamada y $null indica que no se pueden recuperar más detalles.

```
$nextToken = $null
do {
    Get-CFNStack -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte DescribeStacksla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFNStackEvent`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNStackEvent`

Herramientas para PowerShell

Ejemplo 1: Devolución de todos los eventos relacionados con la pila especificada.

```
Get-CFNStackEvent -StackName "myStack"
```

Ejemplo 2: Devolución de todos los eventos relacionados con la pila especificada con una paginación manual que comienza en el token especificado. El token de inicio de la página siguiente se recupera después de cada llamada y $null indica que no quedan más eventos por recuperar.

```
$nextToken = $null
do {
    Get-CFNStack -StackName "myStack" -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte DescribeStackEventsla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFNStackResource`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNStackResource`

Herramientas para PowerShell

Ejemplo 1: devuelve la descripción de un recurso identificado en la plantilla asociada a la pila especificada con el identificador lógico yDBInstance «M».

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Para API obtener más información, consulte DescribeStackResourcela referencia del AWS Tools for PowerShell cmdlet.

`Get-CFNStackResourceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNStackResourceList`

Herramientas para PowerShell

Ejemplo 1: devuelve las descripciones AWS de los recursos de un máximo de 100 recursos asociados a la pila especificada. Para obtener detalles de todos los recursos asociados a una

pila, utilice el comando Get-CFNStackResourceSummary, que también permite la paginación manual de los resultados.

```
Get-CFNStackResourceList -StackName "myStack"
```

Ejemplo 2: devuelve la descripción de la EC2 instancia de Amazon identificada en la plantilla asociada a la pila especificada con el identificador lógico «Ec2Instance».

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Ejemplo 3: devuelve la descripción de hasta 100 recursos asociados a la pila que contiene una instancia de Amazon identificada con el ID de EC2 instancia «i-123456". Para obtener detalles de todos los recursos asociados a una pila, usa el comando Get-CFNStackResourceSummary, que también permite la paginación manual de los resultados.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Ejemplo 4: Devuelve la descripción de la EC2 instancia de Amazon identificada por el ID lógico «Ec2Instance» en la plantilla de una pila. La pila se identifica mediante el ID de recurso físico de un recurso que contiene, en este caso también una instancia de Amazon con el ID de EC2 instancia «i-123456". También se puede utilizar un recurso físico distinto para identificar la pila según el contenido de la plantilla; por ejemplo, un bucket de Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Para obtener API más información, consulte DescribeStackResourcesla referencia de cmdlets.AWS Tools for PowerShell

`Get-CFNStackResourceSummary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNStackResourceSummary`

Herramientas para PowerShell

Ejemplo 1: Devolución de las descripciones de todos los recursos asociados a la pila especificada.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

Ejemplo 2: Devolución de las descripciones de todos los recursos asociados a la pila especificada con una paginación manual de los resultados. El token de inicio de la página siguiente se recupera después de cada llamada y $null indica que no se pueden recuperar más detalles.

```
$nextToken = $null
do {
  Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte ListStackResourcesla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFNStackSummary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNStackSummary`

Herramientas para PowerShell

Ejemplo 1: Devolución de información de resumen de todas las pilas.

```
Get-CFNStackSummary
```

Ejemplo 2: Devolución de información de resumen de todas las pilas que se crean actualmente.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Ejemplo 3: Devolución de información de resumen de todas las pilas que se crean o se actualizan actualmente.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

Ejemplo 4: Devolución de información de resumen de todas las pilas que se crean o se actualizan actualmente con una paginación manual de los resultados. El token de inicio de la página siguiente se recupera después de cada llamada y $null indica que no se pueden recuperar más detalles.

```
$nextToken = $null
do {
```

```
  Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS") -NextToken $nextToken
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte ListStacksla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFNTemplate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFNTemplate`

Herramientas para PowerShell

Ejemplo 1: Devolución de la plantilla asociada a la pila especificada.

```
Get-CFNTemplate -StackName "myStack"
```

- Para API obtener más información, consulte GetTemplatela referencia de AWS Tools for PowerShell cmdlets.

`Measure-CFNTemplateCost`

En el siguiente ejemplo de código se muestra cómo usarlo. `Measure-CFNTemplateCost`

Herramientas para PowerShell

Ejemplo 1: devuelve una calculadora mensual AWS simple URL con una cadena de consulta que describe los recursos necesarios para ejecutar la plantilla. La plantilla se obtiene del Amazon S3 especificado URL y se aplica el parámetro de personalización único. El parámetro también se puede especificar mediante «clave» y «valor» en lugar de «ParameterKey» y «ParameterValue».

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Ejemplo 2: devuelve una calculadora mensual AWS sencilla URL con una cadena de consulta que describe los recursos necesarios para ejecutar la plantilla. La plantilla se analiza a partir

del contenido suministrado y de los parámetros de personalización aplicados (en este ejemplo se supone que el contenido de la plantilla habría declarado dos parámetros, 'KeyName' y 'InstanceType'). Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de «ParameterKey» y «ParameterValue».

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Ejemplo 3: Utiliza New-Object para crear el conjunto de parámetros de la plantilla y devuelve una calculadora mensual AWS simple URL con una cadena de consulta que describe los recursos necesarios para ejecutar la plantilla. La plantilla se analiza a partir del contenido suministrado, con parámetros de personalización (en este ejemplo se supone que el contenido de la plantilla habría declarado dos parámetros, " y KeyName "). 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 )
```

- Para API obtener más información, consulte EstimateTemplateCost AWS Tools for PowerShellCmdlet Reference.

`New-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CFNStack`

Herramientas para PowerShell

Ejemplo 1: Creación de una pila nueva con el nombre especificado. La plantilla se analiza a partir del contenido suministrado con parámetros de personalización (" y PK1 'PK2' representan los nombres de los parámetros declarados en el contenido de la plantilla, 'PV1' y PV2 'representan

los valores de esos parámetros). Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de «ParameterKey» y «ParameterValue». Si la creación de la pila falla, no se revertirá.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Ejemplo 2: Creación de una pila nueva con el nombre especificado. La plantilla se analiza a partir del contenido suministrado con los parámetros de personalización ('PK1' y 'PK2' representan los nombres de los parámetros declarados en el contenido de la plantilla, PV1 " y PV2 'representan los valores de esos parámetros). Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de «ParameterKey» y «ParameterValue». Si la creación de la pila falla, se revertirá.

```
$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"
```

Ejemplo 3: Creación de una pila nueva con el nombre especificado. La plantilla se obtiene de Amazon S3 URL con parámetros de personalización ('PK1' representa el nombre de un parámetro declarado en el contenido de la plantilla, 'PV1' representa el valor del parámetro). Los parámetros de personalización también se pueden especificar mediante «clave» y «valor» en lugar de «ParameterKey» y «ParameterValue». Si la creación de la pila falla, se revertirá (igual que si se especificara: DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
            -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Ejemplo 4: Creación de una pila nueva con el nombre especificado. La plantilla se obtiene de Amazon S3 URL con parámetros de personalización ('PK1' representa el nombre de un parámetro declarado en el contenido de la plantilla, 'PV1' representa el valor del parámetro). Los parámetros de personalización también se pueden especificar mediante «clave» y «valor» en lugar de «ParameterKey» y «ParameterValue». Si la creación de la pila falla, se revertirá (igual que si se especificara: DisableRollback $false). La notificación especificada AENs recibirá los eventos publicados relacionados con la pila.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Para API obtener más información, consulte la referencia del CreateStack AWS Tools for PowerShellcmdlet.

`Remove-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CFNStack`

Herramientas para PowerShell

Ejemplo 1: Eliminación de la pila especificada.

```
Remove-CFNStack -StackName "myStack"
```

- Para API obtener más información, consulte DeleteStackla referencia de AWS Tools for PowerShell cmdlets.

`Resume-CFNUpdateRollback`

En el siguiente ejemplo de código se muestra cómo usarlo. `Resume-CFNUpdateRollback`

Herramientas para PowerShell

Ejemplo 1: Continúa la reversión de la pila nombrada, que debería estar en el estado 'UPDATE_ ROLLBACK _FAILED'. Si la reversión continua se realiza correctamente, la pila entrará en el estado 'UPDATE_ ROLLBACK _COMPLETE'.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Para API obtener más información, consulte la Referencia ContinueUpdateRollbackde AWS Tools for PowerShell cmdlets.

`Stop-CFNUpdateStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-CFNUpdateStack`

Herramientas para PowerShell

Ejemplo 1: Cancelación de una actualización en la pila especificada.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Para API obtener más información, consulte CancelUpdateStackla referencia de AWS Tools for PowerShell cmdlets.

`Test-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Test-CFNStack`

Herramientas para PowerShell

Ejemplo 1: Comprueba si la pila ha alcanzado uno de los estados UPDATE _ ROLLBACK _COMPLETE, CREATE _COMPLETE, ROLLBACK _ COMPLETE o UPDATE _COMPLETE.

```
Test-CFNStack -StackName MyStack
```

Salida:

```
False
```

Ejemplo 2: Comprueba si la pila ha alcanzado un estado de UPDATE _ COMPLETE o UPDATE _ ROLLBACK _COMPLETE.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Salida:

```
True
```

- Para API obtener más información, consulte Test- CFNStack in AWS Tools for PowerShell Cmdlet Reference.

`Test-CFNTemplate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Test-CFNTemplate`

Herramientas para PowerShell

Ejemplo 1: Validación del contenido de la plantilla especificada. El resultado detalla las capacidades, la descripción y los parámetros de la plantilla.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Ejemplo 2: valida la plantilla especificada a la que se ha accedido a través de Amazon S3URL. El resultado detalla las capacidades, la descripción y los parámetros de la plantilla.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Para API obtener más información, consulte la referencia ValidateTemplatede AWS Tools for PowerShell cmdlets.

`Update-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CFNStack`

Herramientas para PowerShell

Ejemplo 1: actualiza la pila 'myStack' con la plantilla y los parámetros de personalización especificados. 'PK1' representa el nombre de un parámetro declarado en la plantilla y 'PV1'

representa su valor. Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de ParameterKey «» y «ParameterValue».

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Ejemplo 2: actualiza la pila 'myStack' con la plantilla y los parámetros de personalización especificados. 'PK1' y 'PK2' representan los nombres de los parámetros declarados en la plantilla, PV1 " y 'PV2' representan los valores solicitados. Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de ParameterKey «» y «ParameterValue».

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Ejemplo 3: actualiza la pila 'myStack' con la plantilla y los parámetros de personalización especificados. 'PK1' representa el nombre de un parámetro declarado en la plantilla y 'PV2' representa su valor. Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de ParameterKey «» y «ParameterValue».

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Ejemplo 4: actualiza la pila 'myStack' con la plantilla especificada, obtenida de Amazon S3, y los parámetros de personalización. 'PK1' y 'PK2' representan los nombres de los parámetros declarados en la plantilla, PV1 " y 'PV2' representan los valores solicitados. Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de ParameterKey «» y «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" } )
```

Ejemplo 5: actualiza la pila 'myStack', que en este ejemplo se supone que contiene IAM recursos, con la plantilla especificada, obtenida de Amazon S3, y los parámetros de personalización. 'PK1' y 'PK2' representan los nombres de los parámetros declarados en la plantilla, PV1 " y 'PV2' representan los valores solicitados. Los parámetros de personalización también se pueden especificar mediante «Clave» y «Valor» en lugar de ParameterKey «» y «ParameterValue». Las pilas que contienen IAM recursos requieren que especifiques el parámetro -Capabilities "CAPABILITY_IAM"; de lo contrario, la actualización fallará y aparecerá un error de «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"
```

- Para API obtener más información, consulte la referencia de UpdateStack AWS Tools for PowerShellcmdlets.

`Wait-CFNStack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Wait-CFNStack`

Herramientas para PowerShell

Ejemplo 1: Comprueba si la pila ha alcanzado uno de los estados UPDATE _ ROLLBACK _COMPLETE, CREATE _COMPLETE, ROLLBACK _ COMPLETE o UPDATE _COMPLETE. Si la pila no está en uno de los estados, el comando permanece inactivo durante dos segundos antes de volver a probar el estado. Esto se repite hasta que la pila alcance uno de los estados solicitados o hasta que transcurra el tiempo de espera predeterminado de 60 segundos. Si se supera el tiempo de espera, se produce una excepción. Si la pila alcanza uno de los estados solicitados dentro del período de tiempo de espera, se devuelve a la canalización.

```
$stack = Wait-CFNStack -StackName MyStack
```

Ejemplo 2: En este ejemplo, se espera un total de 5 minutos (300 segundos) para que la pila alcance cualquiera de los estados especificados. En este ejemplo, el estado se alcanza antes del tiempo de espera y, por lo tanto, el objeto de pila se devuelve a la canalización.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Salida:

```
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
```

Ejemplo 3: Este ejemplo muestra el error generado cuando una pila no alcanza uno de los estados solicitados dentro del período de tiempo de espera (en este caso, el período predeterminado de 60 segundos).

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Salida:

```
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.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
InvalidOperationException
    + FullyQualifiedErrorId :
InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Wait- CFNStack in.

CloudFront ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CloudFront.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-CFCloudFrontOriginAccessIdentity`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-CFCloudFrontOriginAccessIdentity`.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una identidad de acceso a Amazon CloudFront Origin específica, especificada mediante el parámetro -Id. Aunque el parámetro -Id no es obligatorio, si no lo especifica, no se devolverá ningún resultado.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Salida:

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    ------------------------------------    --
-----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- Para API obtener más información, consulte la referencia GetCloudFrontOriginAccessIdentitydel AWS Tools for PowerShell cmdlet.

`Get-CFCloudFrontOriginAccessIdentityConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFCloudFrontOriginAccessIdentityConfig`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve información de configuración sobre una única identidad de acceso a Amazon CloudFront Origin, especificada mediante el parámetro -Id. Se producen errores si no se especifica ningún parámetro -Id.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Salida:

```
    CallerReference                                          Comment
    ---------------                                          -------
    mycallerreference: 2/1/2011 1:16:32 PM                   Caller reference:
2/1/2011 1:16:32 PM
```

- Para API obtener más información, consulte la referencia GetCloudFrontOriginAccessIdentityConfigdel AWS Tools for PowerShell cmdlet.

`Get-CFCloudFrontOriginAccessIdentityList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFCloudFrontOriginAccessIdentityList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una lista de identidades de acceso a Amazon CloudFront Origin. Como el MaxItem parámetro - especifica un valor de 2, los resultados incluyen dos identidades.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Salida:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Para API obtener más información, consulte ListCloudFrontOriginAccessIdentities AWS Tools for PowerShellCmdlet Reference.

`Get-CFDistribution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFDistribution`

Herramientas para PowerShell

Ejemplo 1: Recupera la información de una distribución específica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Para API obtener más información, consulte GetDistributionla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFDistributionConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFDistributionConfig`

Herramientas para PowerShell

Ejemplo 1: Recupera la configuración de una distribución específica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Para API obtener más información, consulte GetDistributionConfigla referencia de AWS Tools for PowerShell cmdlets.

`Get-CFDistributionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFDistributionList`

Herramientas para PowerShell

Ejemplo 1: Devuelve distribuciones.

```
Get-CFDistributionList
```

- Para API obtener más información, consulte ListDistributionsla referencia de AWS Tools for PowerShell cmdlets.

`New-CFDistribution`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CFDistribution`

Herramientas para PowerShell

Ejemplo 1: Crea una CloudFront distribución básica, configurada con registro y almacenamiento en caché.

```
$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
```

- Para API obtener más información, consulte la referencia CreateDistributionde AWS Tools for PowerShell cmdlets.

`New-CFInvalidation`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CFInvalidation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva invalidación en una distribución con un identificador deEXAMPLENSTXAXE. CallerReference Es un identificador único elegido por el usuario; en este caso, se utiliza una marca de tiempo que representa el 15 de mayo de 2019 a las 9:00 a. m. La variable $Paths almacena tres rutas a archivos multimedia y de imagen que el usuario no desea que formen parte de la memoria caché de la distribución. El valor del parámetro -Paths_Quantity es el número total de rutas especificadas en el parámetro -Paths_Item.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

Salida:

```
Invalidation                        Location

------------                        --------
```

```
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Para API obtener más información, consulte CreateInvalidation AWS Tools for PowerShellCmdlet Reference.

`New-CFSignedCookie`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CFSignedCookie`

Herramientas para PowerShell

Ejemplo 1: crea una cookie firmada para el recurso especificado mediante una política predefinida. La cookie tendrá una validez de un año.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Salida:

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

Ejemplo 2: crea una cookie firmada para los recursos especificados mediante una política personalizada. La cookie será válida en 24 horas y caducará una semana después.

```
$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
```

Salida:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

Ejemplo 3: crea una cookie firmada para los recursos especificados mediante una política personalizada. La cookie será válida en 24 horas y caducará una semana después. El acceso a los recursos está restringido al rango de IP especificado.

```
$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
```

Salida:

```
Policy
                                                        ------

                                  [CloudFront-Policy, eyJTd...wIjo...
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet New- CFSignedCookie in.

`New-CFSignedUrl`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CFSignedUrl`

Ejemplo 1: crea una URL firmada para el recurso especificado mediante una política predefinida. La URL será válida durante una hora. Se envía a la canalización un objeto System.Uri que contiene la URL firmada.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Ejemplo 2: crea una URL firmada para el recurso especificado mediante una política personalizada. La URL será válida a partir de 24 horas y caducará una semana después.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

Ejemplo 3: crea una URL firmada para el recurso especificado mediante una política personalizada. La URL será válida a partir de 24 horas y caducará una semana después. El acceso al recurso está restringido al rango de IP especificado.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet New- CFSignedUrl in.

CloudTrail ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CloudTrail.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Find-CTEvent`

En el siguiente ejemplo de código se muestra cómo usarlo`Find-CTEvent`.

Herramientas para PowerShell

Ejemplo 1: Devuelve todos los eventos que se han producido en los últimos siete días. De forma predeterminada, el cmdlet realiza automáticamente varias llamadas para entregar todos los eventos y se cierra cuando el servicio indica que no hay más datos disponibles.

```
Find-CTEvent
```

Ejemplo 2: devuelve todos los eventos que se han producido en los últimos siete días y especifica una región que no es la predeterminada del shell actual.

```
Find-CTEvent -Region eu-central-1
```

Ejemplo 3: Devuelve todos los eventos asociados RunInstances API a la llamada.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Ejemplo 4: Devuelve los primeros 5 eventos disponibles. El token que se utilizará para recuperar más eventos se adjunta al **`$AWSHistory.LastServiceResponse`** miembro como una propiedad de nota denominada NextToken ".

```
Find-CTEvent -MaxResult 5
```

Ejemplo 5: devuelve los 10 eventos siguientes utilizando el símbolo de «página siguiente» de una llamada anterior para indicar desde dónde empezar a devolver los eventos de la secuencia.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

Ejemplo 6: En este ejemplo se muestra cómo recorrer los eventos disponibles mediante la paginación manual, obteniendo un máximo de 5 eventos por llamada.

```
$nextToken = $null
do
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet LookupEventsReference.

`Get-CTTrail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CTTrail`

Herramientas para PowerShell

Ejemplo 1: Devuelve la configuración de todos los senderos asociados a la región actual de tu cuenta.

```
Get-CTTrail
```

Ejemplo 2: Devuelve la configuración de las rutas especificadas.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Ejemplo 3: devuelve la configuración de las rutas especificadas que se crearon en una región distinta de la predeterminada de shell actual (en este caso, la región de Fráncfort (eu-central-1)).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Para API obtener más información, consulte Cmdlet DescribeTrailsReference AWS Tools for PowerShell .

`Get-CTTrailStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CTTrailStatus`

Herramientas para PowerShell

Ejemplo 1: Devuelve la información de estado de la ruta con el nombre 'myExampleTrail'. Los datos devueltos incluyen información sobre los errores de entrega, los errores de Amazon SNS y Amazon S3 y los tiempos de inicio y finalización del registro de rutas. En este ejemplo, se supone que la ruta se creó en la misma región que el shell predeterminado actual.

```
Get-CTTrailStatus -Name myExampleTrail
```

Ejemplo 2: devuelve la información de estado de una ruta que se creó en una región distinta de la predeterminada de shell actual (en este caso, la región de Fráncfort (eu-central-1)).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Para API obtener más información, consulte Cmdlet GetTrailStatusReference AWS Tools for PowerShell .

`New-CTTrail`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CTTrail`

Herramientas para PowerShell

Ejemplo 1: crea una ruta que utilizará el depósito «mycloudtrailbucket» para almacenar los archivos de registro.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Ejemplo 2: crea un rastro que utilizará el depósito «mycloudtrailbucket» para almacenar los archivos de registro. Los objetos S3 que representan los registros tendrán un key prefijo común de «mylogs». Cuando se envíen nuevos registros al depósito, se enviará una notificación al tema «mlog-deliverytopic». SNS En este ejemplo, se utiliza el splatting para proporcionar los valores de los parámetros al cmdlet.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Para obtener API más información, consulte CreateTrailla referencia del cmdlet.AWS Tools for PowerShell

`Remove-CTTrail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CTTrail`

Herramientas para PowerShell

Ejemplo 1: Elimina la ruta especificada. Se le pedirá que lo confirme antes de ejecutar el comando. Para suprimir la confirmación, añada el parámetro -Force switch.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Para API obtener más información, consulte DeleteTrail AWS Tools for PowerShellCmdlet Reference.

`Start-CTLogging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-CTLogging`

Herramientas para PowerShell

Ejemplo 1: Inicia la grabación de las AWS API llamadas y la entrega del archivo de registro para la ruta denominada 'myExampleTrail'. En este ejemplo, se supone que el sendero se creó en la misma región que el shell por defecto actual.

```
Start-CTLogging -Name myExampleTrail
```

Ejemplo 2: Inicia la grabación de las AWS API llamadas y la entrega de los archivos de registro de una ruta que se creó en una región distinta de la predeterminada de shell actual (en este caso, la región de Fráncfort (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para API obtener más información, consulte Cmdlet StartLoggingReference AWS Tools for PowerShell .

`Stop-CTLogging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-CTLogging`

Herramientas para PowerShell

Ejemplo 1: Suspende la grabación de las AWS API llamadas y la entrega de archivos de registro para la ruta denominada 'myExampleTrail'. En este ejemplo, se supone que la ruta se creó en la misma región que la configuración predeterminada del shell actual.

```
Stop-CTLogging -Name myExampleTrail
```

Ejemplo 2: suspende la grabación de AWS API llamadas y la entrega de archivos de registro de una ruta que se creó en una región distinta de la predeterminada de shell actual (en este caso, la región de Fráncfort (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obtener API más información, consulte StopLoggingCmdlet Reference.AWS Tools for PowerShell

`Update-CTTrail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CTTrail`

Herramientas para PowerShell

Ejemplo 1: actualiza el registro especificado para que se registren los eventos de servicio globales (como los deIAM) y cambia el prefijo clave común de los archivos de registro en el futuro para que sea «globallogs».

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Ejemplo 2: actualiza el registro especificado para que las notificaciones sobre las nuevas entregas de registros se envíen al tema especificado. SNS

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Ejemplo 3: actualiza la ruta especificada para que los registros se entreguen a un depósito diferente.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Para API obtener más información, consulte UpdateTrail AWS Tools for PowerShellCmdlet Reference.

CloudWatch ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CloudWatch.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-CWDashboard`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-CWDashboard`.

Herramientas para PowerShell

Ejemplo 1: devuelve el arn al cuerpo del panel especificado.

```
Get-CWDashboard -DashboardName Dashboard1
```

Salida:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Para API obtener más información, consulte GetDashboardla referencia de AWS Tools for PowerShell cmdlets.

`Get-CWDashboardList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CWDashboardList`

Herramientas para PowerShell

Ejemplo 1: devuelve el conjunto de paneles de su cuenta.

```
Get-CWDashboardList
```

Salida:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Ejemplo 2: devuelve el conjunto de paneles de su cuenta cuyos nombres comienzan con el prefijo “dev”.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Para API obtener más información, consulte ListDashboardsla referencia de AWS Tools for PowerShell cmdlets.

`Remove-CWDashboard`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CWDashboard`

Herramientas para PowerShell

Ejemplo 1: elimina el panel especificado y lo confirma antes de continuar. Para omitir la confirmación, agregue el modificador -Force al comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Para API obtener más información, consulte DeleteDashboardsla referencia de AWS Tools for PowerShell cmdlets.

`Write-CWDashboard`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-CWDashboard`

Herramientas para PowerShell

Ejemplo 1: crea o actualiza el panel denominado “Dashboard1” para incluir dos widgets de métricas uno al lado del otro.

```
$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
```

Ejemplo 2: crea o actualiza el panel y canaliza el contenido que describe el panel al cmdlet.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Para API obtener más información, consulte PutDashboardla referencia de AWS Tools for PowerShell cmdlets.

`Write-CWMetricData`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-CWMetricData`

Herramientas para PowerShell

Ejemplo 1: crea un MetricDatum objeto nuevo y lo escribe en Amazon Web Services CloudWatch Metrics.

```
### 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
```

- Para API obtener más información, consulte PutMetricDatala referencia de AWS Tools for PowerShell cmdlets.

CodeCommit ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CodeCommit.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-CCBranch`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-CCBranch`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre la rama especificada para el repositorio especificado.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Salida:

```
BranchName                          CommitId
----------                          --------
MyNewBranch                         7763222d...561fc9c9
```

- Para API obtener más información, consulte GetBranch AWS Tools for PowerShellCmdlet Reference.

`Get-CCBranchList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CCBranchList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene una lista de los nombres de las ramas del repositorio especificado.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Salida:

```
master
MyNewBranch
```

- Para API obtener más información, consulte ListBranches AWS Tools for PowerShellCmdlet Reference.

`Get-CCRepository`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CCRepository`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información del repositorio especificado.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Salida:

```
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
```

- Para API obtener más información, consulte GetRepository AWS Tools for PowerShellCmdlet Reference.

`Get-CCRepositoryBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CCRepositoryBatch`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo confirma cuáles de los repositorios especificados se encuentran y cuáles no.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Salida:

```
Repositories                          RepositoriesNotFound
------------                          --------------------
{MyDemoRepo, MyNewRepo}               {AMissingRepo}
```

- Para API obtener más información, consulte la referencia BatchGetRepositoriesde AWS Tools for PowerShell cmdlets.

`Get-CCRepositoryList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CCRepositoryList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todos los repositorios en orden ascendente por nombre de repositorio.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Salida:

```
RepositoryId                          RepositoryName
------------                          --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE  MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE  MyNewRepo
```

- Para API obtener más información, consulte la referencia de ListRepositories AWS Tools for PowerShellcmdlets.

`New-CCBranch`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CCBranch`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva rama con el nombre especificado para el repositorio especificado y el ID de confirmación especificado.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Para API obtener más información, consulte CreateBranch AWS Tools for PowerShellCmdlet Reference.

`New-CCRepository`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CCRepository`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un repositorio nuevo con el nombre y la descripción especificados.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateRepository AWS Tools for PowerShellCmdlet Reference.

`Remove-CCRepository`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CCRepository`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se borra por la fuerza el repositorio especificado. El comando solicitará una confirmación antes de continuar. Añada el parámetro -Force para eliminar el repositorio sin necesidad de solicitarlo.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Salida:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Para API obtener más información, consulte DeleteRepository AWS Tools for PowerShellCmdlet Reference.

`Update-CCDefaultBranch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CCDefaultBranch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cambia la rama predeterminada del repositorio especificado por la rama especificada.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Para API obtener más información, consulte UpdateDefaultBranch AWS Tools for PowerShellCmdlet Reference.

`Update-CCRepositoryDescription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CCRepositoryDescription`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cambia la descripción del repositorio especificado.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Para API obtener más información, consulte UpdateRepositoryDescription AWS Tools for PowerShellCmdlet Reference.

`Update-CCRepositoryName`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CCRepositoryName`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el nombre del repositorio especificado.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Para API obtener más información, consulte UpdateRepositoryName AWS Tools for PowerShellCmdlet Reference.

CodeDeploy ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CodeDeploy.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-CDOnPremiseInstanceTag`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-CDOnPremiseInstanceTag`.

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega una etiqueta de instancia local con la clave y el valor especificados para la instancia local especificada.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- Para API obtener más información, consulta la referencia del AddTagsToOnPremisesInstances AWS Tools for PowerShellcmdlet.

`Get-CDApplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDApplication`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre la aplicación especificada.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Salida:

```
ApplicationId                         ApplicationName             CreateTime
         LinkedToGitHub
-------------                         ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE  CodeDeployDemoApplication   7/20/2015
 9:49:48 PM    False
```

- Para API obtener más información, consulte GetApplication AWS Tools for PowerShellCmdlet Reference.

`Get-CDApplicationBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDApplicationBatch`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre las aplicaciones especificadas.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Salida:

```
ApplicationId                           ApplicationName             CreateTime
         LinkedToGitHub
-------------                           ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Para API obtener más información, consulte BatchGetApplications AWS Tools for PowerShellCmdlet Reference.

`Get-CDApplicationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDApplicationList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene una lista de las aplicaciones disponibles.

```
Get-CDApplicationList
```

Salida:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Para API obtener más información, consulte ListApplications AWS Tools for PowerShellCmdlet Reference.

`Get-CDApplicationRevision`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDApplicationRevision`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre la revisión de la aplicación especificada.

```
$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)
```

Salida:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Para API obtener más información, consulte GetApplicationRevision AWS Tools for PowerShellCmdlet Reference.

`Get-CDApplicationRevisionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDApplicationRevisionList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información sobre las revisiones disponibles para la aplicación especificada.

```
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)
>>   }
>> }
```

```
>>
```

Salida:

```
Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Para API obtener más información, consulte ListApplicationRevisions AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeployment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeployment`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información resumida sobre la implementación especificada.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se obtiene información sobre el estado de las instancias que participan en la implementación especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Salida:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Ejemplo 3: en este ejemplo se obtiene información sobre la revisión de la aplicación para la implementación especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Salida:

```
Bucket     : MyBucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Para API obtener más información, consulte GetDeployment AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeploymentBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentBatch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre las implementaciones especificadas.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Salida:

```
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
```

- Para API obtener más información, consulte BatchGetDeployments AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeploymentConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información resumida sobre la configuración de despliegue especificada.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Salida:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Ejemplo 2: En este ejemplo se obtiene información sobre la definición de la configuración de despliegue especificada.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Salida:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- Para API obtener más información, consulte GetDeploymentConfig AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeploymentConfigList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentConfigList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene una lista de las configuraciones de despliegue disponibles.

```
Get-CDDeploymentConfigList
```

Salida:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
```

```
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Para API obtener más información, consulte ListDeploymentConfigsla referencia de AWS Tools for PowerShell cmdlets.

`Get-CDDeploymentGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre el grupo de despliegue especificado.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Salida:

```
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-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Para API obtener más información, consulte GetDeploymentGroupla referencia de AWS Tools for PowerShell cmdlets.

`Get-CDDeploymentGroupList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentGroupList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene una lista de grupos de despliegue para la aplicación especificada.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Salida:

```
ApplicationName             DeploymentGroups
 NextToken
---------------             ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Para API obtener más información, consulte ListDeploymentGroupsla referencia de AWS Tools for PowerShell cmdlets.

`Get-CDDeploymentInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre la instancia especificada para la implementación especificada.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Salida:

```
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
```

- Para API obtener más información, consulte GetDeploymentInstance AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeploymentInstanceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentInstanceList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene una lista de instancias IDs para la implementación especificada.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Salida:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Para API obtener más información, consulte ListDeploymentInstances AWS Tools for PowerShellCmdlet Reference.

`Get-CDDeploymentList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDDeploymentList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene una lista de despliegues IDs para la aplicación y el grupo de despliegues especificados.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

Salida:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Para API obtener más información, consulte ListDeploymentsla referencia de AWS Tools for PowerShell cmdlets.

`Get-CDOnPremiseInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDOnPremiseInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre la instancia local especificada.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Salida:

```
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}
```

- Para API obtener más información, consulte GetOnPremisesInstance AWS Tools for PowerShellCmdlet Reference.

`Get-CDOnPremiseInstanceBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDOnPremiseInstanceBatch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre las instancias locales especificadas.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Salida:

```
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}
```

- Para API obtener más información, consulte la referencia BatchGetOnPremisesInstancesde AWS Tools for PowerShell cmdlets.

`Get-CDOnPremiseInstanceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CDOnPremiseInstanceList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene una lista de los nombres de las instancias locales disponibles.

```
Get-CDOnPremiseInstanceList
```

Salida:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Para API obtener más información, consulte la referencia ListOnPremisesInstancesde AWS Tools for PowerShell cmdlets.

`New-CDApplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CDApplication`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva aplicación con el nombre especificado.

```
New-CDApplication -ApplicationName MyNewApplication
```

Salida:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Para API obtener más información, consulte CreateApplication AWS Tools for PowerShellCmdlet Reference.

`New-CDDeployment`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CDDeployment`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva implementación para la aplicación y el grupo de implementación especificados con la configuración de implementación y la revisión de la aplicación especificadas.

```
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
```

Salida:

```
d-ZHROG7UEX
```

Ejemplo 2: En este ejemplo, se muestra cómo especificar grupos de EC2 etiquetas de instancia con las que se debe identificar una instancia para poder incluirla en el entorno de reemplazo de una implementación azul/verde.

```
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=
```

Salida:

```
d-ZHROG7UEX
```

- Para API obtener más información, consulta la referencia del CreateDeployment AWS Tools for PowerShellcmdlet.

`New-CDDeploymentConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CDDeploymentConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una nueva configuración de despliegue con el nombre y el comportamiento especificados.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Salida:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Para API obtener más información, consulte CreateDeploymentConfigla referencia de AWS Tools for PowerShell cmdlets.

`New-CDDeploymentGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CDDeploymentGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un grupo de despliegue con el nombre especificado, el grupo de Auto Scaling, la configuración de despliegue, la etiqueta y el rol de servicio, para la aplicación especificada.

```
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
```

Salida:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Ejemplo 2: En este ejemplo, se muestra cómo especificar grupos de EC2 etiquetas de instancia con las que se debe identificar una instancia para poder incluirla en el entorno de reemplazo de una implementación azul/verde.

```
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=
```

Salida:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Para API obtener más información, consulta la referencia del CreateDeploymentGroup AWS Tools for PowerShellcmdlet.

`Register-CDApplicationRevision`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-CDApplicationRevision`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se registra una revisión de aplicación en la ubicación de Amazon S3 especificada para la aplicación especificada.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Para API obtener más información, consulte RegisterApplicationRevisionla referencia del AWS Tools for PowerShell cmdlet.

`Register-CDOnPremiseInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-CDOnPremiseInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra una instancia local con el nombre y el IAM usuario especificados.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Para API obtener más información, consulte la referencia RegisterOnPremisesInstancede AWS Tools for PowerShell cmdlets.

`Remove-CDApplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CDApplication`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la aplicación con el nombre especificado. El comando solicitará una confirmación antes de continuar. Añada el parámetro -Force para eliminar la aplicación sin necesidad de solicitarla.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Para API obtener más información, consulte DeleteApplication AWS Tools for PowerShellCmdlet Reference.

`Remove-CDDeploymentConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CDDeploymentConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la configuración de despliegue con el nombre especificado. El comando solicitará una confirmación antes de continuar. Añada el parámetro -Force para eliminar la configuración de despliegue sin una solicitud.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Para API obtener más información, consulte DeleteDeploymentConfig AWS Tools for PowerShellCmdlet Reference.

`Remove-CDDeploymentGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CDDeploymentGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el grupo de despliegue con el nombre especificado para la aplicación especificada. El comando solicitará una confirmación antes de continuar. Añada el parámetro -Force para eliminar el grupo de despliegue sin que se le pida nada.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Para API obtener más información, consulte DeleteDeploymentGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-CDOnPremiseInstanceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CDOnPremiseInstanceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta especificada para la instancia local con el nombre especificado. El comando solicitará una confirmación antes de continuar. Agrega el parámetro -Force para eliminar la etiqueta sin que se te pida nada.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para API obtener más información, consulte RemoveTagsFromOnPremisesInstances AWS Tools for PowerShellCmdlet Reference.

`Stop-CDDeployment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-CDDeployment`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se intenta detener la implementación con el ID de implementación especificado.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Salida:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Para API obtener más información, consulte StopDeploymentla referencia de AWS Tools for PowerShell cmdlets.

`Unregister-CDOnPremiseInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-CDOnPremiseInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se anula el registro de la instancia local con el nombre especificado.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Para API obtener más información, consulte la referencia de DeregisterOnPremisesInstancecmdlets AWS Tools for PowerShell .

`Update-CDApplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CDApplication`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cambia el nombre de la aplicación especificada.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Para API obtener más información, consulte UpdateApplication AWS Tools for PowerShellCmdlet Reference.

`Update-CDDeploymentGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CDDeploymentGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cambia el nombre del grupo de despliegue especificado para la aplicación especificada.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Ejemplo 2: en este ejemplo se muestra cómo especificar grupos de etiquetas de EC2 instancia con las que se debe identificar una instancia para poder incluirla en el entorno de reemplazo de una implementación azul/verde.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Para API obtener más información, consulta la referencia del UpdateDeploymentGroup AWS Tools for PowerShellcmdlet.

CodePipeline ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with CodePipeline.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Confirm-CPJob`

En el siguiente ejemplo de código se muestra cómo usarlo`Confirm-CPJob`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene el estado del trabajo especificado.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Salida:

```
Value
-----
InProgress
```

- Para API obtener más información, consulte AcknowledgeJob AWS Tools for PowerShellCmdlet Reference.

`Disable-CPStageTransition`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-CPStageTransition`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo deshabilita la transición entrante para la etapa especificada de la canalización especificada.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DisableStageTransitionReference.

`Enable-CPStageTransition`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-CPStageTransition`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita la transición entrante para la etapa especificada de la canalización especificada.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Para API obtener más información, consulte EnableStageTransition AWS Tools for PowerShellCmdlet Reference.

`Get-CPActionType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPActionType`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene información sobre todas las acciones disponibles para el propietario especificado.

```
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)
}
```

Salida:

```
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}
```

- Para API obtener más información, consulte ListActionTypes AWS Tools for PowerShellCmdlet Reference.

`Get-CPActionableJobList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPActionableJobList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo obtiene información sobre todos los trabajos procesables para la categoría de acción, el propietario, el proveedor, la versión y los parámetros de consulta especificados.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Salida:

```
AccountId       Data                                Id
         Nonce
---------       ----                                --
         -----
```

```
80398EXAMPLE    Amazon.CodePipeline.Model.JobData    0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Para API obtener más información, consulte PollForJobs AWS Tools for PowerShellCmdlet Reference.

`Get-CPJobDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPJobDetail`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información general sobre el trabajo especificado.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Salida:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Ejemplo 2: En este ejemplo se obtiene información detallada sobre el trabajo especificado.

```
$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)
```

Salida:

```
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
```

- Para API obtener más información, consulte GetJobDetails AWS Tools for PowerShellCmdlet Reference.

`Get-CPPipeline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPPipeline`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información general sobre la canalización especificada.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Salida:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Ejemplo 2: En este ejemplo se obtiene información detallada sobre la canalización especificada.

```
$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)
  }
}
```

Salida:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = MyBucketName
  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
```

- Para API obtener más información, consulte GetPipeline AWS Tools for PowerShellCmdlet Reference.

`Get-CPPipelineList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPPipelineList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene una lista de las canalizaciones disponibles.

```
Get-CPPipelineList
```

Salida:

```
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
```

- Para API obtener más información, consulte ListPipelines AWS Tools for PowerShellCmdlet Reference.

`Get-CPPipelineState`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CPPipelineState`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene información general sobre las etapas de la canalización especificada.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se obtiene información detallada sobre el estado de la canalización especificada.

```
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)
  }
}
```

Salida:

```
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 =
```

- Para API obtener más información, consulte GetPipelineState AWS Tools for PowerShellCmdlet Reference.

`New-CPCustomActionType`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CPCustomActionType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva acción personalizada con las propiedades especificadas.

```
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
```

Salida:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Para API obtener más información, consulte CreateCustomActionType AWS Tools for PowerShellCmdlet Reference.

`New-CPPipeline`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CPPipeline`

Ejemplo 1: Este ejemplo crea una nueva canalización con la configuración especificada.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.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 = "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
```

Salida:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Para API obtener más información, consulte CreatePipeline AWS Tools for PowerShellCmdlet Reference.

`Remove-CPCustomActionType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CPCustomActionType`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la acción personalizada especificada. El comando solicitará una confirmación antes de continuar. Añada el parámetro -Force para eliminar la acción personalizada sin una solicitud.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- Para API obtener más información, consulte DeleteCustomActionType AWS Tools for PowerShellCmdlet Reference.

`Remove-CPPipeline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CPPipeline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la canalización especificada. El comando solicitará una confirmación antes de continuar. Agregue el parámetro -Force para eliminar la canalización sin una solicitud.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Para API obtener más información, consulte DeletePipeline AWS Tools for PowerShellCmdlet Reference.

`Start-CPPipelineExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-CPPipelineExecution`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se inicia la ejecución de la canalización especificada.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Para API obtener más información, consulte StartPipelineExecution AWS Tools for PowerShellCmdlet Reference.

`Update-CPPipeline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CPPipeline`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo actualiza la canalización existente especificada con la configuración especificada.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.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
```

Salida:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Para API obtener más información, consulte UpdatePipeline AWS Tools for PowerShellCmdlet Reference.

Ejemplos de Amazon Cognito Identity que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante Amazon Cognito Identity. AWS Tools for PowerShell

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-CGIIdentityPool`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-CGIIdentityPool`.

Herramientas para PowerShell

Ejemplo 1: recupera información sobre un grupo de identidades específico por su identificador.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeIdentityPool AWS Tools for PowerShellCmdlet Reference.

`Get-CGIIdentityPoolList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CGIIdentityPoolList`

Herramientas para PowerShell

Ejemplo 1: recupera una lista de grupos de identidades existentes.

```
Get-CGIIdentityPoolList
```

Salida:

```
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
```

- Para API obtener más información, consulte ListIdentityPools AWS Tools for PowerShellCmdlet Reference.

`Get-CGIIdentityPoolRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CGIIdentityPoolRole`

Herramientas para PowerShell

Ejemplo 1: Obtiene la información sobre las funciones de un grupo de identidades específico.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Salida:

```
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
```

- Para API obtener más información, consulte GetIdentityPoolRoleslа referencia de AWS Tools for PowerShell cmdlets.

`New-CGIIdentityPool`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-CGIIdentityPool`

Herramientas para PowerShell

Ejemplo 1: Crea un nuevo grupo de identidades que permite identidades no autenticadas.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia de CreateIdentityPool AWS Tools for PowerShellcmdlets.

`Remove-CGIIdentityPool`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CGIIdentityPool`

Herramientas para PowerShell

Ejemplo 1: Elimina un grupo de identidades específico.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Para API obtener más información, consulte DeleteIdentityPool AWS Tools for PowerShellCmdlet Reference.

`Set-CGIIdentityPoolRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-CGIIdentityPoolRole`

Herramientas para PowerShell

Ejemplo 1: Configura el grupo de identidades específico para que tenga un rol no IAM autenticado.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Para API obtener más información, consulte la referencia de SetIdentityPoolRolescmdlets AWS Tools for PowerShell .

`Update-CGIIdentityPool`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-CGIIdentityPool`

Herramientas para PowerShell

Ejemplo 1: actualiza algunas de las propiedades del grupo de identidades, en este caso el nombre del grupo de identidades.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Salida:

```
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
```

- Para API obtener más información, consulte UpdateIdentityPool AWS Tools for PowerShellCmdlet Reference.

AWS Config ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Config.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-CFGResourceTag`

En el siguiente ejemplo de código se muestra cómo usarloAdd-CFGResourceTag.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia la etiqueta especificada al recursoARN, que en este caso es config-rule-16iyn0.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- Para TagResourceobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

`Get-CFGAggregateComplianceByConfigRuleList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateComplianceByConfigRuleList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se obtienen los detalles del filtrado ConfigurationAggregator «kaju» de la regla de configuración dada y se amplía o devuelve la «conformidad» de la regla.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Salida:

```
ComplianceContributorCount                          ComplianceType
--------------------------                          --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Ejemplo 2: En este ejemplo, se obtienen los detalles de lo indicado ConfigurationAggregator, se filtran para la cuenta en cuestión para todas las regiones incluidas en el agregador y, además, se devuelve el cumplimiento de todas las reglas.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Salida:

```
ConfigRuleName                     Compliance
--------------                     ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip          NON_COMPLIANT
desired-instance-type              NON_COMPLIANT
```

- Para obtener API más información, consulte DescribeAggregateComplianceByConfigRulesCmdlet Reference.AWS Tools for PowerShell

`Get-CFGAggregateComplianceDetailsByConfigRule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateComplianceDetailsByConfigRule`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se devuelven los resultados de la evaluación y se selecciona el resultado con el identificador de recurso y el tipo de recurso para la regla de AWS Config "desired-instance-type, que están en el estado 'COMPLIANT' para la cuenta, el agregador, la región y la regla de configuración determinados

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

Salida:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- Para obtener API más información, consulte Cmdlet Reference. GetAggregateComplianceDetailsByConfigRuleAWS Tools for PowerShell

`Get-CFGAggregateConfigRuleComplianceSummary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateConfigRuleComplianceSummary`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el número de reglas no conformes para el agregador dado.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Salida:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet GetAggregateConfigRuleComplianceSummaryReference.

`Get-CFGAggregateDiscoveredResourceCount`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateDiscoveredResourceCount`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el recuento de recursos del agregador dado filtrado para la región us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Salida:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- -----------------------
           {}                              455
```

Ejemplo 2: Este ejemplo devuelve el recuento de recursos agrupado por RESOURCE _ TYPE para la región filtrada del agregador dado.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

Salida:

```
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
```

- Para API obtener más información, consulte GetAggregateDiscoveredResourceCounts AWS Tools for PowerShellCmdlet Reference.

`Get-CFGAggregateDiscoveredResourceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateDiscoveredResourceList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve los identificadores de recursos para el tipo de recurso dado agregados en el agregador «Irlanda». Para ver la lista de tipos de recursos, consulte v3/apidocs/index.html? https://docs.aws.amazon.com/sdkfornet/ page=ConfigService/TConfigServiceResourceTypeConfigService.html&tocid=Amazon_ _. ResourceType

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

Salida:

```
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
```

Ejemplo 2: Este ejemplo devuelve el tipo de recurso **`AwsEC2SecurityGroup`** denominado «default» para el agregador dado filtrado con la región us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del ListAggregateDiscoveredResourcescmdlet AWS Tools for PowerShell .

`Get-CFGAggregateResourceConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateResourceConfig`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el elemento de configuración del recurso dado agregado y expande la configuración.

```
(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
```

Salida:

```
{"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"}
```

- Para API obtener más información, consulte GetAggregateResourceconfig-service en la referencia del AWS Tools for PowerShell cmdlet.

`Get-CFGAggregateResourceConfigBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregateResourceConfigBatch`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo busca el elemento de configuración actual del recurso (identificado) presente en el agregador dado.

```
$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
```

Salida:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- ------------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Para API obtener más información, consulte BatchGetAggregateResourceconfig-service en AWS Tools for PowerShell la referencia del cmdlet.

`Get-CFGAggregationAuthorizationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGAggregationAuthorizationList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera las autorizaciones concedidas a los agregadores.

```
Get-CFGAggregationAuthorizationList
```

Salida:

```
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
```

- Para API obtener más información, consulte Cmdlet DescribeAggregationAuthorizationsReference AWS Tools for PowerShell .

`Get-CFGComplianceByConfigRule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceByConfigRule`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera los detalles de conformidad de la regla ebs-optimized-instance, para los que no hay resultados de evaluación actuales de la regla, por lo que devuelve _ INSUFFICIENT DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Salida:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Ejemplo 2: Este ejemplo devuelve el número de recursos no conformes para la regla ALB _ HTTP HTTPS _TO_ _ _. REDIRECTION CHECK

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Salida:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Para API obtener más información, consulte Cmdlet DescribeComplianceByConfigRuleReference AWS Tools for PowerShell .

`Get-CFGComplianceByResource`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceByResource`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se comprueba el tipo de **`AWS::SSM::ManagedInstanceInventory`** recurso para ver si el tipo de conformidad es COMPLIANT «».

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Salida:

```
Compliance                            ResourceId          ResourceType
----------                            ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Para API obtener más información, consulte DescribeComplianceByResource AWS Tools for PowerShellCmdlet Reference.

`Get-CFGComplianceDetailsByConfigRule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceDetailsByConfigRule`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los resultados de la evaluación de la regla access-keys-rotated y se devuelve el resultado agrupado por tipo de conformidad

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Salida:

```
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...
```

Ejemplo 2: En este ejemplo se consultan los detalles de conformidad de la regla para los recursos. access-keys-rotated COMPLIANT

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Salida:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Para API obtener más información, consulte GetComplianceDetailsByConfigRule AWS Tools for PowerShellCmdlet Reference.

`Get-CFGComplianceDetailsByResource`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceDetailsByResource`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los resultados de la evaluación para el recurso dado.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

Salida:

```
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               :
```

- Para API obtener más información, consulte la referencia del GetComplianceDetailsByResource AWS Tools for PowerShellcmdlet.

`Get-CFGComplianceSummaryByConfigRule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceSummaryByConfigRule`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el número de reglas de Config que no son compatibles.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Salida:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Para API obtener más información, consulte la referencia GetComplianceSummaryByConfigRulede AWS Tools for PowerShell cmdlets.

`Get-CFGComplianceSummaryByResourceType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGComplianceSummaryByResourceType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se devuelve el número de recursos que son compatibles o no y se convierte el resultado en 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
  }
}
```

- Para API obtener más información, consulte la referencia GetComplianceSummaryByResourceTypede AWS Tools for PowerShell cmdlets.

`Get-CFGConfigRule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigRule`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las reglas de configuración de la cuenta, con las propiedades seleccionadas.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Salida:

```
ConfigRuleName                                  ConfigRuleId       ConfigRuleArn
                                                    ConfigRuleState
--------------                                  ------------       -------------
                                                    ---------------
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
```

- Para API obtener más información, consulte DescribeConfigRules AWS Tools for PowerShellCmdlet Reference.

`Get-CFGConfigRuleEvaluationStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigRuleEvaluationStatus`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve la información de estado de las reglas de configuración dadas.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeConfigRuleEvaluationStatus AWS Tools for PowerShellCmdlet Reference.

`Get-CFGConfigurationAggregatorList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigurationAggregatorList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran todos los agregadores de la región/cuenta.

```
Get-CFGConfigurationAggregatorList
```

Salida:

```
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
```

- Para API obtener más información, consulte Cmdlet DescribeConfigurationAggregatorsReference AWS Tools for PowerShell .

`Get-CFGConfigurationAggregatorSourcesStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigurationAggregatorSourcesStatus`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra los campos solicitados para las fuentes del agregador en cuestión.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeConfigurationAggregatorSourcesStatus AWS Tools for PowerShellCmdlet Reference.

`Get-CFGConfigurationRecorder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigurationRecorder`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve los detalles de los grabadores de configuración.

```
Get-CFGConfigurationRecorder | Format-List
```

Salida:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Para API obtener más información, consulte DescribeConfigurationRecorders AWS Tools for PowerShellCmdlet Reference.

`Get-CFGConfigurationRecorderStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConfigurationRecorderStatus`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el estado de los registradores de configuración.

```
Get-CFGConfigurationRecorderStatus
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeConfigurationRecorderStatusde AWS Tools for PowerShell cmdlets.

`Get-CFGConformancePack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGConformancePack`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todos los paquetes de conformidad.

```
Get-CFGConformancePack
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeConformancePacksde AWS Tools for PowerShell cmdlets.

`Get-CFGDeliveryChannel`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGDeliveryChannel`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera el canal de entrega de la región y se muestran los detalles.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Salida:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Para API obtener más información, consulte DescribeDeliveryChannels AWS Tools for PowerShellCmdlet Reference.

`Get-CFGResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-CFGResourceTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las etiquetas asociadas al recurso dado

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Salida:

```
Key     Value
---     -----
Version 1.3
```

- Para API obtener más información, consulte ListTagsForResourcela referencia del AWS Tools for PowerShell cmdlet.

`Remove-CFGConformancePack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-CFGConformancePack`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el paquete de conformidad indicado, junto con todas las reglas, las acciones correctivas y los resultados de la evaluación del paquete.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteConformancePackReference.

`Write-CFGConformancePack`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-CFGConformancePack`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un paquete de conformidad y se obtiene la plantilla del archivo yaml dado.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- Para API obtener más información, consulte la referencia del PutConformancePackcmdlet AWS Tools for PowerShell .

`Write-CFGDeliveryChannel`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-CFGDeliveryChannel`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia la deliveryFrequency propiedad de un canal de entrega existente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Para API obtener más información, consulte PutDeliveryChannel AWS Tools for PowerShellCmdlet Reference.

Ejemplos de Device Farm que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante Device Farm.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`New-DFUpload`

En el siguiente ejemplo de código se muestra cómo usarlo`New-DFUpload`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una carga de AWS Device Farm para una aplicación de Android. Puede obtener el proyecto ARN a partir de los resultados de New- DFProject o Get-DFProjectList. Usa el mensaje URL de DFUpload salida New- registrado para subir un archivo a 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
```

- Para API obtener más información, consulte CreateUploadla referencia de AWS Tools for PowerShell cmdlets.

AWS Directory Service ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Directory Service.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-DSIpRoute`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-DSIpRoute`.

Herramientas para PowerShell

Ejemplo 1: Este comando elimina la etiqueta de recurso asignada al identificador de directorio especificado

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- Para API obtener más información, consulte la referencia del AddIpRoutes AWS Tools for PowerShellcmdlet.

`Add-DSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-DSResourceTag`

Herramientas para PowerShell

Ejemplo 1: Este comando añade la etiqueta de recurso al identificador de directorio especificado

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Para API obtener más información, consulte la referencia del AddTagsToResource AWS Tools for PowerShellcmdlet.

`Approve-DSTrust`

En el siguiente ejemplo de código se muestra cómo usarlo. `Approve-DSTrust`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se llama a la VerifyTrust API operación AWS Directory Service para el Trustid especificado.

```
Approve-DSTrust -TrustId t-9067157123
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet VerifyTrustReference.

`Confirm-DSSharedDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Confirm-DSSharedDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se acepta una solicitud de compartición de directorios enviada por el propietario del directorio Cuenta de AWS.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Salida:

```
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
```

- Para API obtener más información, consulte AcceptSharedDirectory AWS Tools for PowerShellCmdlet Reference.

`Connect-DSDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Connect-DSDirectory`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un AD Connector para conectarse a un directorio local.

```
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
```

- Para API obtener más información, consulte la referencia ConnectDirectoryde AWS Tools for PowerShell cmdlets.

`Deny-DSSharedDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Deny-DSSharedDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se rechaza una solicitud de uso compartido de directorios enviada desde la cuenta del propietario del directorio.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Salida:

```
d-9067012345
```

- Para API obtener más información, consulte RejectSharedDirectoryla referencia de AWS Tools for PowerShell cmdlets.

`Disable-DSDirectoryShare`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-DSDirectoryShare`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo detiene el uso compartido de directorios entre el propietario del directorio y la cuenta del consumidor.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

Salida:

```
d-9067012345
```

- Para API obtener más información, consulte UnshareDirectoryla referencia del AWS Tools for PowerShell cmdlet.

`Disable-DSLDAPS`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-DSLDAPS`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo desactiva las llamadas LDAP seguras para el directorio especificado.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Para API obtener más información, consulte Disable LDAPS in AWS Tools for PowerShell Cmdlet Reference.

`Disable-DSRadius`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-DSRadius`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo deshabilita el RADIUS servidor configurado para un directorio AD Connector o Microsoft AD.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Para API obtener más información, consulte la referencia DisableRadiusde AWS Tools for PowerShell cmdlets.

`Disable-DSSso`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-DSSso`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se deshabilita el inicio de sesión único en un directorio.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Para API obtener más información, consulte Cmdlet DisableSsoReference AWS Tools for PowerShell .

`Enable-DSDirectoryShare`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-DSDirectoryShare`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se comparte un directorio específico de tu AWS cuenta con otra AWS cuenta mediante el método Handshake.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Salida:

```
d-9067012345
```

- Para API obtener más información, consulte la referencia ShareDirectorydel AWS Tools for PowerShell cmdlet.

`Enable-DSLDAPS`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-DSLDAPS`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo activa el conmutador para que el directorio específico utilice siempre llamadas LDAP seguras.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Para API obtener más información, consulte Enable LDAPS in AWS Tools for PowerShell Cmdlet Reference.

`Enable-DSRadius`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-DSRadius`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita la autenticación multifactor (MFA) con la configuración de RADIUS servidor proporcionada para un AD Connector o un directorio Microsoft AD.

```
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
```

- Para API obtener más información, consulte la referencia EnableRadiusde AWS Tools for PowerShell cmdlets.

`Enable-DSSso`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-DSSso`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita el inicio de sesión único en un directorio.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet EnableSsoReference.

`Get-DSCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSCertificate`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra información sobre el certificado registrado para una LDAP conexión segura.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Salida:

```
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.
```

- Para API obtener más información, consulte DescribeCertificate AWS Tools for PowerShellCmdlet Reference.

`Get-DSCertificateList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSCertificateList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los certificados registrados para una LDAP conexión segura en el directorio especificado.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Salida:

```
CertificateId CommonName                ExpiryDateTime       State
```

```
------------ ----------                 --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- Para API obtener más información, consulte ListCertificatesla referencia de AWS Tools for PowerShell cmdlets.

`Get-DSConditionalForwarder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSConditionalForwarder`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene todos los reenviadores condicionales configurados para un identificador de directorio determinado.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Salida:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- Para API obtener más información, consulte la referencia del DescribeConditionalForwarderscmdlet AWS Tools for PowerShell .

`Get-DSDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSDirectory`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene información sobre los directorios que pertenecen a esta cuenta.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeDirectories AWS Tools for PowerShellCmdlet Reference.

`Get-DSDirectoryLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSDirectoryLimit`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra la información del límite de directorios para la región us-east-1.

```
Get-DSDirectoryLimit -Region us-east-1
```

Salida:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- Para obtener API más información, consulte GetDirectoryLimitsla referencia del cmdlet.AWS Tools for PowerShell

`Get-DSDomainControllerList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSDomainControllerList`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene la lista detallada de los controladores de dominio lanzados para el identificador de directorio mencionado

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Salida:

```
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
```

- Para API obtener más información, consulte la Referencia de DescribeDomainControllers AWS Tools for PowerShellcmdlets.

`Get-DSEventTopic`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSEventTopic`

Herramientas para PowerShell

Ejemplo 1: Este comando muestra la información del SNS tema configurado para su notificación mientras cambia el estado del directorio.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Salida:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- Para API obtener más información, consulte DescribeEventTopicsla referencia de AWS Tools for PowerShell cmdlets.

`Get-DSIpRouteList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSIpRouteList`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene los bloques de direcciones IP públicas configurados en el enrutamiento IP de directorio

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Salida:

```
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 :
```

- Para API obtener más información, consulte ListIpRoutesla referencia de AWS Tools for PowerShell cmdlets.

`Get-DSLDAPSSetting`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSLDAPSSetting`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el estado de LDAP seguridad del directorio especificado.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

Salida:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- Para API obtener más información, consulte D escribeLDAPSSettings en la referencia del AWS Tools for PowerShell cmdlet.

`Get-DSLogSubscriptionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSLogSubscriptionList`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene la información de registro de suscripciones del identificador de directorio especificado

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Salida:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 --------------------------
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- Para API obtener más información, consulte la referencia del ListLogSubscriptions AWS Tools for PowerShellcmdlet.

`Get-DSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSResourceTag`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene todas las etiquetas del directorio especificado.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Salida:

```
Key   Value
---   -----
myTag myTagValue
```

- Para API obtener más información, consulte ListTagsForResourcela referencia del AWS Tools for PowerShell cmdlet.

`Get-DSSchemaExtension`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSSchemaExtension`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las extensiones de esquema aplicadas a un directorio de Microsoft AD.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Salida:

```
Description                : ManagedADSchemaExtension
DirectoryId                : d-123456ijkl
EndDateTime                : 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
```

- Para API obtener más información, consulte ListSchemaExtensions AWS Tools for PowerShellCmdlet Reference.

`Get-DSSharedDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSSharedDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los directorios compartidos de su AWS cuenta

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeSharedDirectoriesla referencia de AWS Tools for PowerShell cmdlets.

`Get-DSSnapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSSnapshot`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene información sobre las instantáneas de directorio especificadas que pertenecen a esta cuenta.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeSnapshotsdel AWS Tools for PowerShell cmdlet.

`Get-DSSnapshotLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSSnapshotLimit`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene los límites de instantáneas manuales para un directorio específico.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Salida:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- Para API obtener más información, consulte GetSnapshotLimitsla referencia de AWS Tools for PowerShell cmdlets.

`Get-DSTrust`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DSTrust`

Herramientas para PowerShell

Ejemplo 1: Este comando obtiene la información de las relaciones de confianza creadas para el identificador de directorio especificado.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DescribeTrusts AWS Tools for PowerShellcmdlet.

`New-DSAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSAlias`

Herramientas para PowerShell

Ejemplo 1: Este comando crea un alias para un directorio y lo asigna al identificador de directorio especificado.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Salida:

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- Para API obtener más información, consulte la referencia del CreateAliascmdlet AWS Tools for PowerShell .

`New-DSComputer`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSComputer`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo objeto de equipo de Active Directory.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

Salida:

```
ComputerAttributes                  ComputerId
 ComputerName
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- Para API obtener más información, consulte CreateComputer AWS Tools for PowerShellCmdlet Reference.

`New-DSConditionalForwarder`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSConditionalForwarder`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un reenviador condicional en el ID de AWS directorio especificado.

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Para API obtener más información, consulte Cmdlet CreateConditionalForwarderReference AWS Tools for PowerShell .

`New-DSDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo directorio Simple AD.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para API obtener más información, consulte CreateDirectorylla referencia del AWS Tools for PowerShell cmdlet.

`New-DSLogSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSLogSubscription`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una suscripción para reenviar los registros de seguridad del controlador de dominio de Directory Service en tiempo real al grupo de registros de Amazon CloudWatch especificado en su Cuenta de AWS.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- Para API obtener más información, consulte CreateLogSubscriptionla referencia de AWS Tools for PowerShell cmdlets.

`New-DSMicrosoftAD`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSMicrosoftAD`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un nuevo directorio de Microsoft AD en Nube de AWS.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para API obtener más información, consulte CreateMicrosoftAD en la referencia de AWS Tools for PowerShell cmdlets.

`New-DSSnapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSSnapshot`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una instantánea del directorio

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Para API obtener más información, consulte CreateSnapshot AWS Tools for PowerShellCmdlet Reference.

`New-DSTrust`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DSTrust`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una confianza bidireccional en todo el bosque entre el directorio administrado de AWS Microsoft AD y el Microsoft Active Directory local existente.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

Salida:

```
t-9067157123
```

- Para API obtener más información, consulte la referencia de CreateTrustcmdlets AWS Tools for PowerShell .

`Register-DSCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-DSCertificate`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se registra un certificado para una LDAP conexión segura.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Salida:

```
c-906731e350
```

- Para API obtener más información, consulte RegisterCertificate AWS Tools for PowerShellCmdlet Reference.

`Register-DSEventTopic`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-DSEventTopic`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se asocia un directorio como editor a un SNS tema.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para API obtener más información, consulte RegisterEventTopic AWS Tools for PowerShellCmdlet Reference.

`Remove-DSConditionalForwarder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSConditionalForwarder`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el reenviador condicional que se ha configurado para su AWS directorio.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- Para API obtener más información, consulte la referencia de DeleteConditionalForwardercmdlets AWS Tools for PowerShell .

`Remove-DSDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina un AWS directorio de servicios de directorio (conector AD/Microsoft AD/AD simple)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Para obtener API más información, consulte la referencia de cmdlets. DeleteDirectoryAWS Tools for PowerShell

`Remove-DSIpRoute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSIpRoute`

Herramientas para PowerShell

Ejemplo 1: Este comando elimina la IP especificada de las rutas IP configuradas del ID de directorio.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Para API obtener más información, consulte la referencia del RemoveIpRoutes AWS Tools for PowerShellcmdlet.

`Remove-DSLogSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSLogSubscription`

Herramientas para PowerShell

Ejemplo 1: Este comando elimina la suscripción de registro del identificador de directorio especificado

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- Para API obtener más información, consulte la referencia del DeleteLogSubscription AWS Tools for PowerShellcmdlet.

`Remove-DSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSResourceTag`

Herramientas para PowerShell

Ejemplo 1: Este comando elimina la etiqueta de recurso asignada al identificador de directorio especificado

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Para API obtener más información, consulte la referencia del RemoveTagsFromResource AWS Tools for PowerShellcmdlet.

`Remove-DSSnapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSSnapshot`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la instantánea creada manualmente.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Para API obtener más información, consulte DeleteSnapshot AWS Tools for PowerShellCmdlet Reference.

`Remove-DSTrust`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DSTrust`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la relación de confianza existente entre el directorio de AD AWS administrado y un dominio externo.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Salida:

```
t-9067157123
```

- Para API obtener más información, consulta la Referencia de DeleteTrustcmdlets AWS Tools for PowerShell .

`Reset-DSUserPassword`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-DSUserPassword`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se restablece la contraseña del usuario de Active Directory nombrado ADUser en el directorio AWS administrado de Microsoft AD o en el directorio Simple AD

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- Para API obtener más información, consulte la referencia de ResetUserPasswordcmdlets AWS Tools for PowerShell .

`Restore-DSFromSnapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Restore-DSFromSnapshot`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se restaura un directorio utilizando una instantánea de directorio existente.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet RestoreFromSnapshotReference.

`Set-DSDomainControllerCount`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-DSDomainControllerCount`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se establece el número de controladores de dominio en 3 para el identificador de directorio especificado.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Para API obtener más información, consulte la referencia del UpdateNumberOfDomainControllers AWS Tools for PowerShellcmdlet.

`Start-DSSchemaExtension`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-DSSchemaExtension`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo aplica una extensión de esquema a un directorio de Microsoft AD.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Salida:

```
e-9067306643
```

- Para API obtener más información, consulte StartSchemaExtension AWS Tools for PowerShellCmdlet Reference.

`Stop-DSSchemaExtension`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-DSSchemaExtension`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cancela una extensión de esquema en curso a un directorio de Microsoft AD.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Para API obtener más información, consulte la referencia de CancelSchemaExtension AWS Tools for PowerShellcmdlets.

`Unregister-DSCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-DSCertificate`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina del sistema el certificado que estaba registrado para una LDAP conexión segura.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Para API obtener más información, consulte DeregisterCertificate AWS Tools for PowerShellCmdlet Reference.

`Unregister-DSEventTopic`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-DSEventTopic`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el directorio especificado como editor del tema especificadoSNS.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeregisterEventTopicReference.

`Update-DSConditionalForwarder`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-DSConditionalForwarder`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza un reenviador condicional que se ha configurado para su AWS directorio.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Para API obtener más información, consulte la Referencia UpdateConditionalForwarderde AWS Tools for PowerShell cmdlets.

`Update-DSRadius`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-DSRadius`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza la información RADIUS del servidor para un directorio AD Connector o Microsoft AD.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Para API obtener más información, consulte UpdateRadiusla referencia de AWS Tools for PowerShell cmdlets.

`Update-DSTrust`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-DSTrust`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza el SelectiveAuth parámetro del identificador de confianza especificado de Desactivado a Activado.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Salida:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- Para API obtener más información, consulte la referencia del UpdateTrust AWS Tools for PowerShellcmdlet.

AWS DMS ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS DMS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`New-DMSReplicationTask`

En el siguiente ejemplo de código se muestra cómo usarlo`New-DMSReplicationTask`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva tarea de replicación del AWS Database Migration Service que utiliza CdcStartTime en lugar de CdcStartPosition. El MigrationType valor está establecido en full-load-and-cdc ««, lo que significa que la tabla de destino debe estar vacía. La nueva tarea se etiqueta con una etiqueta que tiene una clave de Stage y un valor clave de Test. Para obtener más información sobre los valores que utiliza este cmdlet, consulte Creación de una tarea (https://docs.aws.amazon.com/dms/latest/userguide/ CHAP _tasks.Creating.html) en la Guía del usuario de Database Migration Service. 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"
```

- Para obtener API más información CreateReplicationTask, consulte la referencia del cmdlet.AWS Tools for PowerShell

Ejemplos de DynamoDB que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS Tools for PowerShell

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-DDBIndexSchema`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-DDBIndexSchema`.

Herramientas para PowerShell

Ejemplo 1: crea un TableSchema objeto vacío y le añade una nueva definición de índice secundario local antes de escribir el TableSchema objeto en la canalización.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Salida:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

Ejemplo 2: añade una nueva definición de índice secundario local al TableSchema objeto suministrado antes de volver a escribir el TableSchema objeto en la canalización. El TableSchema objeto también se puede proporcionar mediante el parámetro -Scheme.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Salida:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Add- DDBIndexSchema in.

Add-DDBKeySchema

En el siguiente ejemplo de código se muestra cómo usarlo. Add-DDBKeySchema

Herramientas para PowerShell

Ejemplo 1: crea un TableSchema objeto vacío y le añade entradas de definición de claves y atributos utilizando los datos clave especificados antes de escribir el TableSchema objeto en la canalización. El tipo de clave se declara como 'HASH' de forma predeterminada; utilice el KeyType parámetro - con el valor 'RANGE' para declarar una clave de rango.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Salida:

```
AttributeSchema                          KeySchema
    LocalSecondaryIndexSchema
---------------                          ---------
    -------------------------
{ForumName}                              {ForumName}
    {}
```

Ejemplo 2: añade nuevas entradas de definición de atributos y claves al TableSchema objeto suministrado antes de escribir el TableSchema objeto en la canalización. El tipo de clave se declara como 'HASH' de forma predeterminada; utilice el KeyType parámetro - con el valor 'RANGE' para declarar una clave de rango. El TableSchema objeto también se puede proporcionar mediante el parámetro -Scheme.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Salida:

```
AttributeSchema                          KeySchema
    LocalSecondaryIndexSchema
---------------                          ---------
    -------------------------
{ForumName}                              {ForumName}
    {}
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Add- DDBKeySchema in.

`ConvertFrom-DDBItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `ConvertFrom-DDBItem`

Herramientas para PowerShell

Ejemplo 1: ConvertFrom - DDBItem se usa para convertir el resultado de Get- DDBItem de una tabla hash de AttributeValues DynamoDB a una tabla hash de tipos comunes como string y double.

```
@{
```

```
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Salida:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Para API obtener más información, consulte ConvertFrom- DDBItem en Cmdlet Reference.AWS Tools for PowerShell

`ConvertTo-DDBItem`

El siguiente ejemplo de código muestra cómo usarlo. `ConvertTo-DDBItem`

Herramientas para PowerShell

Ejemplo 1: ejemplo de conversión de una tabla hash en un diccionario de valores de atributos de DynamoDB.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

Ejemplo 2: ejemplo de conversión de una tabla hash en un diccionario de valores de atributos de DynamoDB.

```
@{
    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
```

Salida:

```
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
```

- Para API obtener más información, consulte ConvertTo- DDBItem en AWS Tools for PowerShell Cmdlet Reference.

`Get-DDBBatchItem`

El siguiente ejemplo de código muestra cómo usarlo. `Get-DDBBatchItem`

Herramientas para PowerShell

Ejemplo 1: Obtiene el elemento SongTitle «Somewhere Down The Road» de las tablas «Music» y «Songs» de DynamoDB.

```
$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
```

Salida:

```
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
```

- Para obtener API más información, consulte BatchGetItemCmdlet Reference.AWS Tools for PowerShell

`Get-DDBItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DDBItem`

Herramientas para PowerShell

Ejemplo 1: devuelve el elemento de DynamoDB con la clave de partición y la SongTitle clave de clasificación Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Salida:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet GetItemReference.

`Get-DDBTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DDBTable`

Herramientas para PowerShell

Ejemplo 1: devuelve información detallada de la tabla especificada.

```
Get-DDBTable -TableName "myTable"
```

- Para API obtener más información, consulte DescribeTablela referencia de AWS Tools for PowerShell cmdlets.

`Get-DDBTableList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-DDBTableList`

Herramientas para PowerShell

Ejemplo 1: devuelve los detalles de todas las tablas y se itera automáticamente hasta que el servicio indique que no existen más tablas.

```
Get-DDBTableList
```

Ejemplo 2: itera manualmente los detalles de todas las tablas y devuelve hasta 10 tablas por llamada que el servicio indique que no existen más tablas.

```
$nextToken = $null
do {
  Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
  $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte ListTablesla referencia de AWS Tools for PowerShell cmdlets.

`Invoke-DDBQuery`

En el siguiente ejemplo de código se muestra cómo usarlo. `Invoke-DDBQuery`

Herramientas para PowerShell

Ejemplo 1: Invoca una consulta que devuelve elementos de DynamoDB con el identificador y el artista especificados. 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
```

Salida:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Para API obtener más información, consulte Query in AWS Tools for PowerShell Cmdlet Reference.

`Invoke-DDBScan`

En el siguiente ejemplo de código se muestra cómo usarlo. `Invoke-DDBScan`

Herramientas para PowerShell

Ejemplo 1: devuelve todos los elementos de la tabla Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Salida:

```
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
```

Ejemplo 2: Devuelve los elementos de la tabla Música con un valor CriticRating mayor o igual a nueve.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Salida:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Para API obtener más información, consulte Scan in AWS Tools for PowerShell Cmdlet Reference.

`New-DDBTable`

El siguiente ejemplo de código muestra cómo usarlo. `New-DDBTable`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una tabla denominada Thread que tiene una clave principal compuesta por «ForumName» (tipo hash de clave) y «Subject» (rango de tipos de clave). El esquema utilizado para construir la tabla se puede canalizar hacia cada cmdlet tal como se muestra o se especifica con el parámetro -Schema.

```
$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
```

Salida:

```
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  : {}
```

Ejemplo 2: en este ejemplo se crea una tabla denominada Thread que tiene una clave principal compuesta por «ForumName» (tipo hash de clave) y «Asunto» (rango de tipos de clave). También se define un índice secundario local. La clave del índice secundario local se establecerá automáticamente a partir de la clave hash principal de la tabla (ForumName). El esquema utilizado para construir la tabla se puede canalizar hacia cada cmdlet tal como se muestra o se especifica con el parámetro -Schema.

```
$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
```

Salida:

```
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}
```

Ejemplo 3: En este ejemplo se muestra cómo utilizar una canalización única para crear una tabla denominada Thread que tenga una clave principal compuesta por «ForumName» (hash de

tipo de clave) y «Asunto» (rango de tipos de clave) y un índice secundario local. Los comandos Add- DDBKeySchema y Add- DDBIndexSchema crean un TableSchema objeto nuevo si no lo proporcionan la canalización o el parámetro -Scheme.

```
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
```

Salida:

```
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}
```

- Para API obtener más información, consulte la referencia CreateTabledel AWS Tools for PowerShell cmdlet.

`New-DDBTableSchema`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-DDBTableSchema`

Herramientas para PowerShell

Ejemplo 1: crea un TableSchema objeto vacío listo para aceptar definiciones de claves e índices para su uso en la creación de una nueva tabla de Amazon DynamoDB. El objeto devuelto puede canalizarse a los DDBTable cmdlets Add-DDBKeySchema, Add- DDBIndexSchema y New- o pasarse a ellos mediante el parámetro -Scheme de cada cmdlet.

```
New-DDBTableSchema
```

Salida:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{}                                          {}
    {}
```

- Para obtener API más información, consulte la referencia del cmdlet New- in. DDBTableSchema AWS Tools for PowerShell

`Remove-DDBItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DDBItem`

Herramientas para PowerShell

Ejemplo 1: elimina el elemento DynamoDB que coincide con la clave proporcionada.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Para API obtener más información, consulte DeleteItemla referencia de AWS Tools for PowerShell cmdlets.

`Remove-DDBTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-DDBTable`

Herramientas para PowerShell

Ejemplo 1: elimina la tabla especificada. Se le solicitará una confirmación antes de continuar con la operación.

```
Remove-DDBTable -TableName "myTable"
```

Ejemplo 2: elimina la tabla especificada. No se le solicitará una confirmación antes de continuar con la operación.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Para API obtener más información, consulte DeleteTablela referencia de AWS Tools for PowerShell cmdlets.

`Set-DDBBatchItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-DDBBatchItem`

Herramientas para PowerShell

Ejemplo 1: crea un nuevo elemento o sustituye un elemento existente por uno nuevo en las tablas Music y Songs de DynamoDB.

```
$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
```

Salida:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Para API obtener más información, consulte BatchWriteItemla referencia de AWS Tools for PowerShell cmdlets.

`Set-DDBItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-DDBItem`

Herramientas para PowerShell

Ejemplo 1: crea un nuevo elemento o sustituye un elemento existente por uno nuevo.

```
$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
```

- Para API obtener más información, consulte PutItemla referencia de AWS Tools for PowerShell cmdlets.

`Update-DDBItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-DDBItem`

Herramientas para PowerShell

Ejemplo 1: Establece el atributo de género en «Rap» en el elemento de DynamoDB con la clave de partición y la SongTitle clave de clasificación Artist.

```
$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
```

Salida:

```
Name                           Value
----                           -----
Genre                          Rap
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet UpdateItemReference.

`Update-DDBTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-DDBTable`

Herramientas para PowerShell

Ejemplo 1: actualiza el rendimiento aprovisionado de la tabla en cuestión.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Para API obtener más información, consulte UpdateTablela referencia de AWS Tools for PowerShell cmdlets.

EC2Ejemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonEC2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-EC2CapacityReservation`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-EC2CapacityReservation`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva reserva de capacidad con los atributos especificados

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateCapacityReservation AWS Tools for PowerShellCmdlet Reference.

`Add-EC2InternetGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-EC2InternetGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se adjunta la puerta de enlace de Internet especificada a la especificadaVPC.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Ejemplo 2: Este ejemplo crea una puerta de enlace de Internet VPC y, a continuación, conecta la puerta de enlace de Internet a. VPC

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Para API obtener más información, consulte la referencia AttachInternetGatewaydel AWS Tools for PowerShell cmdlet.

`Add-EC2NetworkInterface`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-EC2NetworkInterface`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta la interfaz de red especificada a la instancia especificada.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Salida:

```
eni-attach-1a2b3c4d
```

- Para API obtener más información, consulte AttachNetworkInterface AWS Tools for PowerShellCmdlet Reference.

`Add-EC2Volume`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-EC2Volume`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta el volumen especificado a la instancia especificada y se expone con el nombre de dispositivo especificado.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Salida:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet AttachVolumeReference.

`Add-EC2VpnGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-EC2VpnGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta la puerta de enlace privada virtual especificada a la especificadaVPC.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Salida:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Para API obtener más información, consulte AttachVpnGateway AWS Tools for PowerShellCmdlet Reference.

`Approve-EC2VpcPeeringConnection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Approve-EC2VpcPeeringConnection`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se aprueba el pcx-1dfad234b56ff78be solicitado VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Salida:

```
AccepterVpcInfo        : 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
```

- APIPara obtener más AcceptVpcPeeringConnection AWS Tools for PowerShellinformación, consulte la referencia del cmdlet.

`Confirm-EC2ProductInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Confirm-EC2ProductInstance`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo determina si el código de producto especificado está asociado a la instancia especificada.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Para API obtener más información, consulte ConfirmProductInstance AWS Tools for PowerShellCmdlet Reference.

`Copy-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Copy-EC2Image`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se copia lo especificado AMI en la región «UE (Irlanda)» en la región «EE.UU. Oeste (Oregón)». Si no se especifica -Region, la región predeterminada actual se utiliza como región de destino.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2
 -Name "Copy of ami-12345678"
```

Salida:

```
ami-87654321
```

- Para API obtener más información, consulte la referencia CopyImagede AWS Tools for PowerShell cmdlets.

`Copy-EC2Snapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Copy-EC2Snapshot`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se copia la instantánea especificada de la región de la UE (Irlanda) a la región de EE.UU. Oeste (Oregón).

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-
west-2
```

Ejemplo 2: Si establece una región predeterminada y omite el parámetro Región, la región de destino predeterminada será la región predeterminada.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Para API obtener más información, consulte la referencia CopySnapshotdel AWS Tools for PowerShell cmdlet.

`Deny-EC2VpcPeeringConnection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Deny-EC2VpcPeeringConnection`

Herramientas para PowerShell

Ejemplo 1: El ejemplo anterior deniega la solicitud del identificador de solicitud VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Para obtener RejectVpcPeeringConnectionmás AWS Tools for PowerShell información, consulte la referencia del cmdlet. API

`Disable-EC2VgwRoutePropagation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-EC2VgwRoutePropagation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo impide que las VGW rutas se propaguen automáticamente a la tabla de enrutamiento especificada.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DisableVgwRoutePropagationReference.

`Disable-EC2VpcClassicLink`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-EC2VpcClassicLink`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo se desactiva EC2VpcClassicLink para el vpc-01e23c4a5d6db78e9. Devuelve Verdadero o Falso

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Para API obtener más información, consulte DisableVpcClassicLink AWS Tools for PowerShellCmdlet Reference.

`Disable-EC2VpcClassicLinkDnsSupport`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-EC2VpcClassicLinkDnsSupport`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se deshabilita la ClassicLink DNS compatibilidad con el vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Para obtener DisableVpcClassicLinkDnsSupportmás AWS Tools for PowerShell información, consulte la referencia del cmdlet. API

`Dismount-EC2InternetGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-EC2InternetGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se separa la puerta de enlace de Internet especificada de la especificadaVPC.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Para API obtener más información, consulte la referencia DetachInternetGatewaydel AWS Tools for PowerShell cmdlet.

`Dismount-EC2NetworkInterface`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-EC2NetworkInterface`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el adjunto especificado entre una interfaz de red y una instancia.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Para API obtener más información, consulte DetachNetworkInterface AWS Tools for PowerShellCmdlet Reference.

`Dismount-EC2Volume`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-EC2Volume`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se separa el volumen especificado.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Salida:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Ejemplo 2: También puede especificar el ID de la instancia y el nombre del dispositivo para asegurarse de que está separando el volumen correcto.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Para API obtener más información, consulte la referencia DetachVolumedel AWS Tools for PowerShell cmdlet.

`Dismount-EC2VpnGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-EC2VpnGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se separa la puerta de enlace privada virtual especificada de la especificadaVPC.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Para API obtener más información, consulte la referencia DetachVpnGatewaydel AWS Tools for PowerShell cmdlet.

`Edit-EC2CapacityReservation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2CapacityReservation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica el CapacityReservationId cr-0c1f2345db6f7cdba cambiando el recuento de instancias a 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Salida:

```
True
```

- Para obtener ModifyCapacityReservation AWS Tools for PowerShellmás información, consulte Cmdlet Reference. API

`Edit-EC2Host`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2Host`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica la AutoPlacement configuración a desactivada para el host dedicado h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Salida:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Para obtener ModifyHostsmás AWS Tools for PowerShell información, consulte Cmdlet Reference. API

`Edit-EC2IdFormat`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2IdFormat`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita el formato de ID más largo para el tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Ejemplo 2: Este ejemplo deshabilita el formato de ID más largo para el tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Para API obtener más información, consulte ModifyIdFormat AWS Tools for PowerShellCmdlet Reference.

`Edit-EC2ImageAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2ImageAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza la descripción de lo especificadoAMI.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Ejemplo 2: En este ejemplo se hace AMI público (por ejemplo, para que cualquiera Cuenta de AWS pueda usarlo).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Ejemplo 3: Este ejemplo hace que lo sea AMI privado (por ejemplo, para que solo tú, como propietario, puedas usarlo).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Ejemplo 4: en este ejemplo se concede el permiso de lanzamiento a la persona especificada Cuenta de AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Ejemplo 5: en este ejemplo se elimina el permiso de lanzamiento del especificado Cuenta de AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Para API obtener más información, consulte ModifyImageAttribute AWS Tools for PowerShellCmdlet Reference.

`Edit-EC2InstanceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2InstanceAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se modifica el tipo de instancia de la instancia especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Ejemplo 2: Este ejemplo permite una red mejorada para la instancia especificada, especificando «simple» como el valor del parámetro de soporte de red de virtualización de E/S (SR-IOV) de raíz única, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Ejemplo 3: En este ejemplo se modifican los grupos de seguridad de la instancia especificada. La instancia debe estar en unVPC. Debe especificar el ID de cada grupo de seguridad, no su nombre.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Ejemplo 4: Este ejemplo permite la optimización EBS de E/S para la instancia especificada. Esta función no está disponible en todos los tipos de instancias. Se aplican cargos de uso adicionales al usar una instancia EBS optimizada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Ejemplo 5: Este ejemplo permite comprobar el origen y el destino de la instancia especificada. Para que una NAT instancia funcioneNAT, el valor debe ser «falso».

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Ejemplo 6: En este ejemplo se inhabilita la terminación de la instancia especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Ejemplo 7: En este ejemplo se cambia la instancia especificada para que finalice cuando se inicie el cierre desde la instancia.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- Para API obtener más información, consulte la referencia ModifyInstanceAttributedel AWS Tools for PowerShell cmdlet.

`Edit-EC2InstanceCreditSpecification`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2InstanceCreditSpecification`

Herramientas para PowerShell

Ejemplo 1: Esto permite créditos T2 ilimitados, por ejemplo, el i-01234567890abcdef.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
```

```
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Para obtener ModifyInstanceCreditSpecificationmás AWS Tools for PowerShell información, consulte la referencia del cmdlet. API

`Edit-EC2NetworkInterfaceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2NetworkInterfaceAttribute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica la interfaz de red especificada para que el adjunto especificado se elimine al finalizar.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Ejemplo 2: este ejemplo modifica la descripción de la interfaz de red especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

Ejemplo 3: Este ejemplo modifica el grupo de seguridad de la interfaz de red especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Ejemplo 4: En este ejemplo, se deshabilita la comprobación del origen y el destino de la interfaz de red especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- Para API obtener más información, consulte la referencia de ModifyNetworkInterfaceAttributecmdlets AWS Tools for PowerShell .

`Edit-EC2ReservedInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2ReservedInstance`

Herramientas para PowerShell

Ejemplo 1: este ejemplo modifica la zona de disponibilidad, el recuento de instancias y la plataforma de las instancias reservadas especificadas.

```
$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
```

- Para API obtener más información, consulte la referencia ModifyReservedInstancesdel AWS Tools for PowerShell cmdlet.

`Edit-EC2SnapshotAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2SnapshotAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se hace pública la instantánea especificada mediante el establecimiento de su CreateVolumePermission atributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Para API obtener más información, consulte ModifySnapshotAttribute AWS Tools for PowerShellCmdlet Reference.

`Edit-EC2SpotFleetRequest`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2SpotFleetRequest`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza la capacidad objetivo de la solicitud de flota de Spot especificada.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Salida:

```
True
```

- Para API obtener más información, consulte ModifySpotFleetRequest AWS Tools for PowerShellCmdlet Reference.

`Edit-EC2SubnetAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2SubnetAttribute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita el direccionamiento IP público para la subred especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Ejemplo 2: Este ejemplo deshabilita el direccionamiento IP público para la subred especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Para API obtener más información, consulte la referencia de ModifySubnetAttribute AWS Tools for PowerShellcmdlets.

`Edit-EC2VolumeAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2VolumeAttribute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica el atributo especificado del volumen especificado. Las operaciones de E/S del volumen se reanudan automáticamente tras la suspensión debido a la posible incoherencia de los datos.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Para API obtener más información, consulte la referencia ModifyVolumeAttributede AWS Tools for PowerShell cmdlets.

`Edit-EC2VpcAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-EC2VpcAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se habilita la compatibilidad con los DNS nombres de host del objeto especificadoVPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Ejemplo 2: En este ejemplo se deshabilita la compatibilidad con los DNS nombres de host de los especificados. VPC

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Ejemplo 3: Este ejemplo habilita la compatibilidad con la DNS resolución de lo especificado. VPC

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Ejemplo 4: En este ejemplo se deshabilita la compatibilidad con la DNS resolución de lo especificadoVPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Para API obtener más información, consulte la referencia ModifyVpcAttributedel AWS Tools for PowerShell cmdlet.

`Enable-EC2VgwRoutePropagation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-EC2VgwRoutePropagation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite VGW al especificado propagar las rutas automáticamente a la tabla de enrutamiento especificada.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para API obtener más información, consulte EnableVgwRoutePropagation AWS Tools for PowerShellCmdlet Reference.

`Enable-EC2VolumeIO`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-EC2VolumeIO`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita las operaciones de E/S para el volumen especificado, si las operaciones de E/S estaban deshabilitadas.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Para API obtener más información, consulte la referencia de EnableVolumeIo AWS Tools for PowerShellcmdlets.

`Enable-EC2VpcClassicLink`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-EC2VpcClassicLink`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita VPC vpc-0123456b789b0d12f para ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Salida:

```
True
```

- Para API obtener más EnableVpcClassicLinkinformación AWS Tools for PowerShell , consulte la referencia del cmdlet.

`Enable-EC2VpcClassicLinkDnsSupport`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-EC2VpcClassicLinkDnsSupport`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite que vpc-0b12d3456a7e8910d admita la resolución de nombres de host para DNS ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- APIPara obtener EnableVpcClassicLinkDnsSupportmás AWS Tools for PowerShell información, consulte Cmdlet Reference.

`Get-EC2AccountAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2AccountAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe si puede lanzar instancias en EC2 -Classic y EC2 - VPC en la región, o solo en EC2 -VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Salida:

```
AttributeValue
--------------
EC2
VPC
```

Ejemplo 2: En este ejemploVPC, se describe el valor predeterminado o es «ninguno» si no se tiene un valor predeterminado VPC en la región.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Salida:

```
AttributeValue
--------------
vpc-12345678
```

Ejemplo 3: en este ejemplo se describe el número máximo de instancias bajo demanda que puede ejecutar.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Salida:

```
AttributeValue
--------------
20
```

- Para API obtener más información, consulte DescribeAccountAttributesla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2Address`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Address`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la dirección IP elástica especificada para las instancias de EC2 -Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen las direcciones IP elásticas para las instancias de unVPC. Esta sintaxis requiere PowerShell la versión 3 o posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Ejemplo 3: en este ejemplo se describe la dirección IP elástica especificada para las instancias de EC2 -Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Salida:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Ejemplo 4: En este ejemplo, se describen las direcciones IP elásticas para las instancias de EC2 -Classic. Esta sintaxis requiere la PowerShell versión 3 o posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Ejemplo 5: en este ejemplo se describen todas las direcciones IP elásticas.

```
Get-EC2Address
```

Ejemplo 6: Este ejemplo devuelve la IP pública y privada del identificador de instancia proporcionado en el filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Salida:

```
PrivateIpAddress PublicIp
```

```
---------------- --------
10.0.0.99        63.36.5.227
```

Ejemplo 7: Este ejemplo recupera todo el Elastic IPs con su identificador de asignación, su identificador de asociación y sus identificadores de instancia

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Salida:

```
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
```

Ejemplo 8: Este ejemplo busca una lista de direcciones EC2 IP que coinciden con la clave de etiqueta «Categoría» con el valor «Prod»

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Salida:

```
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}
```

- Para API obtener más información, consulte la referencia del DescribeAddresses AWS Tools for PowerShellcmdlet.

`Get-EC2AvailabilityZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2AvailabilityZone`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen las zonas de disponibilidad de la región actual que están disponibles para usted.

```
Get-EC2AvailabilityZone
```

Salida:

```
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
```

Ejemplo 2: Este ejemplo describe cualquier zona de disponibilidad que se encuentre en un estado deteriorado. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Ejemplo 3: Con la PowerShell versión 2, debe usar New-Object para crear el filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"
```

```
Get-EC2AvailabilityZone -Filter $filter
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeAvailabilityZonesReference.

`Get-EC2BundleTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2BundleTask`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la tarea de agrupamiento especificada.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Ejemplo 2: en este ejemplo se describen las tareas del paquete cuyo estado es «completado» o «fallido».

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Para API obtener más información, consulte la referencia del DescribeBundleTaskscmdlet AWS Tools for PowerShell .

`Get-EC2CapacityReservation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2CapacityReservation`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen una o más de sus reservas de capacidad para la región

```
Get-EC2CapacityReservation -Region eu-west-1
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeCapacityReservationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2ConsoleOutput`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ConsoleOutput`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene el resultado de la consola para la instancia de Linux especificada. La salida de la consola está codificada.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Salida:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Ejemplo 2: Este ejemplo almacena la salida de la consola codificada en una variable y, a continuación, la decodifica.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
```

```
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- Para API obtener más información, consulte GetConsoleOutput AWS Tools for PowerShellCmdlet Reference.

`Get-EC2CustomerGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2CustomerGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la pasarela de clientes especificada.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Salida:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

Ejemplo 2: En este ejemplo se describe cualquier pasarela de clientes cuyo estado esté pendiente o disponible.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Ejemplo 3: En este ejemplo se describen todas las pasarelas de clientes.

```
Get-EC2CustomerGateway
```

- Para API obtener más información, consulte la referencia DescribeCustomerGatewaysde AWS Tools for PowerShell cmdlets.

`Get-EC2DhcpOption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2DhcpOption`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los conjuntos de DHCP opciones.

```
Get-EC2DhcpOption
```

Salida:

```
DhcpConfigurations                  DhcpOptionsId    Tag
------------------                  -------------    ---
{domain-name, domain-name-servers}  dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}  dopt-2a3b4c5d    {}
{domain-name-servers}               dopt-3a4b5c6d    {}
```

Ejemplo 2: en este ejemplo se obtienen los detalles de configuración del conjunto de DHCP opciones especificado.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Salida:

```
Key                 Values
---                 ------
domain-name         {abc.local}
domain-name-servers {10.0.0.101, 10.0.0.102}
```

- Para API obtener más información, consulte DescribeDhcpOptions AWS Tools for PowerShellCmdlet Reference.

`Get-EC2FlowLog`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2FlowLog`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen uno o más registros de flujo con el tipo de destino de registro 's3'

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Salida:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- Para API obtener más información, consulte DescribeFlowLogsla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2Host`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Host`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve los detalles del EC2 anfitrión

```
Get-EC2Host
```

Salida:

```
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              : {}
```

Ejemplo 2: En este ejemplo se consulta el servidor AvailableInstanceCapacity h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Salida:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Para obtener DescribeHostsmás AWS Tools for PowerShell información, consulte la referencia del cmdlet. API

`Get-EC2HostReservationOffering`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2HostReservationOffering`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen las reservas de hosts dedicados que se pueden adquirir para el filtro «instance-family» determinado, donde está PaymentOption «» NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Salida:

```
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
```

- Para API obtener más información, consulte Cmdlet DescribeHostReservationOfferingsReference AWS Tools for PowerShell .

`Get-EC2HostReservationPurchasePreview`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2HostReservationPurchasePreview`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra una vista previa de la compra de una reserva con configuraciones que coinciden con las de su servidor dedicado h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Salida:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- Para GetHostReservationPurchasePreviewobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

`Get-EC2IdFormat`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2IdFormat`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el formato de ID del tipo de recurso especificado.

```
Get-EC2IdFormat -Resource instance
```

Salida:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Ejemplo 2: En este ejemplo se describen los formatos de ID de todos los tipos de recursos que admiten una extensión más largaIDs.

```
Get-EC2IdFormat
```

Salida:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Para API obtener más información, consulte DescribeIdFormatla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2IdentityIdFormat`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2IdentityIdFormat`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el formato de ID del recurso «imagen» para el rol asignado

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

Salida:

```
Deadline            Resource UseLongIds
--------            -------- ----------
```

```
8/2/2018 11:30:00 PM image    True
```

- Para API obtener más información, consulte la referencia del DescribeIdentityIdFormat AWS Tools for PowerShellcmdlet.

`Get-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Image`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe lo especificadoAMI.

```
Get-EC2Image -ImageId ami-12345678
```

Salida:

```
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
```

Ejemplo 2: Este ejemplo describe lo AMIs que tienes.

```
Get-EC2Image -owner self
```

Ejemplo 3: En este ejemplo se describe el público AMIs que ejecuta Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Ejemplo 4: En este ejemplo se describen todos los públicos AMIs de la región «us-west-2".

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

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeImagesReference.

`Get-EC2ImageAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ImageAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtiene la descripción de lo especificadoAMI.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Salida:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Ejemplo 2: en este ejemplo se obtienen los permisos de lanzamiento de lo especificadoAMI.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Salida:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Ejemplo 3: En este ejemplo se comprueba si la red mejorada está habilitada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Salida:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- Para API obtener más información, consulte DescribeImageAttributela referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2ImageByName`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ImageByName`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el conjunto completo de nombres de filtros que se admiten actualmente.

```
Get-EC2ImageByName
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describe lo especificadoAMI. El uso de este comando para localizar un AMI dispositivo es útil porque AWS lanza nuevos Windows AMIs con las actualizaciones más recientes cada mes. Puede especificar el valor 'ImageId' New-EC2Instance para lanzar una instancia utilizando el filtro actual AMI para el filtro especificado.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

Salida:

```
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-xxxxxxxx
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
```

- Para API obtener más información, consulte Get-EC2ImageByNamela referencia del AWS Tools for PowerShell cmdlet.

`Get-EC2ImportImageTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ImportImageTask`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la tarea de importación de imágenes especificada.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Salida:

```
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   :
```

Ejemplo 2: En este ejemplo se describen todas las tareas de importación de imágenes.

```
Get-EC2ImportImageTask
```

Salida:

```
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   :
```

- Para API obtener más información, consulte DescribeImportImageTasksla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2ImportSnapshotTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ImportSnapshotTask`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la tarea de importación de instantáneas especificada.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Salida:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Ejemplo 2: En este ejemplo se describen todas las tareas de importación de instantáneas.

```
Get-EC2ImportSnapshotTask
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeImportSnapshotTasks AWS Tools for PowerShellCmdlet Reference.

`Get-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la instancia especificada.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describen todas las instancias de la región actual, agrupadas por reserva. Para ver los detalles de la instancia, amplíe la colección de instancias dentro de cada objeto de reserva.

```
Get-EC2Instance
```

Salida:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Ejemplo 3: Este ejemplo ilustra el uso de un filtro para consultar EC2 instancias en una subred específica de unVPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeInstances AWS Tools for PowerShellCmdlet Reference.

`Get-EC2InstanceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2InstanceAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el tipo de instancia de la instancia especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Salida:

```
InstanceType                     : t2.micro
```

Ejemplo 2: En este ejemplo se describe si la red mejorada está habilitada para la instancia especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Salida:

```
SriovNetSupport                  : simple
```

Ejemplo 3: en este ejemplo se describen los grupos de seguridad de la instancia especificada.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Salida:

```
GroupId
```

```
-------
sg-12345678
sg-45678901
```

Ejemplo 4: En este ejemplo se describe si EBS la optimización está habilitada para la instancia especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Salida:

```
EbsOptimized                     : False
```

Ejemplo 5: en este ejemplo se describe el atributo disableApiTermination " de la instancia especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Salida:

```
DisableApiTermination            : False
```

Ejemplo 6: En este ejemplo se describe el atributo 'instanceInitiatedShutdownBehavior' de la instancia especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Salida:

```
InstanceInitiatedShutdownBehavior : stop
```

- Para API obtener más información, consulte la referencia del DescribeInstanceAttribute AWS Tools for PowerShellcmdlet.

`Get-EC2InstanceMetadata`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2InstanceMetadata`

Ejemplo 1: Muestra las categorías disponibles de metadatos de instancia que se pueden consultar.

```
Get-EC2InstanceMetadata -ListCategory
```

Salida:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Ejemplo 2: Devuelve el identificador de la Amazon Machine Image (AMI) que se utilizó para lanzar la instancia.

```
Get-EC2InstanceMetadata -Category AmiId
```

Salida:

```
ami-b2e756ca
```

Ejemplo 3: En este ejemplo, se consulta el documento JSON de identidad con formato L de la instancia.

```
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"
}
```

Ejemplo 4: En este ejemplo, se utiliza una consulta de ruta para obtener los macs de la interfaz de red de la instancia.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Salida:

```
02:80:7f:ef:4c:e0/
```

Ejemplo 5: Si hay un IAM rol asociado a la instancia, devuelve información sobre la última vez que se actualizó el perfil de la instancia, incluida la LastUpdated fecha de la instancia InstanceProfileArn, y InstanceProfileId.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Salida:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Para API obtener más información, consulte Get-EC2InstanceMetadata AWS Tools for PowerShellCmdlet Reference.

`Get-EC2InstanceStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2InstanceStatus`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el estado de la instancia especificada.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Salida:

```
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
```

Salida:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Salida:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Salida:

```
Details           Status
-------           ------
{reachability}    ok
```

- Para API obtener más información, consulte DescribeInstanceStatus AWS Tools for PowerShellCmdlet Reference.

`Get-EC2InternetGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2InternetGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la puerta de enlace de Internet especificada.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Salida:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Ejemplo 2: En este ejemplo se describen todas las puertas de enlace de Internet.

```
Get-EC2InternetGateway
```

Salida:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Para API obtener más información, consulte la referencia DescribeInternetGatewaysde AWS Tools for PowerShell cmdlets.

`Get-EC2KeyPair`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2KeyPair`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el key pair especificado.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen todos los pares de claves.

```
Get-EC2KeyPair
```

- Para API obtener más información, consulte DescribeKeyPairs AWS Tools for PowerShellCmdlet Reference.

`Get-EC2NetworkAcl`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2NetworkAcl`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la red especificadaACL.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Salida:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Ejemplo 2: en este ejemplo se describen las reglas de la red especificadaACL.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Salida:

```
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
```

Ejemplo 3: En este ejemplo se describe toda la redACLs.

```
Get-EC2NetworkAcl
```

- Para API obtener más información, consulte DescribeNetworkAclsla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2NetworkInterface`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2NetworkInterface`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la interfaz de red especificada.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describen todas las interfaces de red.

```
Get-EC2NetworkInterface
```

- Para API obtener más información, consulte DescribeNetworkInterfaceslа referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2NetworkInterfaceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2NetworkInterfaceAttribute`

Ejemplo 1: en este ejemplo se describe la interfaz de red especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Salida:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Ejemplo 2: en este ejemplo se describe la interfaz de red especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Salida:

```
Description        : My description
```

Ejemplo 3: en este ejemplo se describe la interfaz de red especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Salida:

```
Groups             : {my-security-group}
```

Ejemplo 4: En este ejemplo se describe la interfaz de red especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Salida:

```
SourceDestCheck    : True
```

- Para API obtener más información, consulte DescribeNetworkInterfaceAttribute AWS Tools for PowerShellCmdlet Reference.

`Get-EC2PasswordData`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2PasswordData`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se descifra la contraseña que Amazon EC2 asignó a la cuenta de administrador de la instancia de Windows especificada. Cuando se especificó un archivo pem, se asume automáticamente la configuración del modificador -Decrypt.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Salida:

```
mYZ(PA9?C)Q
```

Ejemplo 2: (PowerShell solo en Windows) Inspecciona la instancia para determinar el nombre del par de claves utilizado para lanzar la instancia y, a continuación, intenta buscar los datos del par de claves correspondientes en el almacén de configuración del AWS Toolkit for Visual Studio. Si se encuentran los datos del par de claves, se descifra la contraseña.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Salida:

```
mYZ(PA9?C)Q
```

Ejemplo 3: Devuelve los datos de contraseña cifrados de la instancia.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Salida:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Para API obtener más información, consulte GetPasswordData AWS Tools for PowerShellCmdlet Reference.

`Get-EC2PlacementGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2PlacementGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el grupo de ubicación especificado.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Salida:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Para API obtener más información, consulte DescribePlacementGroups AWS Tools for PowerShellCmdlet Reference.

`Get-EC2PrefixList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2PrefixList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se busca lo disponible Servicios de AWS en un formato de lista de prefijos para la región

```
Get-EC2PrefixList
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribePrefixListsReference.

`Get-EC2Region`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Region`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen las regiones que están disponibles para usted.

```
Get-EC2Region
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeRegions AWS Tools for PowerShellCmdlet Reference.

`Get-EC2RouteTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2RouteTable`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen todas las tablas de rutas.

```
Get-EC2RouteTable
```

Salida:

```
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  :
```

Ejemplo 2: Este ejemplo devuelve los detalles de la tabla de rutas especificada.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ejemplo 3: En este ejemplo se describen las tablas de rutas de lo especificadoVPC.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Salida:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Para API obtener más información, consulte DescribeRouteTables AWS Tools for PowerShellCmdlet Reference.

`Get-EC2ScheduledInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ScheduledInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la instancia programada especificada.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen todas las instancias programadas.

```
Get-EC2ScheduledInstance
```

- Para API obtener más información, consulte DescribeScheduledInstancesla referencia del AWS Tools for PowerShell cmdlet.

`Get-EC2ScheduledInstanceAvailability`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2ScheduledInstanceAvailability`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe un horario que se realiza todos los domingos a partir de la fecha especificada.

```
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
```

Salida:

```
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

...
```

Ejemplo 2: Para limitar los resultados, puede agregar filtros para criterios como el sistema operativo, la red y el tipo de instancia.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Para API obtener más información, consulte DescribeScheduledInstanceAvailabilityla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2SecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SecurityGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el grupo de seguridad especificado para unVPC. Cuando trabaje con grupos de seguridad que pertenezcan a, VPC debe utilizar el ID del grupo de

seguridad (GroupId parámetro -), no el nombre (GroupName parámetro -), para hacer referencia al grupo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describe el grupo de seguridad especificado para EC2 -Classic. Al trabajar con grupos de seguridad para EC2 -Classic, puede utilizar el nombre del grupo (GroupName parámetro -) o el ID del grupo (GroupId parámetro -) para hacer referencia al grupo de seguridad.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Salida:

```
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               :
```

Ejemplo 3: en este ejemplo se recuperan todos los grupos de seguridad del vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Para API AWS Tools for PowerShell obtener DescribeSecurityGroupsmás información, consulte la referencia del cmdlet.

`Get-EC2Snapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Snapshot`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la instantánea especificada.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describen las instantáneas que tienen la etiqueta «Nombre».

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Ejemplo 3: En este ejemplo se describen las instantáneas que tienen una etiqueta de «Nombre» con el valor «». TestValue

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Ejemplo 4: En este ejemplo se describen todas las instantáneas.

```
Get-EC2Snapshot -Owner self
```

- Para API obtener más información, consulte DescribeSnapshots AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SnapshotAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SnapshotAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el atributo especificado de la instantánea especificada.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Salida:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Ejemplo 2: en este ejemplo se describe el atributo especificado de la instantánea especificada.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Salida:

```
Group    UserId
-----    ------
all
```

- Para API obtener más información, consulte DescribeSnapshotAttribute AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SpotDatafeedSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotDatafeedSubscription`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la fuente de datos de su instancia de Spot.

```
Get-EC2SpotDatafeedSubscription
```

Salida:

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para API obtener más información, consulte DescribeSpotDatafeedSubscription AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SpotFleetInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotFleetInstance`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen las instancias asociadas a la solicitud de flota de Spot especificada.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Salida:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Para API obtener más información, consulte DescribeSpotFleetInstances AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SpotFleetRequest`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotFleetRequest`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe la solicitud de flota de Spot especificada.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

Salida:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Ejemplo 2: En este ejemplo se describen todas las solicitudes de flota de Spot.

```
Get-EC2SpotFleetRequest
```

- Para API obtener más información, consulte DescribeSpotFleetRequestsla referencia de AWS Tools for PowerShell cmdlets.

`Get-EC2SpotFleetRequestHistory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotFleetRequestHistory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el historial de la solicitud de flota de Spot especificada.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Salida:

```
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
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeSpotFleetRequestHistory AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SpotInstanceRequest`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotInstanceRequest`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la solicitud de instancia de spot especificada.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen todas las solicitudes de instancias de spot.

```
Get-EC2SpotInstanceRequest
```

- Para API obtener más información, consulte DescribeSpotInstanceRequests AWS Tools for PowerShellCmdlet Reference.

`Get-EC2SpotPriceHistory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2SpotPriceHistory`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen las últimas 10 entradas del historial de precios al contado para el tipo de instancia y la zona de disponibilidad especificados. Tenga en cuenta que el valor especificado para el AvailabilityZone parámetro - debe ser válido para el valor de región proporcionado al parámetro -Region del cmdlet (que no se muestra en el ejemplo) o estar establecido como predeterminado en el shell. Este comando de ejemplo supone que se ha establecido una región predeterminada de 'us-west-2' en el entorno.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Salida:

```
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
...
```

- Para API obtener más información, consulte la referencia del DescribeSpotPriceHistory AWS Tools for PowerShellcmdlet.

`Get-EC2Subnet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Subnet`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la subred especificada.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describen todas las subredes.

```
Get-EC2Subnet
```

- Para API obtener más información, consulte la referencia DescribeSubnetsde AWS Tools for PowerShell cmdlets.

`Get-EC2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Tag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen las etiquetas del tipo de recurso «imagen»

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Salida:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Ejemplo 2: Este ejemplo busca todas las etiquetas de todos los recursos y las agrupa por tipo de recurso

```
Get-EC2Tag | Group-Object resourcetype
```

Salida:

```
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}
```

Ejemplo 3: Este ejemplo muestra todos los recursos con la etiqueta 'eliminación automática' con el valor 'no' para la región dada

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Salida:

```
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
```

Ejemplo 4: Este ejemplo obtiene todos los recursos con la etiqueta «eliminación automática» con el valor «no» y más filtros en el siguiente tubo para analizar solo los tipos de recursos de «instancia» y, finalmente, crea la etiqueta «ThisInstance» para cada recurso de la instancia, cuyo valor es el propio identificador de la instancia

```
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}}
```

Ejemplo 5: Este ejemplo busca las etiquetas de todos los recursos de la instancia, así como las teclas de «Nombre», y las muestra en formato de tabla

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Salida:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Para API obtener más información, consulta DescribeTags AWS Tools for PowerShellCmdlet Reference.

`Get-EC2Volume`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Volume`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el EBS volumen especificado.

```
Get-EC2Volume -VolumeId vol-12345678
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describen EBS los volúmenes que tienen el estado «disponible».

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Salida:

```
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
...
```

Ejemplo 3: En este ejemplo se describen todos los volúmenes. EBS

```
Get-EC2Volume
```

- Para API obtener más información, consulte DescribeVolumes AWS Tools for PowerShellCmdlet Reference.

`Get-EC2VolumeAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VolumeAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el atributo especificado del volumen especificado.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Salida:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Para API obtener más información, consulte DescribeVolumeAttribute AWS Tools for PowerShellCmdlet Reference.

`Get-EC2VolumeStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VolumeStatus`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el estado del volumen especificado.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Salida:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Salida:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Salida:

```
Name                            Status
```

```
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- Para API obtener más información, consulte DescribeVolumeStatus AWS Tools for PowerShellCmdlet Reference.

`Get-EC2Vpc`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2Vpc`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe lo especificadoVPC.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Salida:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Ejemplo 2: En este ejemplo se describe el valor predeterminado VPC (solo puede haber uno por región). Si tu cuenta admite la opción EC2 -Classic en esta región, no existe ningún valor predeterminadoVPC.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Salida:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
```

```
Tags            : {}
VpcId           : vpc-45678901
```

Ejemplo 3: En este ejemplo se describen las VPCs que coinciden con el filtro especificado (es decir, las que tienen un valor CIDR que coincide con el valor «10.0.0.0/16» y están en el estado «disponibles»).

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Ejemplo 4: En este ejemplo se describen todos sus. VPCs

```
Get-EC2Vpc
```

- Para API obtener más información, consulte DescribeVpcs AWS Tools for PowerShellCmdlet Reference.

`Get-EC2VpcAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpcAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el atributo enableDnsSupport ".

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Salida:

```
EnableDnsSupport
----------------
True
```

Ejemplo 2: En este ejemplo se describe el atributo enableDnsHostnames «».

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Salida:

```
EnableDnsHostnames
------------------
True
```

- Para API obtener más información, consulte DescribeVpcAttributela referencia del AWS Tools for PowerShell cmdlet.

`Get-EC2VpcClassicLink`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpcClassicLink`

Herramientas para PowerShell

Ejemplo 1: El ejemplo anterior devuelve todos los VPCs valores con su ClassicLinkEnabled estado para la región

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Salida:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Para API obtener más información, consulte DescribeVpcClassicLinkla referencia del AWS Tools for PowerShell cmdlet.

`Get-EC2VpcClassicLinkDnsSupport`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpcClassicLinkDnsSupport`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el estado de ClassicLink DNS soporte de VPCs la región eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Salida:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Para API obtener más información, consulte la referencia del DescribeVpcClassicLinkDnsSupport AWS Tools for PowerShellcmdlet.

`Get-EC2VpcEndpoint`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpcEndpoint`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describen uno o varios de sus VPC puntos de conexión para la región eu-west-1. A continuación, canaliza el resultado al siguiente comando, que selecciona la VpcEndpointId propiedad y devuelve el VPC ID de la matriz en forma de cadena

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Salida:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Ejemplo 2: Este ejemplo describe todos los puntos finales de vpc de la región eu-west-1 y selecciona VpcEndpointId VpcId, ServiceName y PrivateDnsEnabled propiedades para presentarlos en formato tabular

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Salida:

```
VpcEndpointId          VpcId                ServiceName
 PrivateDnsEnabled
-------------          -----                -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
          True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
          True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
          True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
          True
```

Ejemplo 3: En este ejemplo, se exporta el documento de política del VPC punto final vpce-01a2ab3f4f5cc6f7d a un archivo json

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Para obtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API DescribeVpcEndpoints

`Get-EC2VpcEndpointService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpcEndpointService`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el servicio de EC2 VPC punto final con el filtro indicado, en este caso com.amazonaws.eu-west-1.ecs. Además, también expande la propiedad y muestra los detalles ServiceDetails

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se recuperan todos los servicios de EC2 VPC Endpoint y se devuelve el «ssm» ServiceNames correspondiente

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Salida:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeVpcEndpointServicesReference.

`Get-EC2VpnConnection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpnConnection`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la VPN conexión especificada.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se describe cualquier VPN conexión cuyo estado sea pendiente o disponible.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Ejemplo 3: En este ejemplo se describen todas VPN las conexiones.

```
Get-EC2VpnConnection
```

- Para API obtener más información, consulte DescribeVpnConnections AWS Tools for PowerShellCmdlet Reference.

`Get-EC2VpnGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EC2VpnGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la puerta de enlace privada virtual especificada.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Salida:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
```

```
VpnGatewayId     : vgw-1a2b3c4d
```

Ejemplo 2: Este ejemplo describe cualquier puerta de enlace privada virtual cuyo estado esté pendiente o disponible.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Ejemplo 3: En este ejemplo se describen todas las puertas de enlace privadas virtuales.

```
Get-EC2VpnGateway
```

- Para API obtener más información, consulte la referencia DescribeVpnGatewaysde AWS Tools for PowerShell cmdlets.

`Grant-EC2SecurityGroupEgress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Grant-EC2SecurityGroupEgress`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo define una regla de salida para el grupo de seguridad especificado para EC2 -VPC. La regla concede acceso al intervalo de direcciones IP especificado en el TCP puerto 80. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear el IpPermission objeto.

```
$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
```

Ejemplo 3: Este ejemplo concede acceso al grupo de seguridad de origen especificado en el TCP puerto 80.

```
$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 } )
```

- Para API obtener más información, consulte AuthorizeSecurityGroupEgressla referencia de AWS Tools for PowerShell cmdlets.

`Grant-EC2SecurityGroupIngress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Grant-EC2SecurityGroupIngress`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se definen las reglas de entrada de un grupo de seguridad para EC2 -VPC. Estas reglas otorgan acceso a una dirección IP específica para SSH (puerto 22) y RDC (puerto 3389). Tenga en cuenta que debe identificar los grupos de seguridad, VPC utilizando el ID del grupo de seguridad, no el nombre del grupo de seguridad. EC2 La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
$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 )
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear los IpPermission objetos.

```
$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 )
```

Ejemplo 3: Este ejemplo define las reglas de entrada para un grupo de seguridad de -Classic. EC2 Estas reglas otorgan acceso a una dirección IP específica para SSH (puerto 22) y RDC (puerto 3389). La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o superior.

```
$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 )
```

Ejemplo 4: Con la PowerShell versión 2, debe usar New-Object para crear los IpPermission objetos.

```
$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 )
```

Ejemplo 5: Este ejemplo concede al TCP puerto 8081 acceso desde el grupo de seguridad de origen especificado (sg-1a2b3c4d) al grupo de seguridad especificado (sg-12345678).

```
$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 } )
```

Ejemplo 6: Este ejemplo agrega el CIDR 5.5.5.5/32 a las reglas de entrada del grupo de seguridad sg-1234abcd para el tráfico del puerto 22 con una descripción. TCP

```
$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
```

- Para AuthorizeSecurityGroupIngressobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

`Import-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Import-EC2Image`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo importa una imagen de máquina virtual de disco único desde el bucket de Amazon S3 especificado a Amazon EC2 con un token de idempotencia. El ejemplo requiere que exista un rol de servicio de importación de VM con el nombre predeterminado «vmimport», con una política que permita a Amazon EC2 acceder al depósito especificado, como se explica en el tema Requisitos previos de VM Import. Para usar un rol personalizado, especifique el nombre del rol mediante el parámetro. **`-RoleName`**

```
$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"

$parms = @{
```

```
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Salida:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Para API obtener más información, consulte ImportImagela referencia del AWS Tools for PowerShell cmdlet.

`Import-EC2KeyPair`

En el siguiente ejemplo de código se muestra cómo usarlo. `Import-EC2KeyPair`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se importa una clave pública aEC2. La primera línea almacena el contenido del archivo de clave pública (*.pub) en la variable**`$publickey`**. A continuación, el ejemplo convierte el UTF8 formato del archivo de clave pública en una cadena codificada en Base64 y almacena la cadena convertida en la variable. **`$pkbase64`** En la última línea, se importa la clave pública convertida a. EC2 Como resultado, el cmdlet devuelve la huella digital y el nombre de la clave.

```
$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
```

Salida:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Para API obtener más información, consulte la referencia del ImportKeyPair AWS Tools for PowerShellcmdlet.

`Import-EC2Snapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Import-EC2Snapshot`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo importa una imagen de disco de máquina virtual con el formato 'VMDK' a una EBS instantánea de Amazon. El ejemplo requiere un rol de VM Import Service con el nombre predeterminado «vmimport», con una política que permita a Amazon EC2 acceder al bucket especificado, como se explica en el **`VM Import Prequisites`** tema de http:// docs.aws.amazon.com/ AWSEC2 WindowsGuide /latest//.html. VMImportPrerequisites Para usar un rol personalizado, especifique el nombre del rol mediante el parámetro. **`-RoleName`**

```
$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
```

Salida:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------
```

```
Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para API obtener más información, consulte ImportSnapshotla referencia del AWS Tools for PowerShell cmdlet.

`Move-EC2AddressToVpc`

En el siguiente ejemplo de código se muestra cómo usarlo. `Move-EC2AddressToVpc`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se mueve una EC2 instancia con una dirección IP pública de 12.345.67.89 a la VPC plataforma EC2 - en la región EE.UU. Este (Virginia del Norte).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Ejemplo 2: este ejemplo canaliza los resultados de un Get-EC2Instance comando al cmdlet. Move-EC2AddressToVpc El Get-EC2Instance comando obtiene una instancia que se especifica mediante el ID de la instancia y, a continuación, devuelve la propiedad de dirección IP pública de la instancia.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Para API obtener más información, consulta MoveAddressToVpc AWS Tools for PowerShellCmdlet Reference.

`New-EC2Address`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Address`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asigna una dirección IP elástica para usarla con una instancia en un. VPC

```
New-EC2Address -Domain Vpc
```

Salida:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Ejemplo 2: Este ejemplo asigna una dirección IP elástica para usarla con una instancia en -Classic. EC2

```
New-EC2Address
```

Salida:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Para API obtener más información, consulte Cmdlet AllocateAddressReference AWS Tools for PowerShell .

`New-EC2CustomerGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2CustomerGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea la pasarela de clientes especificada.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Salida:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Para API obtener más información, consulte CreateCustomerGateway AWS Tools for PowerShellCmdlet Reference.

`New-EC2DhcpOption`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2DhcpOption`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea el conjunto de DHCP opciones especificado. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Salida:

```
DhcpConfigurations                  DhcpOptionsId    Tags
------------------                  -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear cada DHCP opción.

```
$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)
```

Salida:

```
DhcpConfigurations                  DhcpOptionsId    Tags
------------------                  -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateDhcpOptionsReference.

`New-EC2FlowLog`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2FlowLog`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un EC2 registro de flujo para la subred-1d234567 en la cloud-watch-log denominada «subnet1-log» para todo el tráfico de «» mediante los permisos de la función «Administrador» REJECT

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Salida:

```
ClientToken                                  FlowLogIds             Unsuccessful
-----------                                  ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Para CreateFlowLogs AWS Tools for PowerShellobtener más información, consulte la referencia de cmdlets. API

`New-EC2Host`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Host`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se asigna un host dedicado a su cuenta para el tipo de instancia y la zona de disponibilidad determinados

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

Salida:

```
h-01e23f4cd567890f3
```

- Para API obtener más información, consulte la referencia de AllocateHosts AWS Tools for PowerShellcmdlets.

`New-EC2HostReservation`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2HostReservation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo compra la oferta de reservas hro-0c1f23456789d0ab con configuraciones que coinciden con las de su host dedicado h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Salida:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Para PurchaseHostReservation AWS Tools for PowerShellobtener más información, consulte Cmdlet Reference. API

`New-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Image`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una AMI con el nombre y la descripción especificados, a partir de la instancia especificada. Amazon EC2 intenta cerrar la instancia de forma limpia antes de crear la imagen y la reinicia al finalizar.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Ejemplo 2: en este ejemplo se crea una AMI con el nombre y la descripción especificados, a partir de la instancia especificada. Amazon EC2 crea la imagen sin cerrar ni reiniciar la instancia; por lo tanto, no se puede garantizar la integridad del sistema de archivos de la imagen creada.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Ejemplo 3: En este ejemplo se crea una AMI con tres volúmenes. El primer volumen se basa en una EBS instantánea de Amazon. El segundo volumen es un volumen Amazon EBS vacío de 100 GiB. El tercer volumen es un volumen de almacén de instancias. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
$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"})
```

- Para API obtener más información, consulte CreateImage AWS Tools for PowerShellCmdlet Reference.

`New-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo lanza una sola instancia de lo especificado AMI en EC2 -Classic o una instancia predeterminadaVPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Ejemplo 2: Este ejemplo lanza una única instancia de lo especificado AMI en unVPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Ejemplo 3: Para añadir un EBS volumen o un volumen de almacén de instancias, defina un mapeo de dispositivos de bloques y agréguelo al comando. En este ejemplo, se agrega un volumen de almacén de instancias.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Ejemplo 4: Para especificar uno de los Windows actualesAMIs, obtenga su AMI ID utilizandoGet-EC2ImageByName. En este ejemplo, se lanza una instancia desde la base actual AMI de Windows Server 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Ejemplo 5: lanza una instancia en el entorno de host dedicado especificado.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Ejemplo 6: Esta solicitud lanza dos instancias y aplica a las instancias una etiqueta con una clave del servidor web y un valor de producción. La solicitud también aplica una etiqueta con la clave cost-center y un valor de cc123 a los volúmenes que se crean (en este caso, el volumen raíz de cada instancia).

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet RunInstancesReference.

`New-EC2InstanceExportTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2InstanceExportTask`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se exporta una instancia detenida**`i-0800b00a00EXAMPLE`**, como un disco duro virtual (VHD) al bucket de S3**`testbucket-export-instances-2019`**. El entorno de destino es**`Microsoft`**, y el parámetro region se añade porque la instancia está en la **`us-east-1`** región, mientras que la AWS región predeterminada del usuario no es us-east-1. Para obtener el estado de la tarea de exportación, copia el **`ExportTaskId`** valor de los resultados de este comando y ejecuta **`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
```

Salida:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Para API obtener más información, consulte CreateInstanceExportTask AWS Tools for PowerShellCmdlet Reference.

`New-EC2InternetGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2InternetGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una puerta de enlace a Internet.

```
New-EC2InternetGateway
```

Salida:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Para API obtener más información, consulte CreateInternetGateway AWS Tools for PowerShellCmdlet Reference.

`New-EC2KeyPair`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2KeyPair`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un key pair y captura la clave RSA privada PEM codificada en un archivo con el nombre especificado. Cuando lo utilice PowerShell, la codificación debe estar establecida en ascii para generar una clave válida. Para obtener más información, consulte Crear, mostrar y eliminar pares de EC2 claves de Amazon (https://docs.aws.amazon.com/cli/latest/userguide/ cli-services-ec 2-keypairs.html) en la Guía del usuario de la interfaz de línea de AWS comandos.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Para obtener API más información, consulte la referencia de cmdlets. CreateKeyPairAWS Tools for PowerShell

`New-EC2NetworkAcl`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2NetworkAcl`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una red ACL para lo especificadoVPC.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Salida:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Para API obtener más información, consulte CreateNetworkAcl AWS Tools for PowerShellCmdlet Reference.

`New-EC2NetworkAclEntry`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2NetworkAclEntry`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una entrada para la red especificadaACL. La regla permite el tráfico entrante desde cualquier lugar (0.0.0.0/0) del UDP puerto 53 (DNS) hacia cualquier subred asociada.

```
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
```

- Para obtener API más información, consulte la referencia de cmdlets. CreateNetworkAclEntryAWS Tools for PowerShell

`New-EC2NetworkInterface`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2NetworkInterface`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea la interfaz de red especificada.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateNetworkInterface AWS Tools for PowerShellCmdlet Reference.

`New-EC2PlacementGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2PlacementGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un grupo de ubicación con el nombre especificado.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Para API obtener más información, consulte CreatePlacementGroup AWS Tools for PowerShellCmdlet Reference.

`New-EC2Route`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Route`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea la ruta especificada para la tabla de rutas especificada. La ruta coincide con todo el tráfico y lo envía a la puerta de enlace de Internet especificada.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId
 igw-1a2b3c4d
```

Salida:

```
True
```

- Para API obtener más información, consulte CreateRoutela referencia de AWS Tools for PowerShell cmdlets.

`New-EC2RouteTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2RouteTable`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una tabla de rutas para lo especificadoVPC.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Salida:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Para API obtener más información, consulte CreateRouteTable AWS Tools for PowerShellCmdlet Reference.

`New-EC2ScheduledInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2ScheduledInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se lanza la instancia programada especificada.

```
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
```

- Para API obtener más información, consulte RunScheduledInstances AWS Tools for PowerShellCmdlet Reference.

`New-EC2ScheduledInstancePurchase`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2ScheduledInstancePurchase`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se compra una instancia programada.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Salida:

```
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
```

- Para API obtener más información, consulte PurchaseScheduledInstances AWS Tools for PowerShellCmdlet Reference.

`New-EC2SecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2SecurityGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un grupo de seguridad para el especificadoVPC.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

Salida:

```
sg-12345678
```

Ejemplo 2: En este ejemplo se crea un grupo de seguridad para EC2 -Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Salida:

```
sg-45678901
```

- Para API obtener más información, consulte CreateSecurityGroup AWS Tools for PowerShellCmdlet Reference.

`New-EC2Snapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Snapshot`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una instantánea del volumen especificado.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateSnapshot AWS Tools for PowerShellCmdlet Reference.

`New-EC2SpotDatafeedSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2SpotDatafeedSubscription`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una fuente de datos de instancias puntuales.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Salida:

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para API obtener más información, consulte CreateSpotDatafeedSubscription AWS Tools for PowerShellCmdlet Reference.

`New-EC2Subnet`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Subnet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una subred con lo especificadoCIDR.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateSubnet AWS Tools for PowerShellCmdlet Reference.

`New-EC2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Tag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega una sola etiqueta al recurso especificado. La clave de la etiqueta es 'myTag' y el valor de la etiqueta es 'myTagValue'. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Ejemplo 2: Este ejemplo actualiza o agrega las etiquetas especificadas al recurso especificado. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Ejemplo 3: Con la PowerShell versión 2, debe usar New-Object para crear la etiqueta para el parámetro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateTagsReference.

`New-EC2Volume`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Volume`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea el volumen especificado.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Salida:

```
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
```

Ejemplo 2: Esta solicitud de ejemplo crea un volumen y aplica una etiqueta con una clave de pila y un valor de producción.

```
$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
```

- Para API obtener más información, consulte CreateVolume AWS Tools for PowerShellCmdlet Reference.

`New-EC2Vpc`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2Vpc`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un VPC con el especificadoCIDR. Amazon VPC también crea lo siguiente paraVPC: un conjunto de DHCP opciones predeterminado, una tabla de rutas principal y una red predeterminadaACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Salida:

```
CidrBlock       : 10.0.0.0/16
```

```
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Para API obtener más información, consulte CreateVpcla referencia de AWS Tools for PowerShell cmdlets.

`New-EC2VpcEndpoint`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2VpcEndpoint`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo VPC punto final para el servicio com.amazonaws.eu-west-1.s3 en el vpc-0fc1ff23f45b678eb VPC

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Salida:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Para obtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API CreateVpcEndpoint

`New-EC2VpnConnection`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2VpnConnection`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una VPN conexión entre la puerta de enlace privada virtual especificada y la puerta de enlace del cliente especificada. El resultado incluye la información de configuración que necesita el administrador de red, en XML formato.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Salida:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Ejemplo 2: en este ejemplo se crea la VPN conexión y se captura la configuración en un archivo con el nombre especificado.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Ejemplo 3: Este ejemplo crea una VPN conexión, con enrutamiento estático, entre la puerta de enlace privada virtual especificada y la puerta de enlace del cliente especificada.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Para API obtener más información, consulte CreateVpnConnectionla referencia de AWS Tools for PowerShell cmdlets.

`New-EC2VpnConnectionRoute`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2VpnConnectionRoute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea la ruta estática especificada para la VPN conexión especificada.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Para API obtener más información, consulte CreateVpnConnectionRoute AWS Tools for PowerShellCmdlet Reference.

`New-EC2VpnGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EC2VpnGateway`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea la puerta de enlace privada virtual especificada.

```
New-EC2VpnGateway -Type ipsec.1
```

Salida:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Para API obtener más información, consulte CreateVpnGateway AWS Tools for PowerShellCmdlet Reference.

`Register-EC2Address`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-EC2Address`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia la dirección IP elástica especificada a la instancia especificada en unVPC.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Salida:

```
eipassoc-12345678
```

Ejemplo 2: Este ejemplo asocia la dirección IP elástica especificada a la instancia especificada en EC2 -Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Para API obtener más información, consulte AssociateAddress AWS Tools for PowerShellCmdlet Reference.

`Register-EC2DhcpOption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-EC2DhcpOption`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia el conjunto de DHCP opciones especificado al especificadoVPC.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Ejemplo 2: Este ejemplo asocia el conjunto de DHCP opciones predeterminado al especificadoVPC.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Para API obtener más información, consulte AssociateDhcpOptions AWS Tools for PowerShellCmdlet Reference.

`Register-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-EC2Image`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra y AMI utiliza el archivo de manifiesto especificado en Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Para API obtener más información, consulte RegisterImagela referencia de AWS Tools for PowerShell cmdlets.

`Register-EC2PrivateIpAddress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-EC2PrivateIpAddress`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asigna la dirección IP privada secundaria especificada a la interfaz de red especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Ejemplo 2: este ejemplo crea dos direcciones IP privadas secundarias y las asigna a la interfaz de red especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Para API obtener más información, consulte la referencia AssignPrivateIpAddressesde AWS Tools for PowerShell cmdlets.

`Register-EC2RouteTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-EC2RouteTable`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia la tabla de rutas especificada a la subred especificada.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Salida:

```
rtbassoc-12345678
```

- Para API obtener más información, consulte AssociateRouteTable AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2Address`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Address`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se publica la dirección IP elástica especificada para las instancias de unVPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Ejemplo 2: En este ejemplo, se publica la dirección IP elástica especificada para las instancias de EC2 -Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Para API obtener más información, consulte ReleaseAddress AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2CapacityReservation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2CapacityReservation`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cancela la reserva de capacidad cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Salida:

```
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
```

- APIPara obtener CancelCapacityReservationmás AWS Tools for PowerShell información, consulte la referencia del cmdlet.

`Remove-EC2CustomerGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2CustomerGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la pasarela de clientes especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteCustomerGateway AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2DhcpOption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2DhcpOption`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el conjunto de DHCP opciones especificado. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteDhcpOptions AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2FlowLog`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2FlowLog`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el FlowLogId fl-01a2b3456a789c01 dado

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Salida:

```
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
```

- APIPara obtener más DeleteFlowLogs AWS Tools for PowerShellinformación, consulte la referencia del cmdlet.

`Remove-EC2Host`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Host`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se publica el ID de host indicado h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Salida:

```
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} {}
```

- APIPara obtener ReleaseHostsmás AWS Tools for PowerShell información, consulte la referencia del cmdlet.

`Remove-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo finaliza la instancia especificada (la instancia puede estar en ejecución o en estado «detenida»). El cmdlet solicitará una confirmación antes de continuar; utilice el modificador -Force para suprimir la solicitud.

```
Remove-EC2Instance -InstanceId i-12345678
```

Salida:

```
CurrentState                    InstanceId    PreviousState
------------                    ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para API obtener más información, consulte Cmdlet TerminateInstancesReference AWS Tools for PowerShell .

`Remove-EC2InternetGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2InternetGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la puerta de enlace de Internet especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteInternetGateway AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2KeyPair`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2KeyPair`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el key pair especificado. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteKeyPair AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2NetworkAcl`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2NetworkAcl`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la red ACL especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteNetworkAcl AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2NetworkAclEntry`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2NetworkAclEntry`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la regla especificada de la red especificadaACL. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteNetworkAclEntry AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2NetworkInterface`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2NetworkInterface`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la interfaz de red especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteNetworkInterface AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2PlacementGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2PlacementGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el grupo de ubicación especificado. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeletePlacementGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2Route`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Route`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la ruta especificada de la tabla de rutas especificada. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteRoute AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2RouteTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2RouteTable`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la tabla de rutas especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteRouteTable AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2SecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2SecurityGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el grupo de seguridad especificado para EC2 -VPC. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Salida:

```
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"):
```

Ejemplo 2: en este ejemplo se elimina el grupo de seguridad especificado para EC2 -Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteSecurityGroupReference.

`Remove-EC2Snapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Snapshot`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la instantánea especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteSnapshot AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2SpotDatafeedSubscription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2SpotDatafeedSubscription`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la fuente de datos de la instancia de Spot. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2SpotDatafeedSubscription
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteSpotDatafeedSubscription AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2Subnet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Subnet`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la subred especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteSubnet AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Tag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta especificada del recurso especificado, independientemente del valor de la etiqueta. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Ejemplo 2: en este ejemplo se elimina la etiqueta especificada del recurso especificado, pero solo si el valor de la etiqueta coincide. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Ejemplo 3: en este ejemplo se elimina la etiqueta especificada del recurso especificado, independientemente del valor de la etiqueta.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Ejemplo 4: en este ejemplo se elimina la etiqueta especificada del recurso especificado, pero solo si el valor de la etiqueta coincide.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Para API obtener más información, consulte la referencia DeleteTagsdel AWS Tools for PowerShell cmdlet.

`Remove-EC2Volume`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Volume`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se separa el volumen especificado. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteVolume AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2Vpc`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2Vpc`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina lo especificadoVPC. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteVpc AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2VpnConnection`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2VpnConnection`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la VPN conexión especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteVpnConnection AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2VpnConnectionRoute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2VpnConnectionRoute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la ruta estática especificada de la VPN conexión especificada. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteVpnConnectionRoute AWS Tools for PowerShellCmdlet Reference.

`Remove-EC2VpnGateway`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EC2VpnGateway`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la puerta de enlace privada virtual especificada. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteVpnGateway AWS Tools for PowerShellCmdlet Reference.

`Request-EC2SpotFleet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Request-EC2SpotFleet`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se crea una solicitud de flota puntual en la zona de disponibilidad con el precio más bajo para el tipo de instancia especificado. Si su cuenta VPC solo admite EC2 (), la flota de Spot lanza las instancias en la zona de disponibilidad más económica que tenga una subred predeterminada. Si su cuenta admite EC2 -Classic, la flota de Spot lanza las instancias en EC2 -Classic en la zona de disponibilidad con el precio más bajo. Tenga en cuenta que el precio que pague no superará el precio puntual especificado para la solicitud.

```
$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
```

- Para API obtener más información, consulte RequestSpotFleetla referencia del AWS Tools for PowerShell cmdlet.

`Request-EC2SpotInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Request-EC2SpotInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se solicita una instancia de spot única en la subred especificada. Tenga en cuenta que el grupo de seguridad debe crearse para el grupo VPC que contiene la subred especificada y debe especificarse mediante un ID mediante la interfaz de red. Al especificar una interfaz de red, debe incluir el ID de subred mediante la interfaz de red.

```
$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
```

Salida:

```
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
```

- Para API obtener más información, consulte la Referencia RequestSpotInstancesde AWS Tools for PowerShell cmdlets.

`Reset-EC2ImageAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-EC2ImageAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se restablece el atributo launchPermission 'a su valor predeterminado. De forma predeterminada, AMIs son privados.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Para API obtener más información, consulte ResetImageAttribute AWS Tools for PowerShellCmdlet Reference.

`Reset-EC2InstanceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-EC2InstanceAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se restablece el atributo sriovNetSupport '' de la instancia especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ejemplo 2: en este ejemplo se restablece el atributo 'ebsOptimized' de la instancia especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ejemplo 3: en este ejemplo se restablece el atributo 'sourceDestCheck' de la instancia especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Ejemplo 4: en este ejemplo se restablece el atributo 'disableApiTermination' de la instancia especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ejemplo 5: en este ejemplo se restablece el atributo 'instanceInitiatedShutdownBehavior' de la instancia especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Para API obtener más información, consulte la referencia del ResetInstanceAttributecmdlet AWS Tools for PowerShell .

`Reset-EC2NetworkInterfaceAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-EC2NetworkInterfaceAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se restablece la comprobación de origen y destino de la interfaz de red especificada.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Para API obtener más información, consulte la referencia de ResetNetworkInterfaceAttributecmdlets AWS Tools for PowerShell .

`Reset-EC2SnapshotAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-EC2SnapshotAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se restablece el atributo especificado de la instantánea especificada.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Para API obtener más información, consulte ResetSnapshotAttribute AWS Tools for PowerShellCmdlet Reference.

`Restart-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Restart-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se reinicia la instancia especificada.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Para API obtener más información, consulte RebootInstances AWS Tools for PowerShellCmdlet Reference.

`Revoke-EC2SecurityGroupEgress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Revoke-EC2SecurityGroupEgress`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la regla del grupo de seguridad especificado para EC2 -VPC. Esto revoca el acceso al intervalo de direcciones IP especificado en el TCP puerto 80. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear el IpPermission objeto.

```
$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
```

Ejemplo 3: Este ejemplo revoca el acceso al grupo de seguridad de origen especificado en TCP el puerto 80.

```
$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 } )
```

- Para API obtener más información, consulte la referencia RevokeSecurityGroupEgressde AWS Tools for PowerShell cmdlets.

`Revoke-EC2SecurityGroupIngress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Revoke-EC2SecurityGroupIngress`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo revoca el acceso al TCP puerto 22 desde el rango de direcciones especificado para el grupo de seguridad especificado para EC2 -VPC. Tenga en cuenta que debe identificar los grupos de seguridad paraEC2, VPC utilizando el ID del grupo de seguridad, no

el nombre del grupo de seguridad. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear el IpPermission objeto.

```
$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
```

Ejemplo 3: Este ejemplo revoca el acceso al TCP puerto 22 desde el rango de direcciones especificado para el grupo de seguridad especificado para -Classic. EC2 La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Ejemplo 4: Con la PowerShell versión 2, debe usar New-Object para crear el IpPermission objeto.

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet RevokeSecurityGroupIngressReference.

`Send-EC2InstanceStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Send-EC2InstanceStatus`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo informa sobre el estado de la instancia especificada.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Para API obtener más información, consulte ReportInstanceStatusla referencia del AWS Tools for PowerShell cmdlet.

`Set-EC2NetworkAclAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-EC2NetworkAclAssociation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia la red ACL especificada a la subred de la ACL asociación de red especificada.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Salida:

```
aclassoc-87654321
```

- Para API obtener más información, consulte la referencia ReplaceNetworkAclAssociationde AWS Tools for PowerShell cmdlets.

`Set-EC2NetworkAclEntry`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-EC2NetworkAclEntry`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo reemplaza la entrada especificada para la red especificadaACL. La nueva regla permite el tráfico entrante desde la dirección especificada a cualquier subred asociada.

```
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
```

- Para API obtener más información, consulte la referencia de ReplaceNetworkAclEntry AWS Tools for PowerShellcmdlets.

`Set-EC2Route`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-EC2Route`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo reemplaza la ruta especificada por la tabla de rutas especificada. La nueva ruta envía el tráfico especificado a la puerta de enlace privada virtual especificada.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- Para API obtener más información, consulte ReplaceRoutela referencia de AWS Tools for PowerShell cmdlets.

`Set-EC2RouteTableAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-EC2RouteTableAssociation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia la tabla de rutas especificada a la subred de la asociación de tabla de rutas especificada.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Salida:

```
rtbassoc-87654321
```

- Para API obtener más información, consulte ReplaceRouteTableAssociation AWS Tools for PowerShellCmdlet Reference.

`Start-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se inicia la instancia especificada.

```
Start-EC2Instance -InstanceId i-12345678
```

Salida:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Ejemplo 2: en este ejemplo se inician las instancias especificadas.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Ejemplo 3: En este ejemplo se inicia el conjunto de instancias que están detenidas actualmente. Los objetos de instancia devueltos por Get-EC2Instance se canalizan aStart-EC2Instance. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o superior.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

Ejemplo 4: Con la PowerShell versión 2, debe usar New-Object para crear el filtro para el parámetro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet StartInstancesReference.

`Start-EC2InstanceMonitoring`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-EC2InstanceMonitoring`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite una supervisión detallada de la instancia especificada.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Salida:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para API obtener más información, consulte MonitorInstances AWS Tools for PowerShellCmdlet Reference.

`Stop-EC2ImportTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-EC2ImportTask`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cancela la tarea de importación especificada (importación de instantáneas o imágenes). Si es necesario, se puede proporcionar un motivo mediante el **`-CancelReason`** parámetro.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Para API obtener más información, consulte CancelImportTaskla referencia del AWS Tools for PowerShell cmdlet.

`Stop-EC2Instance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-EC2Instance`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo detiene la instancia especificada.

```
Stop-EC2Instance -InstanceId i-12345678
```

Salida:

```
CurrentState                    InstanceId    PreviousState
------------                    ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para API obtener más información, consulte StopInstances AWS Tools for PowerShellCmdlet Reference.

`Stop-EC2InstanceMonitoring`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-EC2InstanceMonitoring`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo deshabilita la supervisión detallada de la instancia especificada.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Salida:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para API obtener más información, consulte UnmonitorInstances AWS Tools for PowerShellCmdlet Reference.

`Stop-EC2SpotFleetRequest`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-EC2SpotFleetRequest`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cancela la solicitud de flota de Spot especificada y se cancelan las instancias de Spot asociadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Ejemplo 2: Este ejemplo cancela la solicitud de flota de Spot especificada sin cancelar las instancias de Spot asociadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CancelSpotFleetRequestsReference.

`Stop-EC2SpotInstanceRequest`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-EC2SpotInstanceRequest`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cancela la solicitud de instancia de spot especificada.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Salida:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- Para API obtener más información, consulte CancelSpotInstanceRequests AWS Tools for PowerShellCmdlet Reference.

`Unregister-EC2Address`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-EC2Address`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo desasocia la dirección IP elástica especificada de la instancia especificada en unVPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Ejemplo 2: Este ejemplo desasocia la dirección IP elástica especificada de la instancia especificada en EC2 -Classic.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DisassociateAddressReference.

`Unregister-EC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-EC2Image`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se anula el registro de lo especificado. AMI

```
Unregister-EC2Image -ImageId ami-12345678
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeregisterImageReference.

`Unregister-EC2PrivateIpAddress`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-EC2PrivateIpAddress`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo anula la asignación de la dirección IP privada especificada de la interfaz de red especificada.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Para API obtener más información, consulte la referencia de UnassignPrivateIpAddresses AWS Tools for PowerShellcmdlets.

`Unregister-EC2RouteTable`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-EC2RouteTable`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la asociación especificada entre una tabla de enrutamiento y una subred.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Para API obtener más información, consulte DisassociateRouteTable AWS Tools for PowerShellCmdlet Reference.

`Update-EC2SecurityGroupRuleIngressDescription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-EC2SecurityGroupRuleIngressDescription`

Herramientas para PowerShell

Ejemplo 1: actualiza la descripción de una regla de grupo de seguridad de entrada (entrante) existente.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

Ejemplo 2: elimina la descripción de una regla de grupo de seguridad de entrada (entrante) existente (omitiendo el parámetro de la solicitud).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
```

```
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithoutDescription
```

- Para API obtener más información, consulte Cmdlet UpdateSecurityGroupRuleDescriptionsIngressReference AWS Tools for PowerShell .

ECREjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonECR.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-ECRLoginCommand`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-ECRLoginCommand`.

Herramientas para PowerShell

Ejemplo 1: Devuelve una información PSObject de inicio de sesión que se puede utilizar para autenticarse en cualquier ECR registro de Amazon al que tenga acceso su IAM director. Las credenciales y el punto final de la región necesarios para que la llamada obtenga el token de autorización se obtienen de los valores predeterminados del shell (configurados por los **`Initialize-AWSDefaultConfiguration`** cmdlets **`Set-AWSCredential/Set-DefaultAWSRegion`** o). Puede usar la propiedad Command con Invoke-Expression para iniciar

sesión en el registro especificado o usar las credenciales devueltas en otras herramientas que requieran iniciar sesión.

```
Get-ECRLoginCommand
```

Salida:

```
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
```

Ejemplo 2: recupera una información de inicio PSObject de sesión que se utiliza como entrada para un comando de inicio de sesión de docker. Puedes especificar cualquier ECR registro de Amazon en el URI que autenticarte siempre que tu IAM director tenga acceso a ese registro.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Get- ECRLoginCommand in.

ECSEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonECS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-ECSClusterDetail`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-ECSClusterDetail`.

Herramientas para PowerShell

Ejemplo 1: este cmdlet describe uno o más de sus ECS clústeres.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Salida:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Para API obtener más información, consulte la referencia del DescribeClusters AWS Tools for PowerShellcmdlet.

`Get-ECSClusterList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ECSClusterList`

Herramientas para PowerShell

Ejemplo 1: este cmdlet devuelve una lista de los clústeres existentesECS.

```
Get-ECSClusterList
```

Salida:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Para API obtener más información, consulte la referencia del ListClusters AWS Tools for PowerShellcmdlet.

`Get-ECSClusterService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ECSClusterService`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los servicios que se ejecutan en el clúster predeterminado.

```
Get-ECSClusterService
```

Ejemplo 2: En este ejemplo se enumeran todos los servicios que se ejecutan en el clúster especificado.

```
Get-ECSClusterService -Cluster myCluster
```

Ejemplo 3: En este ejemplo se enumeran los servicios que se ejecutan en el clúster especificado y se obtienen un máximo de 10 detalles de servicio a la vez.

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte ListServices AWS Tools for PowerShellCmdlet Reference.

`Get-ECSService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ECSService`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo recuperar los detalles de un servicio específico del clúster predeterminado.

```
Get-ECSService -Service my-hhtp-service
```

Ejemplo 2: en este ejemplo se muestra cómo recuperar los detalles de un servicio específico que se ejecuta en el clúster mencionado.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Para API obtener más información, consulte DescribeServicesla referencia de AWS Tools for PowerShell cmdlets.

`New-ECSCluster`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ECSCluster`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet crea un nuevo clúster de AmazonECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Salida:

```
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                              : {}
```

- Para API obtener más información, consulte la referencia del CreateCluster AWS Tools for PowerShellcmdlet.

`New-ECSService`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ECSService`

Herramientas para PowerShell

Ejemplo 1: Este comando de ejemplo crea un servicio en el clúster predeterminado llamado `ecs-simple-service`. El servicio usa la definición de tarea `ecs-demo` y mantiene 10 instancias de esa tarea.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Ejemplo 2: Este comando de ejemplo crea un servicio detrás de un balanceador de cargas en tu clúster predeterminado llamado ``. ecs-simple-service El servicio usa la definición de tarea `ecs-demo` y mantiene 10 instancias de esa tarea.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Para obtener API más información, consulte la referencia del cmdlet. CreateServiceAWS Tools for PowerShell

`Remove-ECSCluster`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ECSCluster`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina el clúster especificado. ECS Debe anular el registro de todas las instancias de contenedor de este clúster para poder eliminarlo.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia de DeleteCluster AWS Tools for PowerShellcmdlets.

`Remove-ECSService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ECSService`

Herramientas para PowerShell

Ejemplo 1: Elimina el servicio denominado 'my-http-service' del clúster predeterminado. El servicio debe tener el recuento deseado y el recuento en ejecución igual a 0 para poder eliminarlo.
Se le solicitará que lo confirme antes de continuar con el comando. Para omitir el mensaje de confirmación, añada el conmutador -Force.

```
Remove-ECSService -Service my-http-service
```

Ejemplo 2: elimina el servicio denominado 'my-http-service' en el clúster nombrado.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Para API obtener más información, consulte DeleteService AWS Tools for PowerShellCmdlet Reference.

`Update-ECSClusterSetting`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-ECSClusterSetting`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet modifica la configuración para utilizarla en un clúster. ECS

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

Salida:

```
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                              : {}
```

- Para API obtener más información, consulte la referencia del UpdateClusterSettingscmdlet AWS Tools for PowerShell .

`Update-ECSService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-ECSService`

Herramientas para PowerShell

Ejemplo 1: Este comando de ejemplo actualiza el servicio my-http-service `` para usar la definición de tarea amazon-ecs-sample ``.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Ejemplo 2: Este comando de ejemplo actualiza el recuento deseado del servicio my-http-service `` a 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Para API obtener más información, consulte UpdateServicela referencia del AWS Tools for PowerShell cmdlet.

EFSEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonEFS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Edit-EFSMountTargetSecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo`Edit-EFSMountTargetSecurityGroup`.

Herramientas para PowerShell

Ejemplo 1: actualiza los grupos de seguridad vigentes para el destino de montaje especificado. Se pueden especificar hasta 5, en el formato «sg-xxxxxxxx».

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- Para obtener API más información, consulte ModifyMountTargetSecurityGroupsla Referencia de cmdlets.AWS Tools for PowerShell

`Get-EFSFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EFSFileSystem`

Ejemplo 1: Devuelve la colección de todos los sistemas de archivos propiedad de la cuenta de la persona que llama en la región.

```
Get-EFSFileSystem
```

Salida:

```
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
...
```

Ejemplo 2: devuelve los detalles del sistema de archivos especificado.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Ejemplo 3: Devuelve los detalles de un sistema de archivos mediante el token de creación de idempotencia que se especificó en el momento de crear el sistema de archivos.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeFileSystemsReference.

`Get-EFSMountTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EFSMountTarget`

Herramientas para PowerShell

Ejemplo 1: Devuelve la colección de objetivos de montaje asociados al sistema de archivos especificado.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Salida:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Para API obtener más información, consulte DescribeMountTargets AWS Tools for PowerShellCmdlet Reference.

`Get-EFSMountTargetSecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EFSMountTargetSecurityGroup`

Herramientas para PowerShell

Ejemplo 1: Devuelve los identificadores de los grupos de seguridad actualmente asignados a la interfaz de red asociada al destino de montaje.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Salida:

```
sg-1a2b3c4d
```

- Para API obtener más información, consulte DescribeMountTargetSecurityGroupsla referencia de AWS Tools for PowerShell cmdlets.

`Get-EFSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EFSTag`

Herramientas para PowerShell

Ejemplo 1: Devuelve el conjunto de etiquetas actualmente asociadas al sistema de archivos especificado.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Salida:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Para API obtener más información, consulte DescribeTags AWS Tools for PowerShellCmdlet Reference.

`New-EFSFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EFSFileSystem`

Herramientas para PowerShell

Ejemplo 1: Crea un nuevo sistema de archivos vacío. El token utilizado para garantizar la creación idempotente se generará automáticamente y se podrá acceder a él desde el **`CreationToken`** elemento del objeto devuelto.

```
New-EFSFileSystem
```

Salida:

```
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
```

Ejemplo 2: Crea un nuevo sistema de archivos vacío mediante un token personalizado para garantizar la creación idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateFileSystemReference.

`New-EFSMountTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EFSMountTarget`

Herramientas para PowerShell

Ejemplo 1: Crea un nuevo destino de montaje para un sistema de archivos. La subred especificada se utilizará para determinar la nube privada virtual (VPC) en la que se creará el destino de montaje y la dirección IP que se asignará automáticamente (del rango de direcciones de la subred). La dirección IP asignada se puede utilizar para montar este sistema de archivos en una EC2 instancia de Amazon. Como no se especificó ningún grupo de seguridad, la interfaz de red creada para el destino se asocia al grupo de seguridad predeterminado de VPC la subred.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Salida:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Ejemplo 2: crea un nuevo destino de montaje para el sistema de archivos especificado con una dirección IP asignada automáticamente. La interfaz de red creada para el destino de montaje está

asociada a los grupos de seguridad especificados (se pueden especificar hasta 5, en el formato «sg-xxxxxxxx»).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Ejemplo 3: crea un nuevo destino de montaje para el sistema de archivos especificado con la dirección IP especificada.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Para API obtener más información, consulte CreateMountTargetla referencia del AWS Tools for PowerShell cmdlet.

`New-EFSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EFSTag`

Herramientas para PowerShell

Ejemplo 1: Aplica la colección de etiquetas al sistema de archivos especificado. Si ya existe una etiqueta con la clave especificada en el sistema de archivos, se actualiza el valor de la etiqueta.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Ejemplo 2: Establece la etiqueta de nombre para el sistema de archivos especificado. Este valor se devuelve junto con otros detalles del sistema de archivos cuando se utiliza el EFSFileSystem cmdlet Get-.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Para API obtener más información, consulte la referencia del CreateTags AWS Tools for PowerShellcmdlet.

`Remove-EFSFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EFSFileSystem`

Herramientas para PowerShell

Ejemplo 1: Elimina el sistema de archivos especificado que ya no se utiliza (si el sistema de archivos tiene destinos de montaje, primero hay que eliminarlos). Se le solicitará una confirmación antes de continuar con el cmdlet; para suprimir la confirmación, utilice el conmutador. **`-Force`**

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Para API obtener más información, consulte la referencia del DeleteFileSystem AWS Tools for PowerShellcmdlet.

`Remove-EFSMountTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EFSMountTarget`

Herramientas para PowerShell

Ejemplo 1: Elimina el objetivo de montaje especificado. Se le solicitará una confirmación antes de continuar con la operación. Para suprimir el mensaje, utilice el **`-Force`** conmutador. Tenga en cuenta que esta operación interrumpe por la fuerza cualquier montaje del sistema de archivos a través del destino; si es posible, puede considerar la posibilidad de desmontar el sistema de archivos antes de ejecutar este comando.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Para obtener API más información, consulte DeleteMountTargetla Referencia de cmdlets.AWS Tools for PowerShell

`Remove-EFSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EFSTag`

Herramientas para PowerShell

Ejemplo 1: Elimina la colección de una o más etiquetas de un sistema de archivos. Se le solicitará la confirmación antes de continuar con el cmdlet; para suprimir la confirmación, utilice el conmutador. **`-Force`**

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Para API obtener más información, consulte la referencia del DeleteTags AWS Tools for PowerShellcmdlet.

EKSEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonEKS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-EKSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-EKSResourceTag`.

Herramientas para PowerShell

Ejemplo 1: este cmdlet asocia las etiquetas especificadas a un recurso con el especificado. resourceArn

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- Para API obtener más información, consulte la referencia del TagResource AWS Tools for PowerShellcmdlet.

`Get-EKSCluster`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSCluster`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet devuelve información descriptiva sobre un clúster de AmazonEKS.

```
Get-EKSCluster -Name "PROD"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DescribeCluster AWS Tools for PowerShellcmdlet.

`Get-EKSClusterList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSClusterList`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet muestra los EKS clústeres de Amazon de la Cuenta de AWS región especificada.

```
Get-EKSClusterList
```

Salida:

```
PROD
```

- Para API obtener más información, consulte la referencia de ListClusters AWS Tools for PowerShellcmdlets.

`Get-EKSFargateProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSFargateProfile`

Herramientas para PowerShell

Ejemplo 1: este cmdlet devuelve información descriptiva sobre un perfil de Fargate AWS .

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Salida:

```
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                : {}
```

- Para API obtener más información, consulte la referencia del DescribeFargateProfilecmdlet AWS Tools for PowerShell .

`Get-EKSFargateProfileList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSFargateProfileList`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet muestra los perfiles de AWS Fargate asociados al clúster especificado en la región especificada. Cuenta de AWS

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Salida:

```
EKSFargate
EKSFargateProfile
```

- Para API obtener más información, consulte la referencia del ListFargateProfilescmdlet AWS Tools for PowerShell .

`Get-EKSNodegroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSNodegroup`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet devuelve información descriptiva sobre un grupo de EKS nodos de Amazon.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DescribeNodegroup AWS Tools for PowerShellcmdlet.

`Get-EKSNodegroupList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSNodegroupList`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet muestra los grupos de EKS nodos de Amazon asociados al clúster especificado Cuenta de AWS en la región especificada.

```
Get-EKSNodegroupList -ClusterName PROD
```

Salida:

```
ProdEKSNodeGroup
```

- Para API obtener más información, consulte la referencia del ListNodegroups AWS Tools for PowerShellcmdlet.

`Get-EKSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSResourceTag`

Herramientas para PowerShell

Ejemplo 1: este cmdlet muestra las etiquetas de un recurso de AmazonEKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Salida:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Para API obtener más información, consulte la referencia del ListTagsForResource AWS Tools for PowerShellcmdlet.

`Get-EKSUpdate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSUpdate`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet devuelve información descriptiva sobre una actualización de su EKS clúster de Amazon o grupo de nodos gestionados asociado.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Salida:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Para API obtener más información, consulte la referencia del DescribeUpdate AWS Tools for PowerShellcmdlet.

`Get-EKSUpdateList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-EKSUpdateList`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet muestra las actualizaciones asociadas a un EKS clúster de Amazon o a un grupo de nodos gestionado en su región Cuenta de AWS, en la región especificada.

```
Get-EKSUpdateList -Name "PROD"
```

Salida:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Para API obtener más información, consulte la referencia del ListUpdates AWS Tools for PowerShellcmdlet.

`New-EKSCluster`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EKSCluster`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo clúster denominado «prod».

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateClusterReference.

`New-EKSFargateProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EKSFargateProfile`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet crea un perfil de AWS Fargate para tu clúster de Amazon. EKS Debe tener al menos un perfil de Fargate en un clúster para poder programar pods en la infraestructura de Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

Salida:

```
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                : {}
```

- Para API obtener más información, consulte la referencia de CreateFargateProfile AWS Tools for PowerShellcmdlets.

`New-EKSNodeGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-EKSNodeGroup`

Herramientas para PowerShell

Ejemplo 1: este cmdlet crea un grupo de nodos de trabajo gestionado para un clúster de AmazonEKS. Solo puede crear un grupo de nodos para el clúster que sea igual a la versión actual de Kubernetes para el clúster. Todos los grupos de nodos se crean con la AMI versión más reciente de la versión secundaria de Kubernetes correspondiente del clúster.

```
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"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia de CreateNodegroupcmdlets AWS Tools for PowerShell .

`Remove-EKSCluster`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EKSCluster`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina el plano de control del EKS clúster de Amazon.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DeleteClustercmdlet AWS Tools for PowerShell .

`Remove-EKSFargateProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EKSFargateProfile`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina un perfil de AWS Fargate. Al eliminar un perfil de Fargate, se eliminan todos los pods que se ejecuten en Fargate y que se hayan creado con el perfil.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Salida:

```
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                : {}
```

- Para API obtener más información, consulte la referencia de DeleteFargateProfile AWS Tools for PowerShellcmdlets.

`Remove-EKSNodegroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EKSNodegroup`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet elimina un grupo de EKS nodos de Amazon de un clúster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DeleteNodegroupcmdlet AWS Tools for PowerShell .

`Remove-EKSResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-EKSResourceTag`

Herramientas para PowerShell

Ejemplo 1: este cmdlet elimina las etiquetas especificadas de un recurso. EKS

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del UntagResourcecmdlet AWS Tools for PowerShell .

`Update-EKSClusterConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-EKSClusterConfig`

Herramientas para PowerShell

Ejemplo 1: actualiza la configuración de un EKS clúster de Amazon. El clúster seguirá funcionando durante la actualización.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Salida:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Para API obtener más información, consulte UpdateClusterConfigla referencia de AWS Tools for PowerShell cmdlets.

`Update-EKSClusterVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-EKSClusterVersion`

Herramientas para PowerShell

Ejemplo 1: Este cmdlet actualiza un EKS clúster de Amazon a la versión de Kubernetes especificada. El clúster seguirá funcionando durante la actualización.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Salida:

```
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
```

- Para API obtener más información, consulte UpdateClusterVersionla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Elastic Load Balancing: versión 1 con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell con Elastic Load Balancing, versión 1.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-ELBLoadBalancerToSubnet`

En el siguiente ejemplo de código se muestra cómo usarloAdd-ELBLoadBalancerToSubnet.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega la subred especificada al conjunto de subredes configuradas para el balanceador de cargas especificado. El resultado incluye la lista completa de subredes.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Salida:

```
subnet-12345678
subnet-87654321
```

- Para API obtener más información, consulte la referencia AttachLoadBalancerToSubnetsde AWS Tools for PowerShell cmdlets.

`Add-ELBResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-ELBResourceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se añaden las etiquetas especificadas al balanceador de cargas especificado. La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o posterior.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear una etiqueta para el parámetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet AddTagsReference.

`Disable-ELBAvailabilityZoneForLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-ELBAvailabilityZoneForLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la zona de disponibilidad especificada del equilibrador de cargas especificado. El resultado incluye las zonas de disponibilidad restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Salida:

```
us-west-2b
```

- Para API obtener más información, consulte DisableAvailabilityZonesForLoadBalancerla referencia del AWS Tools for PowerShell cmdlet.

`Dismount-ELBLoadBalancerFromSubnet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Dismount-ELBLoadBalancerFromSubnet`

Herramientas para PowerShell

Ejemplo 1: este ejemplo elimina la subred especificada del conjunto de subredes configuradas para el balanceador de cargas especificado. El resultado incluye las subredes restantes.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Salida:

```
subnet-87654321
```

- Para API obtener más información, consulte la referencia DetachLoadBalancerFromSubnetsde AWS Tools for PowerShell cmdlets.

`Edit-ELBLoadBalancerAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELBLoadBalancerAttribute`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita el equilibrio de carga entre zonas para el balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Ejemplo 2: Este ejemplo deshabilita el drenaje de conexiones para el balanceador de cargas especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Ejemplo 3: Este ejemplo habilita el registro de acceso para el balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Para API obtener más información, consulte ModifyLoadBalancerAttributes AWS Tools for PowerShellCmdlet Reference.

`Enable-ELBAvailabilityZoneForLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-ELBAvailabilityZoneForLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega la zona de disponibilidad especificada al balanceador de cargas especificado. El resultado incluye la lista completa de zonas de disponibilidad.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Salida:

```
us-west-2a
us-west-2b
```

- Para API obtener más información, consulte EnableAvailabilityZonesForLoadBalancerla referencia del AWS Tools for PowerShell cmdlet.

`Get-ELBInstanceHealth`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBInstanceHealth`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe el estado de las instancias registradas con el balanceador de cargas especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Salida:

```
Description                  InstanceId                  ReasonCode
      State
-----------                  ----------                  ----------
      -----
N/A                          i-87654321                  N/A
      InService
Instance has failed at lea... i-12345678                 Instance
      OutOfService
```

Ejemplo 2: en este ejemplo se describe el estado de la instancia especificada registrada con el balanceador de cargas especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Ejemplo 3: en este ejemplo se muestra la descripción completa del estado de la instancia especificada.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Salida:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Para API obtener más información, consulte DescribeInstanceHealth AWS Tools for PowerShellCmdlet Reference.

`Get-ELBLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los nombres de los balanceadores de carga.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Salida:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Ejemplo 2: en este ejemplo se describe el balanceador de cargas especificado.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Salida:

```
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
```

Ejemplo 3: En este ejemplo se describen todos los balanceadores de carga de la región actual AWS .

```
Get-ELBLoadBalancer
```

Ejemplo 4: En este ejemplo se describen todos los balanceadores de carga disponibles. Regiones de AWS

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Para API obtener más información, consulta DescribeLoadBalancers AWS Tools for PowerShellCmdlet Reference.

`Get-ELBLoadBalancerAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBLoadBalancerAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen los atributos del balanceador de cargas especificado.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Salida:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Para API obtener más información, consulte DescribeLoadBalancerAttributes AWS Tools for PowerShellCmdlet Reference.

`Get-ELBLoadBalancerPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBLoadBalancerPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen las políticas asociadas al balanceador de cargas especificado.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Salida:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
--------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Ejemplo 2: en este ejemplo se describen los atributos de la política especificada.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Salida:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Ejemplo 3: En este ejemplo se describen las políticas predefinidas, incluidas las políticas de ejemplo. Los nombres de las políticas de muestra tienen el prefijo ELBSample -.

```
Get-ELBLoadBalancerPolicy
```

Salida:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
--------------------------             ----------
 --------------
{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
```

- Para API obtener más información, consulte la referencia DescribeLoadBalancerPoliciesdel AWS Tools for PowerShell cmdlet.

`Get-ELBLoadBalancerPolicyType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBLoadBalancerPolicyType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los tipos de políticas compatibles con Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se describe el tipo de política especificado.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Salida:

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Ejemplo 3: En este ejemplo se muestra la descripción completa del tipo de política especificado.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Salida:

```
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
```

- Para API obtener más información, consulte [DescribeLoadBalancerPolicyTypes AWS Tools for PowerShell](#)Cmdlet Reference.

`Get-ELBResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELBResourceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran las etiquetas de los balanceadores de carga especificados.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Salida:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Ejemplo 2: en este ejemplo se describen las etiquetas del equilibrador de carga especificado.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Salida:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Para API obtener más información, consulte DescribeTags AWS Tools for PowerShellCmdlet Reference.

`Join-ELBSecurityGroupToLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Join-ELBSecurityGroupToLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se reemplaza el grupo de seguridad actual del balanceador de cargas especificado por el grupo de seguridad especificado.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Salida:

```
sg-87654321
```

Ejemplo 2: Para conservar el grupo de seguridad actual y especificar un grupo de seguridad adicional, especifique los grupos de seguridad existentes y nuevos.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Salida:

```
sg-12345678
sg-87654321
```

- Para API obtener más información, consulte ApplySecurityGroupsToLoadBalancerla referencia de AWS Tools for PowerShell cmdlets.

`New-ELBAppCookieStickinessPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELBAppCookieStickinessPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una política de adherencia que sigue la duración de la sesión fija de la cookie generada por la aplicación especificada.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- Para obtener API más información, consulte Cmdlet Reference. CreateAppCookieStickinessPolicyAWS Tools for PowerShell

`New-ELBLBCookieStickinessPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELBLBCookieStickinessPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una política de adherencia con la duración de las sesiones fijas controlada según el período de caducidad especificado (en segundos).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy -CookieExpirationPeriod 60
```

Ejemplo 2: En este ejemplo, se crea una política de continuidad con una duración de sesión fija controlada por la duración del navegador (agente de usuario).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- Para obtener API más información, consulte CreateLbCookieStickinessPolicyCmdlet Reference.AWS Tools for PowerShell

`New-ELBLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELBLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un equilibrador de carga con un HTTP oyente en un. VPC

```
$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
```

Ejemplo 2: En este ejemplo se crea un balanceador de cargas con un agente de HTTP escucha en -Classic. EC2

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

Salida:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Ejemplo 3: En este ejemplo se crea un equilibrador de carga con un agente de escucha. HTTPS

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateLoadBalancerReference.

`New-ELBLoadBalancerListener`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELBLoadBalancerListener`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se añade un agente de HTTPS escucha al balanceador de cargas especificado.

```
$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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet CreateLoadBalancerListenersReference.

`New-ELBLoadBalancerPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELBLoadBalancerPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una nueva política de protocolo de proxy para un balanceador de cargas específico.

```
$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
```

- Para API obtener más información, consulte la referencia CreateLoadBalancerPolicyde AWS Tools for PowerShell cmdlets.

`Register-ELBInstanceWithLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-ELBInstanceWithLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra la EC2 instancia especificada con el balanceador de cargas especificado.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Salida:

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

- Para API obtener más información, consulte RegisterInstancesWithLoadBalancer AWS Tools for PowerShellCmdlet Reference.

`Remove-ELBInstanceFromLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELBInstanceFromLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la EC2 instancia especificada del balanceador de cargas especificado. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Salida:

```
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
```

- Para API obtener más información, consulte DeregisterInstancesFromLoadBalancer AWS Tools for PowerShellCmdlet Reference.

`Remove-ELBLoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELBLoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el balanceador de cargas especificado. Se le solicitará una confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteLoadBalancer AWS Tools for PowerShellCmdlet Reference.

`Remove-ELBLoadBalancerListener`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELBLoadBalancerListener`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el listener del puerto 80 del balanceador de cargas especificado. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteLoadBalancerListeners AWS Tools for PowerShellCmdlet Reference.

`Remove-ELBLoadBalancerPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELBLoadBalancerPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la política especificada del balanceador de cargas especificado. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Salida:

```
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"):
```

- Para API obtener más información, consulte DeleteLoadBalancerPolicy AWS Tools for PowerShellCmdlet Reference.

`Remove-ELBResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELBResourceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta especificada del balanceador de cargas especificado. Se le solicitará la confirmación antes de continuar con la operación, a menos que también especifique el parámetro Force. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Salida:

```
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"):
```

Ejemplo 2: Con la versión 2 de Powershell, debe usar New-Object para crear la etiqueta del parámetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Para API obtener más información, consulte Cmdlet RemoveTagsReference AWS Tools for PowerShell .

`Set-ELBHealthCheck`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELBHealthCheck`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se configuran los ajustes de comprobación de estado del equilibrador de cargas especificado.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
```

```
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Salida:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Para API obtener más información, consulte la referencia de ConfigureHealthCheck AWS Tools for PowerShellcmdlets.

`Set-ELBLoadBalancerListenerSSLCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELBLoadBalancerListenerSSLCertificate`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo reemplaza el certificado que termina las SSL conexiones del oyente especificado.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet SetLoadBalancerListenerSslCertificateReference.

`Set-ELBLoadBalancerPolicyForBackendServer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELBLoadBalancerPolicyForBackendServer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se sustituyen las políticas del puerto especificado por la política especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Ejemplo 2: en este ejemplo se eliminan todas las políticas asociadas al puerto especificado.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Para API obtener más información, consulte SetLoadBalancerPoliciesForBackendServer AWS Tools for PowerShellCmdlet Reference.

`Set-ELBLoadBalancerPolicyOfListener`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELBLoadBalancerPolicyOfListener`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se sustituyen las políticas del agente de escucha especificado por la política especificada.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Ejemplo 2: en este ejemplo se eliminan todas las políticas asociadas al agente de escucha especificado.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Para API obtener más información, consulte SetLoadBalancerPoliciesOfListener AWS Tools for PowerShellCmdlet Reference.

Ejemplos de Elastic Load Balancing: versión 2 con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell con Elastic Load Balancing, versión 2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-ELB2ListenerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-ELB2ListenerCertificate`.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo agrega un certificado adicional al Listener especificado.

```
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'}
```

Salida:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- Para API obtener más información, consulte AddListenerCertificates AWS Tools for PowerShellCmdlet Reference.

`Add-ELB2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-ELB2Tag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se agrega una nueva etiqueta al **`AWS.Tools.ElasticLoadBalancingV2`** recurso especificado.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Para API obtener más información, consulte AddTagsla referencia del AWS Tools for PowerShell cmdlet.

`Edit-ELB2Listener`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELB2Listener`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica la acción predeterminada del oyente especificado a una respuesta fija.

```
$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
```

Salida:

```
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       :
```

- Para obtener API más información, consulte ModifyListenerla referencia del cmdlet.AWS Tools for PowerShell

`Edit-ELB2LoadBalancerAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELB2LoadBalancerAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se modifican los atributos del balanceador de cargas especificado.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ModifyLoadBalancerAttributesReference.

`Edit-ELB2Rule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELB2Rule`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se modifican las configuraciones de las reglas de escucha especificadas.

```
$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
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ModifyRuleReference.

`Edit-ELB2TargetGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELB2TargetGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se modifican las propiedades del grupo objetivo especificado.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Salida:

```
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
```

- Para API obtener más información, consulte ModifyTargetGroup AWS Tools for PowerShellCmdlet Reference.

`Edit-ELB2TargetGroupAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-ELB2TargetGroupAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se modifica el atributo deregistration_delay del grupo objetivo especificado.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Salida:

```
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
```

- Para obtener API más información, consulte la referencia del cmdlet. ModifyTargetGroupAttributesAWS Tools for PowerShell

`Get-ELB2AccountLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2AccountLimit`

Herramientas para PowerShell

Ejemplo 1: Este comando muestra los límites de las ELB2 cuentas para una región determinada.

```
Get-ELB2AccountLimit
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeAccountLimitsla referencia del AWS Tools for PowerShell cmdlet.

`Get-ELB2Listener`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2Listener`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo describe a los oyentes delALB/NLBespecificado.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeListenersdel AWS Tools for PowerShell cmdlet.

`Get-ELB2ListenerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2ListenerCertificate`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el certificado del oyente especificado.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Salida:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- Para API obtener más información, consulte DescribeListenerCertificates AWS Tools for PowerShellCmdlet Reference.

`Get-ELB2LoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2LoadBalancer`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran todos los balanceadores de carga de la región en cuestión.

```
Get-ELB2LoadBalancer
```

Salida:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
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
```

- Para API obtener más información, consulte DescribeLoadBalancers AWS Tools for PowerShellCmdlet Reference.

`Get-ELB2LoadBalancerAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2LoadBalancerAttribute`

Herramientas para PowerShell

Ejemplo 1: Este comando describe los atributos de un balanceador de carga determinado.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeLoadBalancerAttributesdel AWS Tools for PowerShell cmdlet.

`Get-ELB2Rule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2Rule`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen las reglas de escucha para el ARN listener especificado.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeRulesReference.

`Get-ELB2SSLPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2SSLPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todas las políticas de escucha disponibles para la ElasticLoadBalancing versión 2.

```
Get-ELB2SSLPolicy
```

Salida:

```
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}
```

- Para API obtener más información, consulte DescribeSslPolicies AWS Tools for PowerShellCmdlet Reference.

`Get-ELB2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2Tag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran las etiquetas del recurso especificado.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Salida:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Para API obtener más información, consulte DescribeTagsla referencia del AWS Tools for PowerShell cmdlet.

`Get-ELB2TargetGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2TargetGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se describe el grupo objetivo especificado.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeTargetGroups AWS Tools for PowerShellCmdlet Reference.

`Get-ELB2TargetGroupAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2TargetGroupAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen los atributos del grupo objetivo especificado.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeTargetGroupAttributes AWS Tools for PowerShellCmdlet Reference.

`Get-ELB2TargetHealth`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ELB2TargetHealth`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el estado de salud de los objetivos presentes en el grupo objetivo especificado.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Salida:

```
HealthCheckPort Target                                                  TargetHealth
--------------- ------                                                  ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Para API obtener más información, consulte DescribeTargetHealth AWS Tools for PowerShellCmdlet Reference.

`New-ELB2Listener`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELB2Listener`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo ALB oyente con la acción predeterminada «Reenviar» para enviar el tráfico al grupo objetivo especificado.

```
$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
```

Salida:

```
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       :
```

- Para API obtener más información, consulte CreateListener AWS Tools for PowerShellCmdlet Reference.

`New-ELB2LoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELB2LoadBalancer`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un nuevo balanceador de carga de aplicaciones orientado a Internet con dos subredes.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Salida:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
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
```

- Para API obtener más información, consulte la referencia de CreateLoadBalancer AWS Tools for PowerShellcmdlets.

New-ELB2Rule

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ELB2Rule`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva regla de escucha con una acción de respuesta fija basada en el valor del encabezado del cliente para el oyente especificado.

```
$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
```

Salida:

```
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
```

- Para obtener API más información, consulte CreateRulela referencia del cmdlet.AWS Tools for PowerShell

New-ELB2TargetGroup

En el siguiente ejemplo de código se muestra cómo usarlo. New-ELB2TargetGroup

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un nuevo grupo objetivo con los parámetros proporcionados.

```
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'
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateTargetGroup AWS Tools for PowerShellCmdlet Reference.

Register-ELB2Target

En el siguiente ejemplo de código se muestra cómo usarlo. Register-ELB2Target

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra la instancia 'i-0672a4c4cdeae3111' con el grupo objetivo especificado.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- Para RegisterTargetsobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

`Remove-ELB2Listener`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2Listener`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el Listener especificado.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se elimina el agente de escucha especificado del balanceador de carga.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Salida:

```
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
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteListenerReference.

`Remove-ELB2ListenerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2ListenerCertificate`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el certificado especificado del grupo objetivo especificado.

```
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'
```

Salida:

```
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
```

- Para API obtener más información, consulte RemoveListenerCertificates AWS Tools for PowerShellCmdlet Reference.

`Remove-ELB2LoadBalancer`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2LoadBalancer`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el balanceador de carga especificado.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DeleteLoadBalancer AWS Tools for PowerShellcmdlet.

`Remove-ELB2Rule`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2Rule`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la regla especificada del Listener

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteRule AWS Tools for PowerShellCmdlet Reference.

`Remove-ELB2Tag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2Tag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta de la clave especificada.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Salida:

```
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
```

- Para API obtener más información, consulte RemoveTags AWS Tools for PowerShellCmdlet Reference.

`Remove-ELB2TargetGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-ELB2TargetGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el grupo objetivo especificado.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssss/4e0b6076bc6483a7'
```

Salida:

```
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/
testssssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para API obtener más información, consulte DeleteTargetGroup AWS Tools for PowerShellCmdlet Reference.

`Set-ELB2IpAddressType`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELB2IpAddressType`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el tipo de dirección IP del balanceador de carga de 'IPv4' a 'DualStack'.

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Salida:

```
Value
-----
dualstack
```

- Para API obtener más información, consulta la referencia SetIpAddressTypede AWS Tools for PowerShell cmdlets.

`Set-ELB2RulePriority`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELB2RulePriority`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia la prioridad de la regla de escucha especificada.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Salida:

```
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
```

- Para API obtener más información, consulte SetRulePriorities AWS Tools for PowerShellCmdlet Reference.

`Set-ELB2SecurityGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELB2SecurityGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega el grupo de seguridad «sg-07c3414abb8811cbd» al balanceador de cargas especificado.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Salida:

```
sg-07c3414abb8811cbd
```

- Para SetSecurityGroupsobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

`Set-ELB2Subnet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-ELB2Subnet`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo modifica las subredes del balanceador de carga especificado.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Salida:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Para API obtener más información, consulte la referencia de SetSubnetscmdlets AWS Tools for PowerShell .

`Unregister-ELB2Target`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-ELB2Target`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se anula el registro de la instancia 'i-0672a4c4cdeae3111' del grupo de destino especificado.

```
$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'
```

- Para DeregisterTargetsobtener AWS Tools for PowerShell más información, consulte Cmdlet Reference. API

FSxEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonFSx.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-FSXResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo`Add-FSXResourceTag`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se añaden etiquetas al recurso dado.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

Salida:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Para API obtener más información, consulte TagResource AWS Tools for PowerShellCmdlet Reference.

`Get-FSXBackup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-FSXBackup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo busca las copias de seguridad creadas desde ayer para el identificador del sistema de archivos indicado.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia DescribeBackupsde AWS Tools for PowerShell cmdlets.

`Get-FSXFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-FSXFileSystem`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve la descripción de lo dadofilesystemId.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeFileSystems AWS Tools for PowerShellCmdlet Reference.

`Get-FSXResourceTagList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-FSXResourceTagList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las etiquetas del recurso arn proporcionado.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Salida:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- Para API obtener más información, consulte la referencia ListTagsForResourcedel AWS Tools for PowerShell cmdlet.

`New-FSXBackup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-FSXBackup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una copia de seguridad del sistema de archivos indicado.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateBackup AWS Tools for PowerShellCmdlet Reference.

`New-FSXFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-FSXFileSystem`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un nuevo sistema de archivos Windows de 300 GB, que permite el acceso desde la subred especificada, y que admite un rendimiento de hasta 8 megabytes por segundo. El nuevo sistema de archivos se une automáticamente al Microsoft Active Directory especificado.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateFileSystemla referencia de AWS Tools for PowerShell cmdlets.

`New-FSXFileSystemFromBackup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-FSXFileSystemFromBackup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un nuevo sistema de FSx archivos de Amazon a partir de una copia de seguridad existente de Amazon FSx for Windows File Server.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia CreateFileSystemFromBackupde AWS Tools for PowerShell cmdlets.

`Remove-FSXBackup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-FSXBackup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el identificador de copia de seguridad indicado.

```
Remove-FSXBackup -BackupId $backupID
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del DeleteBackupcmdlet AWS Tools for PowerShell .

`Remove-FSXFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-FSXFileSystem`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el ID del sistema de FSX archivos indicado.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteFileSystemla referencia del AWS Tools for PowerShell cmdlet.

`Remove-FSXResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-FSXResourceTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la etiqueta de recurso del sistema de FSX archivos correspondienteARN.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Salida:

```
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
```

- Para API obtener más información, consulte UntagResource AWS Tools for PowerShellCmdlet Reference.

`Update-FSXFileSystem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-FSXFileSystem`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo actualiza los días de retención de copias de seguridad automáticas del sistema de FSX archivos mediante UpdateFileSystemWindowsConfiguration.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
 $UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
 Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
 $UpdateFSXWinConfig
```

Salida:

```
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
```

- Para API obtener más información, consulte UpdateFileSystemla referencia de AWS Tools for PowerShell cmdlets.

AWS Glue ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Glue.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`New-GLUEJob`

En el siguiente ejemplo de código se muestra cómo usarlo`New-GLUEJob`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo trabajo en AWS Glue. El valor del nombre del comando es siempre**`glueetl`**. AWS Glue admite la ejecución de scripts de trabajo escritos en Python o Scala. En este ejemplo, el script de trabajo (MyTestGlueJob.py) está escrito en Python. Los parámetros de Python se especifican en la **`$DefArgs`** variable y, a continuación, se pasan al PowerShell comando del **`DefaultArguments`** parámetro, que acepta una tabla hash. Los parámetros de la **`$JobParams`** variable provienen del CreateJob API tema Jobs (https://docs.aws.amazon.com/glue/latest/dg/ aws-glue-api-jobs -job.html) de la referencia de Glue. AWS API

```
$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
```

- Para obtener API más información, consulte CreateJobCmdlet Reference.AWS Tools for PowerShell

AWS Health ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Health.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-HLTHEvent`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-HLTHEvent`.

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los eventos del AWS Personal Health Dashboard. El usuario añade el parámetro -Region para ver los eventos disponibles para el servicio en la región EE.UU. Este (Norte de Virginia), pero el parámetro -Filter_Region filtra los eventos que se registran en las regiones UE (Londres) y EE.UU. Oeste (Oregón) (eu-west-2 y us-west-2). El StartTime parámetro -Filter_ filtra por un rango de horas en las que los eventos pueden comenzar, mientras que el EndTime parámetro -Filter_ filtra por un rango de horas en las que pueden terminar los eventos. El resultado es un evento de mantenimiento programado RDS que comienza dentro del rango -Filter_ especificado y termina dentro del StartTime rango -Filter_ programado. 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"}
```

Salida:

```
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
```

- Para obtener API más información, consulte Cmdlet Reference. DescribeEventsAWS Tools for PowerShell

IAMejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell withIAM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-IAMClientIDToOpenIDConnectProvider`

En el siguiente ejemplo de código se muestra cómo usarloAdd-IAMClientIDToOpenIDConnectProvider.

Herramientas para PowerShell

Ejemplo 1: Este comando agrega el ID de cliente (o audiencia) **`my-application-ID`** al OIDC proveedor existente nombrado**`server.example.com`**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Para API obtener más información, consulte AddClientIdToOpenIdConnectProviderla referencia del AWS Tools for PowerShell cmdlet.

`Add-IAMRoleTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-IAMRoleTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega una etiqueta al rol en el servicio de administración de identidades

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para API obtener más información, consulte TagRolela referencia de AWS Tools for PowerShell cmdlets.

`Add-IAMRoleToInstanceProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-IAMRoleToInstanceProfile`

Herramientas para PowerShell

Ejemplo 1: este comando añade el rol denominado **`S3Access`** a un perfil de instancia existente denominado **`webserver`**. Para crear el perfil de instancia, utilice el comando **`New-IAMInstanceProfile`**. Tras crear el perfil de la instancia y asociarlo a un rol mediante este comando, puede adjuntarlo a una EC2 instancia. Para ello, utilice el cmdlet de **`New-EC2Instance`** con el parámetro **`InstanceProfile_Arn`** o **`InstanceProfile-Name`** para iniciar la nueva instancia.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Para API obtener más información, consulte AddRoleToInstanceProfilela referencia del AWS Tools for PowerShell cmdlet.

`Add-IAMUserTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-IAMUserTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega una etiqueta al usuario en el servicio de administración de identidades

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para API obtener más información, consulte TagUserla referencia de AWS Tools for PowerShell cmdlets.

`Add-IAMUserToGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Add-IAMUserToGroup`

Herramientas para PowerShell

Ejemplo 1: este comando agrega el usuario llamado **`Bob`** al grupo denominado **`Admins`**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Para API obtener más información, consulte AddUserToGroupla referencia de AWS Tools for PowerShell cmdlets.

`Disable-IAMMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Disable-IAMMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este comando desactiva el MFA dispositivo de hardware asociado al usuario **`Bob`** que tiene el número **`123456789012`** de serie.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Ejemplo 2: Este comando deshabilita el MFA dispositivo virtual asociado al usuario **`David`** que tiene el. ARN **`arn:aws:iam::210987654321:mfa/David`** Tenga en cuenta que MFA el dispositivo virtual no se elimina de la cuenta. El dispositivo virtual sigue presente y aparece en la salida del comando **`Get-IAMVirtualMFADevice`**. Antes de poder crear un nuevo MFA dispositivo virtual para el mismo usuario, debe eliminar el anterior mediante el **`Remove-IAMVirtualMFADevice`** comando.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- Para API obtener más información, consulte DeactivateMfaDevicela referencia de AWS Tools for PowerShell cmdlets.

`Edit-IAMPassword`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-IAMPassword`

Herramientas para PowerShell

Ejemplo 1: este comando cambia la contraseña del usuario que ejecuta el comando. Solo los IAM usuarios pueden invocar este comando. Si se ejecuta este comando al iniciar sesión con las credenciales de la AWS cuenta (root), el comando devolverá un error. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Para API obtener más información, consulte la referencia de ChangePasswordcmdlets AWS Tools for PowerShell .

`Enable-IAMMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Enable-IAMMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este comando habilita el MFA dispositivo de hardware con el número de serie **`987654321098`** y lo asocia al usuario**`Bob`**. Incluye los dos primeros códigos secuenciales del dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Ejemplo 2: Este ejemplo crea y habilita un MFA dispositivo virtual. El primer comando crea el dispositivo virtual y devuelve la representación del objeto del dispositivo en la variable **`$MFADevice`**. Puede usar las propiedades **`.Base32StringSeed`** o **`QRCodePng`** para configurar la aplicación de software del usuario. El comando final asigna el dispositivo al usuario **`David`** e identifica el dispositivo por su número de serie. El comando también sincroniza el dispositivo AWS al incluir los dos primeros códigos secuenciales del MFA dispositivo virtual.

```
$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 -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Para API obtener más información, consulte EnableMfaDevice AWS Tools for PowerShellCmdlet Reference.

`Get-IAMAccessKey`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccessKey`

Herramientas para PowerShell

Ejemplo 1: Este comando muestra las claves de acceso del IAM usuario nombrado**Bob**. Tenga en cuenta que no puede enumerar las claves de acceso secretas de IAM los usuarios. Si se pierden las claves de acceso secretas, debe crear nuevas claves de acceso mediante el comando de cmdlet de **`New-IAMAccessKey`**.

```
Get-IAMAccessKey -UserName "Bob"
```

Salida:

```
AccessKeyId              CreateDate                 Status             UserName
-----------              ----------                 ------             --------
AKIAIOSFODNN7EXAMPLE     12/3/2014 10:53:41 AM      Active             Bob
AKIAI44QH8DHBEXAMPLE     6/6/2013 8:42:26 PM        Inactive           Bob
```

- Para API obtener más información, consulte ListAccessKeysla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMAccessKeyLastUsed`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccessKeyLastUsed`

Herramientas para PowerShell

Ejemplo 1: devuelve el nombre de usuario propietario y la información sobre el último uso de la clave de acceso proporcionada.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Para API obtener más información, consulte GetAccessKeyLastUsedla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMAccountAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccountAlias`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el alias de la cuenta para la Cuenta de AWS.

```
Get-IAMAccountAlias
```

Salida:

```
ExampleCo
```

- Para API obtener más información, consulte ListAccountAliasesla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMAccountAuthorizationDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccountAuthorizationDetail`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los detalles de autorización sobre las identidades de la AWS cuenta y se muestra la lista de elementos del objeto devuelto, incluidos los usuarios, los grupos y las funciones. Por ejemplo, la propiedad **`UserDetailList`** muestra detalles sobre los usuarios. Hay información similar disponible en las propiedades **`RoleDetailList`** y **`GroupDetailList`**.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Salida:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Salida:

```
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}
```

- Para API obtener más información, consulte GetAccountAuthorizationDetails AWS Tools for PowerShellCmdlet Reference.

`Get-IAMAccountPasswordPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccountPasswordPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo, se muestran detalles sobre la política de contraseñas de la cuenta actual. Si no se ha definido una política de contraseñas para la cuenta, el comando devuelve un error de **`NoSuchEntity`**.

```
Get-IAMAccountPasswordPolicy
```

Salida:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Para API obtener más información, consulte GetAccountPasswordPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMAccountSummary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAccountSummary`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve información sobre el uso actual de la IAM entidad y las cuotas actuales de la IAM entidad en Cuenta de AWS.

```
Get-IAMAccountSummary
```

Salida:

```
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
...
```

- Para API obtener más información, consulte GetAccountSummary AWS Tools for PowerShellCmdlet Reference.

`Get-IAMAttachedGroupPolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAttachedGroupPolicyList`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los nombres y ARNs las políticas administradas que están asociadas al IAM grupo nombrado **`Admins`** en la AWS cuenta. Para ver la lista de políticas integradas en el grupo, utilice el comando **`Get-IAMGroupPolicyList`**.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Salida:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- Para API obtener más información, consulte ListAttachedGroupPolicieslа referencia del AWS Tools for PowerShell cmdlet.

`Get-IAMAttachedRolePolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAttachedRolePolicyList`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los nombres y ARNs las políticas administradas asociadas a la IAM función nombrada **`SecurityAuditRole`** en la AWS cuenta. Para ver la lista de políticas integradas que están incrustadas en el rol, utilice el comando **`Get-IAMRolePolicyList`**.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Salida:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- Para API obtener más información, consulte ListAttachedRolePolicieslа referencia del AWS Tools for PowerShell cmdlet.

`Get-IAMAttachedUserPolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMAttachedUserPolicyList`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los nombres y ARNs las políticas administradas del IAM usuario nombrado **Bob** en la AWS cuenta. Para ver la lista de políticas integradas que están integradas en el IAM usuario, utilice el **`Get-IAMUserPolicyList`** comando.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Salida:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/TesterPolicy                      TesterPolicy
```

- Para API obtener más información, consulte la referencia ListAttachedUserPoliciesde AWS Tools for PowerShell cmdlets.

`Get-IAMContextKeysForCustomPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMContextKeysForCustomPolicy`

Herramientas para PowerShell

Ejemplo 1: este ejemplo busca todas las claves de contexto presentes en el JSON de la política proporcionada. Para proporcionar varias políticas, puede proporcionarlas como una lista de valores separados por comas.

```
$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
```

- Para API obtener más información, consulte GetContextKeysForCustomPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMContextKeysForPrincipalPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMContextKeysForPrincipalPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo busca todas las claves de contexto presentes en la política json proporcionada y las políticas asociadas a la IAM entidad (usuario/rol, etc.). Para: PolicyInputList puede proporcionar una lista de valores múltiples como valores separados por comas.

```
$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
```

- Para API obtener más información, consulte GetContextKeysForPrincipalPolicy AWS Tools for PowerShellCmdlet Reference.

`Get-IAMCredentialReport`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMCredentialReport`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se abre el informe devuelto y se envía a la :canalización como una matriz de líneas de texto. La primera línea es el encabezado con los nombres de las columnas separados por comas. Cada fila sucesiva es la fila de detalles de un usuario y cada campo está separado por comas. Para poder ver el informe, debe generarlo con el cmdlet **`Request-IAMCredentialReport`**. Para recuperar el informe como una cadena única, utilice **`-Raw`** en lugar de **`-AsTextArray`**. El alias **`-SplitLines`** también se acepta para el conmutador **`-AsTextArray`**. Para ver la lista completa de columnas de la salida, consulte la API referencia del

servicio. Tenga en cuenta que si no utiliza **`-AsTextArray`** o**`-SplitLines`**, debe extraer el texto de la **`.Content`** propiedad utilizando el. NET**`StreamReader`**clase.

```
Request-IAMCredentialReport
```

Salida:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Salida:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,passw
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04
A,false,N/A,false,N/A,false,N/A
Administrator,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+0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/
A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Para API obtener más información, consulte GetCredentialReport AWS Tools for PowerShellCmdlet Reference.

`Get-IAMEntitiesForPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMEntitiesForPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una lista de IAM grupos, funciones y usuarios que tienen la política **`arn:aws:iam::123456789012:policy/TestPolicy`** adjunta.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Salida:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Para API obtener más información, consulte ListEntitiesForPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve detalles sobre el IAM grupo**`Testers`**, incluida una colección de todos los IAM usuarios que pertenecen al grupo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Salida:

```
Group                                   IsTruncated           Marker
  Users
-----                                   -----------           ------
  -----
Amazon.IdentityManagement.Model.Group     False
  {Theresa, David}
```

```
$results.Group
```

Salida:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
```

```
Path       : /
```

```
$results.Users
```

Salida:

```
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           : 4OSVDDJJTF4XEEXAMPLE2
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
```

- Para API obtener más información, consulte GetGroup AWS Tools for PowerShellCmdlet Reference.

`Get-IAMGroupForUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMGroupForUser`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve la lista de IAM grupos a los que **`David`** pertenece el IAM usuario.

```
Get-IAMGroupForUser -UserName David
```

Salida:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
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    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Para API obtener más información, consulte ListGroupsForUser AWS Tools for PowerShellCmdlet Reference.

`Get-IAMGroupList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMGroupList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una colección de todos los IAM grupos definidos en el actual Cuenta de AWS.

```
Get-IAMGroupList
```

Salida:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /
```

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Para API obtener más información, consulte ListGroups AWS Tools for PowerShellCmdlet Reference.

`Get-IAMGroupPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMGroupPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelven detalles sobre la política en línea incrustada llamada **`PowerUserAccess-Testers`** del grupo **`Testers`**. La **`PolicyDocument`** propiedad está URL codificada. En este ejemplo, se decodifica con. **`UrlDecode`** NETmétodo.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Salida:

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

- Para API obtener más información, consulte GetGroupPolicy AWS Tools for PowerShellCmdlet Reference.

`Get-IAMGroupPolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMGroupPolicyList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo, se devuelve una lista de las políticas integradas que están dentro del grupo **`Testers`**. Para obtener las políticas administradas que están asociadas al grupo, utilice el comando **`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Salida:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Para API obtener más información, consulte ListGroupPolicieslа referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMInstanceProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMInstanceProfile`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve los detalles del perfil de instancia denominado **`ec2instancerole`** que está definido en la AWS cuenta corriente.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Salida:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
```

```
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para API obtener más información, consulte GetInstanceProfile AWS Tools for PowerShellCmdlet Reference.

`Get-IAMInstanceProfileForRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMInstanceProfileForRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles del perfil de instancia asociado al rol **`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Salida:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Para API obtener más información, consulte ListInstanceProfilesForRolela referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMInstanceProfileList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMInstanceProfileList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una colección de los perfiles de instancia definidos en la versión actual Cuenta de AWS.

```
Get-IAMInstanceProfileList
```

Salida:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para API obtener más información, consulte ListInstanceProfiles AWS Tools for PowerShellCmdlet Reference.

`Get-IAMLoginProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMLoginProfile`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve la fecha de creación de la contraseña y si es necesario restablecer la contraseña para el IAM usuario**`David`**.

```
Get-IAMLoginProfile -UserName David
```

Salida:

```
CreateDate                PasswordResetRequired              UserName
----------                ---------------------              --------
12/10/2014 3:39:44 PM     False                              David
```

- Para API obtener más información, consulte GetLoginProfile AWS Tools for PowerShellCmdlet Reference.

`Get-IAMMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve detalles sobre el MFA dispositivo asignado al IAM usuario**David**. En este ejemplo, puede darse cuenta de que se trata de un dispositivo virtual porque no **SerialNumber** es el número de serie real de un dispositivo físico. ARN

```
Get-IAMMFADevice -UserName David
```

Salida:

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- Para API obtener más información, consulte ListMfaDevicesla referencia del AWS Tools for PowerShell cmdlet.

Get-IAMOpenIDConnectProvider

En el siguiente ejemplo de código se muestra cómo usarlo. Get-IAMOpenIDConnectProvider

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve detalles sobre el proveedor de OpenID Connect cuyo ARN nombre es. **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** La **ClientIDList** propiedad es una colección que contiene todos los clientes IDs definidos para este proveedor.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Salida:

```
ClientIDList         CreateDate               ThumbprintList
        Url
------------         ----------               --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Para API obtener más información, consulte GetOpenIdConnectProvider AWS Tools for PowerShellCmdlet Reference.

`Get-IAMOpenIDConnectProviderList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMOpenIDConnectProviderList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una lista ARNS de todos los proveedores de OpenID Connect que están definidos en la versión actual. Cuenta de AWS

```
Get-IAMOpenIDConnectProviderList
```

Salida:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Para API obtener más información, consulte la referencia ListOpenIdConnectProvidersdel AWS Tools for PowerShell cmdlet.

`Get-IAMPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMPolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve detalles sobre la política gestionada de la que ARN se trata**`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Salida:

```
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
```

- Para API obtener más información, consulte GetPolicy AWS Tools for PowerShellCmdlet Reference.

`Get-IAMPolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMPolicyList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve una colección de las tres primeras políticas administradas disponibles en la AWS cuenta corriente. Como no **`-scope`** se especifica, de forma predeterminada es **`all`** e incluye tanto las políticas administradas como las AWS administradas por el cliente.

```
Get-IAMPolicyList -MaxItem 3
```

Salida:

```
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         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Ejemplo 2: Este ejemplo devuelve un conjunto de las dos primeras políticas gestionadas por el cliente disponibles en la AWS cuenta corriente. Se utiliza **`-Scope local`** para limitar la salida únicamente a las políticas administradas por el cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Salida:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
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         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Para API obtener más información, consulte ListPolicies AWS Tools for PowerShellCmdlet Reference.

`Get-IAMPolicyVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMPolicyVersion`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el documento de política correspondiente a la **v2** versión de la política cuya versión ARN es**arn:aws:iam::123456789012:policy/MyManagedPolicy**. El documento de política de la **Document** propiedad está URL codificado y decodificado en este ejemplo con. **UrlDecode** NETmétodo.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Salida:

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

- Para API obtener más información, consulte GetPolicyVersion AWS Tools for PowerShellCmdlet Reference.

`Get-IAMPolicyVersionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMPolicyVersionList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve la lista de versiones disponibles de la política cuya política ARN es**`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Para obtener el documento de política de una versión específica, utilice el comando **`Get-IAMPolicyVersion`** y especifique el **`VersionId`** que desee.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Salida:

```
CreateDate                 Document                IsDefaultVersion
    VersionId
----------                 --------                ----------------
    ---------
2/12/2015 9:39:53 AM                               True
    v2
2/12/2015 9:39:09 AM                               False
    v1
```

- Para API obtener más información, consulte ListPolicyVersions AWS Tools for PowerShellCmdlet Reference.

`Get-IAMRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMRole`

Ejemplo 1: este ejemplo devuelve los detalles de**`lamda_exec_role`**. Incluye el documento de política de confianza que especifica quién puede asumir este rol. El documento de política está URL codificado y se puede decodificar mediante. NET**`UrlDecode`**método. En este ejemplo, se eliminaron todos los espacios en blanco de la política original antes de cargarla en la política. Para ver los documentos de la política de permisos que determinan lo que puede hacer una persona que asume el rol, utilice **`Get-IAMRolePolicy`** para las políticas integradas y **`Get-IAMPolicyVersion`** para las políticas administradas adjuntas.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Salida:

```
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
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Salida:

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

- Para API obtener más información, consulte GetRole AWS Tools for PowerShellCmdlet Reference.

`Get-IAMRoleList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMRoleList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se recupera una lista de todas las IAM funciones de. Cuenta de AWS

```
Get-IAMRoleList
```

Ejemplo 2: Este fragmento de código de ejemplo recupera una lista de IAM funciones de la AWS cuenta, las muestra de tres en tres y espera a que pulses Entrar entre cada grupo. Transfiere el valor **`Marker`** de la llamada anterior para especificar dónde debe empezar el siguiente grupo.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- Para obtener API más información, consulte ListRolesCmdlet Reference.AWS Tools for PowerShell

`Get-IAMRolePolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMRolePolicy`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el documento de política de permisos de la política denominada **`oneClick_lambda_exec_role_policy`** que está incrustado en el IAM rol**`lamda_exec_role`**. El documento de política resultante está URL codificado. En este ejemplo, se decodifica con. **`UrlDecode`** NETmétodo.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Salida:

```
PolicyDocument                                                  PolicyName
           UserName
--------------                                                  ----------
           --------
%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)
```

Salida:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- Para API obtener más información, consulte GetRolePolicy AWS Tools for PowerShellCmdlet Reference.

`Get-IAMRolePolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMRolePolicyList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve la lista de nombres de las políticas integradas que están integradas en el IAM rol**`lamda_exec_role`**. Para ver los detalles de una política incrustada, utilice el comando **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Salida:

```
oneClick_lambda_exec_role_policy
```

- Para API obtener más información, consulte ListRolePolicies AWS Tools for PowerShellCmdlet Reference.

`Get-IAMRoleTagList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMRoleTagList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera la etiqueta asociada con el rol.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Para API obtener más información, consulte ListRoleTagsla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMSAMLProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMSAMLProvider`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera los detalles sobre el proveedor SAML 2.0, que ARM es arn:aws:iam: :123456789012:saml-provider/. SAMLADFS La respuesta incluye el documento de

metadatos que obtuvo del proveedor de identidades para crear la entidad proveedora, así como las fechas de creación y caducidad. AWS SAML

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Salida:

```
CreateDate                 SAMLMetadataDocument
     ValidUntil
----------                 --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- Para API obtener más información, consulte GetSamlProviderla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMSAMLProviderList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMSAMLProviderList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera la lista de proveedores SAML 2.0 creada en la versión actual Cuenta de AWS. Devuelve la ARN fecha de creación y la fecha de caducidad de cada SAML proveedor.

```
Get-IAMSAMLProviderList
```

Salida:

```
Arn                                                  CreateDate
 ValidUntil
---                                                  ----------
 ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- Para API obtener más información, consulte L istSAMLProviders en AWS Tools for PowerShell Cmdlet Reference.

`Get-IAMServerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMServerCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recuperan detalles sobre el certificado de servidor llamado **`MyServerCertificate`**. Puede encontrar los detalles del certificado en las propiedades **`CertificateBody`** y **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Salida:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                           YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                           21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                           rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
```

```
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Salida:

```
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
```

- Para API obtener más información, consulte GetServerCertificatela referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMServerCertificateList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMServerCertificateList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera la lista de certificados de servidor que han sido cargados en la Cuenta de AWS actual.

```
Get-IAMServerCertificateList
```

Salida:

```
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
```

- Para API obtener más información, consulte ListServerCertificatesla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMServiceLastAccessedDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMServiceLastAccessedDetail`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo proporciona detalles del servicio al que accedió por última vez la IAM entidad (usuario, grupo, rol o política) asociada a la llamada de solicitud.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Salida:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Para API obtener más información, consulte GetServiceLastAccessedDetails AWS Tools for PowerShellCmdlet Reference.

`Get-IAMServiceLastAccessedDetailWithEntity`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMServiceLastAccessedDetailWithEntity`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo proporciona la marca de tiempo del último acceso al servicio en la solicitud de la entidad correspondienteIAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Salida:

```
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
```

Salida:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Salida:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Para API obtener más información, consulte la referencia del GetServiceLastAccessedDetailsWithEntities AWS Tools for PowerShellcmdlet.

`Get-IAMSigningCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMSigningCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recuperan los detalles sobre el certificado de firma que está asociado con el usuario llamado **Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Salida:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para API obtener más información, consulte ListSigningCertificatesla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMUser`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recuperan detalles sobre el usurario llamado **David**.

```
Get-IAMUser -UserName David
```

Salida:

```
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
```

Ejemplo 2: este ejemplo recupera detalles sobre el usuario que ha iniciado sesión actualmente. IAM

```
Get-IAMUser
```

Salida:

```
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
```

- Para obtener API más información, consulte GetUserCmdlet Reference.AWS Tools for PowerShell

`Get-IAMUserList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMUserList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se recupera un conjunto de usuarios de la versión actual Cuenta de AWS.

```
Get-IAMUserList
```

Salida:

```
    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           : L3EWNONDOM3YUEXAMPLE2
    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
```

- Para API obtener más información, consulte ListUsers AWS Tools for PowerShellCmdlet Reference.

`Get-IAMUserPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMUserPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recuperan los detalles de la política en línea denominada **`Davids_IAM_Admin_Policy`** que está incrustada en el nombre de IAM usuario. **`David`** El documento de política está URL codificado.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Salida:

```
PolicyDocument                                          PolicyName
    UserName
```

```
--------------                                                  ----------
    --------
%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:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- Para API obtener más información, consulte GetUserPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMUserPolicyList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMUserPolicyList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera la lista de nombres de las políticas integradas que están integradas en el nombre de IAM usuario. **`David`**

```
Get-IAMUserPolicyList -UserName David
```

Salida:

```
Davids_IAM_Admin_Policy
```

- Para API obtener más información, consulte la referencia del ListUserPolicies AWS Tools for PowerShellcmdlet.

`Get-IAMUserTagList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMUserTagList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera la etiqueta asociada con el usuario.

```
Get-IAMUserTagList -UserName joe
```

- Para API obtener más información, consulte ListUserTagsla referencia de AWS Tools for PowerShell cmdlets.

`Get-IAMVirtualMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-IAMVirtualMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera una colección de MFA dispositivos virtuales que están asignados a los usuarios de la AWS cuenta. La **`User`** propiedad de cada uno es un objeto con detalles del IAM usuario al que está asignado el dispositivo.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Salida:

```
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
```

- Para API obtener más información, consulte ListVirtualMfaDevicesla Referencia de AWS Tools for PowerShell cmdlets.

`New-IAMAccessKey`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMAccessKey`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un nuevo par de clave de acceso y clave de acceso secreta y se lo asigna al usuario **`David`**. Asegúrese de guardar los valores de **`AccessKeyId`** y **`SecretAccessKey`** en un archivo, porque es la única oportunidad que tiene para obtener la **`SecretAccessKey`**. No puede recuperarla más tarde. Si pierde la clave secreta, debe crear un nuevo par de claves de acceso.

```
New-IAMAccessKey -UserName David
```

Salida:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Para API obtener más información, consulte CreateAccessKeyla referencia de AWS Tools for PowerShell cmdlets.

`New-IAMAccountAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMAccountAlias`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el alias de su AWS cuenta a**`mycompanyaws`**. La dirección de la página de inicio de sesión del usuario cambia a panyaws.signin.aws.amazon.com/console https://mycom. <accountidnumber>La original, que URL utilizaba tu número de ID de cuenta en lugar del alias (https://.signin.aws.amazon.com/console), sigue funcionando. Sin embargo, cualquier alias previamente definido deja de funcionar. URLs

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para API obtener más información, consulte la referencia de CreateAccountAlias AWS Tools for PowerShellcmdlets.

`New-IAMGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo IAM grupo denominado**`Developers`**.

```
New-IAMGroup -GroupName Developers
```

Salida:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Para API obtener más información, consulte CreateGroup AWS Tools for PowerShellCmdlet Reference.

`New-IAMInstanceProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMInstanceProfile`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo perfil de IAM instancia denominado**`ProfileForDevEC2Instance`**. Debe ejecutar el **`Add-IAMRoleToInstanceProfile`** comando por separado para asociar el perfil de la instancia a un IAM rol existente que otorgue permisos a la instancia. Por último, adjunta el perfil de la instancia a una EC2 instancia cuando la lances. Para ello, utilice el cmdlet de **`New-EC2Instance`** con el parámetro **`InstanceProfile_Arn`** o **`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Salida:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Para API obtener más información, consulte CreateInstanceProfilela referencia de AWS Tools for PowerShell cmdlets.

`New-IAMLoginProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMLoginProfile`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una contraseña (temporal) para el IAM usuario llamado Bob y se establece el indicador que exige que el usuario cambie la contraseña la próxima vez que **Bob** inicie sesión.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Salida:

```
CreateDate                  PasswordResetRequired              UserName
----------                  ---------------------              --------
4/14/2015 12:26:30 PM       True                               Bob
```

- Para API obtener más información, consulte CreateLoginProfile AWS Tools for PowerShellCmdlet Reference.

`New-IAMOpenIDConnectProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMOpenIDConnectProvider`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un IAM OIDC proveedor asociado al servicio de proveedor OIDC compatible que se encuentra en URL **`https://example.oidcprovider.com`** y en el

ID de cliente**my-testapp-1**. El OIDC proveedor proporciona la huella digital. Para autenticar la huella digital, siga los pasos que se indican en http://docs.aws.amazon.com/ IAM UserGuide / latest//-thumbprint.html. identity-providers-oidc-obtain

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-
testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Salida:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obtener API más información, consulte la Referencia de cmdlets. CreateOpenIdConnectProviderAWS Tools for PowerShell

`New-IAMPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva IAM política en la AWS cuenta corriente denominada **MySamplePolicy** El archivo **MySamplePolicy.json** proporciona el contenido de la política. Tenga en cuenta que debe usar el parámetro **-Raw** switch para procesar correctamente el archivo JSON de política.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Salida:

```
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
```

- Para API obtener más información, consulte CreatePolicyla referencia de AWS Tools for PowerShell cmdlets.

`New-IAMPolicyVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMPolicyVersion`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una nueva versión «v2" de la IAM política cuya versión ARN es **`arn:aws:iam::123456789012:policy/MyPolicy`** y se convierte en la versión predeterminada. El archivo **`NewPolicyVersion.json`** proporciona el contenido de la política. Tenga en cuenta que debe usar el parámetro **`-Raw`** switch para procesar correctamente el archivo JSON de política.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Salida:

```
CreateDate                          Document                IsDefaultVersion
        VersionId
----------                          --------                ----------------
        ---------
4/15/2015 10:54:54 AM                                       True
        v2
```

- Para API obtener más información, consulte CreatePolicyVersionla referencia de AWS Tools for PowerShell cmdlets.

`New-IAMRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un nuevo rol con el nombre **`MyNewRole`** y se le adjunta la política que se encuentra en el archivo **`NewRoleTrustPolicy.json`**. Tenga en cuenta que

debe usar el parámetro **`-Raw`** switch para procesar correctamente el archivo JSON de política. El documento de política que se muestra en el resultado está URL codificado. En este ejemplo, se decodifica con. **`UrlDecode`** NETmétodo.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Salida:

```
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%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%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%7D%2C%0D%0A%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"
    }
  ]
```

```
}
```

- Para API obtener más información, consulte CreateRole AWS Tools for PowerShellCmdlet Reference.

`New-IAMSAMLProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMSAMLProvider`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva entidad SAML proveedora enIAM. Se denomina **`MySAMLProvider`** y se describe en el documento de SAML metadatos que se encuentra en el archivo**`SAMLMetaData.xml`**, que se descargó por separado del sitio web del proveedor de SAML servicios.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Salida:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Para API obtener más información, consulte C reateSAMLProvider en la referencia de AWS Tools for PowerShell cmdlets.

`New-IAMServiceLinkedRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMServiceLinkedRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un rol vinculado a un servicio para el servicio de escalado automático.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Para API obtener más información, consulte CreateServiceLinkedRolela referencia de AWS Tools for PowerShell cmdlets.

`New-IAMUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMUser`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un IAM usuario llamado**Bob**. Si Bob necesita iniciar sesión en la AWS consola, debe ejecutar el comando por separado **`New-IAMLoginProfile`** para crear un perfil de inicio de sesión con una contraseña. Si Bob necesita ejecutar CLI comandos multiplataforma AWS PowerShell o realizar AWS API llamadas, debe ejecutar el **`New-IAMAccessKey`** comando por separado para crear las claves de acceso.

```
New-IAMUser -UserName Bob
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia CreateUserde AWS Tools for PowerShell cmdlets.

`New-IAMVirtualMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-IAMVirtualMFADevice`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo MFA dispositivo virtual. Las líneas 2 y 3 extraen el **`Base32StringSeed`** valor que el programa de MFA software virtual necesita para crear una cuenta (como alternativa al código QR). Después de configurar el programa con el valor, obtenga dos códigos de autenticación secuenciales del programa. Por último, utilice el último comando para vincular el MFA dispositivo virtual al IAM usuario **Bob** y sincronizar la cuenta con los dos códigos de autenticación.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
```

```
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Salida:

```
-- 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
```

Ejemplo 2: en este ejemplo se crea un nuevo MFA dispositivo virtual. Las líneas 2 y 3 extraen el valor **QRCodePNG** y lo escriben en un archivo. El programa de MFA software virtual puede escanear esta imagen para crear una cuenta (como alternativa a introducir manualmente el StringSeed valor de Base32). Después de crear la cuenta en su MFA programa virtual, obtenga dos códigos de autenticación secuenciales e introdúzcalos en los últimos comandos para vincular el MFA dispositivo virtual al IAM usuario **Bob** y sincronizar la cuenta.

```
$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
```

Salida:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Para API obtener más información, consulte la referencia de CreateVirtualMfaDevice AWS Tools for PowerShellcmdlets.

`Publish-IAMServerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Publish-IAMServerCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se carga un nuevo certificado de servidor en la IAM cuenta. Todos los archivos que contienen el cuerpo del certificado, la clave privada y (opcionalmente) la cadena de certificados deben estar PEM codificados. Tenga en cuenta que los parámetros requieren el contenido real de los archivos y no los nombres de los archivos. Tenga en cuenta que debe usar el parámetro de conmutación **`-Raw`** para procesar correctamente el contenido del archivo.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Salida:

```
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
```

- Para API obtener más información, consulte UploadServerCertificatela referencia del AWS Tools for PowerShell cmdlet.

`Publish-IAMSigningCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Publish-IAMSigningCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se carga un nuevo certificado de firma X.509 y se asocia al IAM usuario nombrado. **`Bob`** El archivo que contiene el cuerpo del certificado está PEM codificado. El parámetro **`CertificateBody`** requiere el contenido real del archivo de certificado y no el nombre del archivo. Debe usar el parámetro de conmutación **`-Raw`** para procesar correctamente el archivo.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Salida:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para API obtener más información, consulte UploadSigningCertificatela referencia del AWS Tools for PowerShell cmdlet.

`Register-IAMGroupPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-IAMGroupPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta la política gestionada por el cliente denominada **`TesterPolicy`** al IAM grupo**`Testers`**. Los usuarios de ese grupo se ven afectados inmediatamente por los permisos definidos en la versión predeterminada de esa política.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Ejemplo 2: en este ejemplo se adjunta la política AWS gestionada denominada **`AdministratorAccess`** al IAM grupo. **`Admins`** Los usuarios de ese grupo se ven afectados inmediatamente por los permisos definidos en la última versión de esa política.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Para API obtener más información, consulte AttachGroupPolicy AWS Tools for PowerShellCmdlet Reference.

`Register-IAMRolePolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-IAMRolePolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta la política AWS administrada denominada **`SecurityAudit`** al IAM rol**`CoSecurityAuditors`**. Los usuarios que asumen ese rol se ven afectados de inmediato por los permisos definidos en la última versión de esa política.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Para API obtener más información, consulte la referencia AttachRolePolicydel AWS Tools for PowerShell cmdlet.

`Register-IAMUserPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-IAMUserPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se adjunta la política AWS administrada denominada **`AmazonCognitoPowerUser`** al IAM usuario**`Bob`**. El usuario se ve afectado de inmediato por los permisos definidos en la última versión de esa política.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Para API obtener más información, consulte la referencia AttachUserPolicydel AWS Tools for PowerShell cmdlet.

`Remove-IAMAccessKey`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMAccessKey`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el par de claves de AWS acceso con el ID **`AKIAIOSFODNN7EXAMPLE`** de clave del usuario nombrado**`Bob`**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Para API obtener más información, consulte la referencia DeleteAccessKeydel AWS Tools for PowerShell cmdlet.

`Remove-IAMAccountAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMAccountAlias`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el alias de la cuenta de su Cuenta de AWS. La página de inicio de sesión del usuario con el alias https://mycom panyaws.signin.aws.amazon.com/console ya no funciona. En su lugar, debes usar el original con tu número de identificación en https://.signin.aws.amazon.com/console. URL Cuenta de AWS <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obtener más información, consulte la referencia del cmdlet. API DeleteAccountAliasAWS Tools for PowerShell

`Remove-IAMAccountPasswordPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMAccountPasswordPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la política de contraseñas Cuenta de AWS y se restablecen todos los valores a sus valores predeterminados originales. Si actualmente no existe

una política de contraseñas, aparece el siguiente mensaje de error: PasswordPolicy No se encuentra la política de cuentas con el nombre.

```
Remove-IAMAccountPasswordPolicy
```

- Para API obtener más información, consulte DeleteAccountPasswordPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMClientIDFromOpenIDConnectProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMClientIDFromOpenIDConnectProvider`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el ID **`My-TestApp-3`** de cliente de la lista de clientes IDs asociados al IAM OIDC proveedor del que se ARN encuentra**`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
```

- Para API obtener más información, consulte RemoveClientIdFromOpenIdConnectProvider AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el IAM grupo denominado**`MyTestGroup`**. El primer comando elimina todos IAM los usuarios que son miembros del grupo y el segundo elimina el IAM grupo. Ambos comandos funcionan sin ninguna solicitud de confirmación.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Para API obtener más información, consulte la referencia DeleteGroupde AWS Tools for PowerShell cmdlets.

`Remove-IAMGroupPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMGroupPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina del IAM grupo **`Testers`** la política en línea nombrada**`TesterPolicy`**. Los usuarios de ese grupo pierden los permisos definidos en esa política de inmediato.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Para API obtener más información, consulte DeleteGroupPolicy AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMInstanceProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMInstanceProfile`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el perfil de EC2 instancia denominado**`MyAppInstanceProfile`**. El primer comando separa todos los roles del perfil de instancia y, a continuación, el segundo comando elimina el perfil de instancia.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Para API obtener más información, consulte DeleteInstanceProfile AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMLoginProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMLoginProfile`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el perfil de inicio de sesión del IAM usuario nombrado**Bob**. Esto impide que el usuario inicie sesión en la consola. AWS No impide que el usuario ejecute ninguna AWS CLI API llamada o que utilice claves de AWS acceso que aún estén adjuntas a la cuenta de usuario. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Para API obtener más información, consulte DeleteLoginProfilela Referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMOpenIDConnectProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMOpenIDConnectProvider`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el IAM OIDC proveedor que se conecta al proveedor**example.oidcprovider.com**. Asegúrese de actualizar o eliminar todos los roles que hagan referencia a este proveedor en el elemento **Principal** de la política de confianza del rol.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para API obtener más información, consulte DeleteOpenIdConnectProvider AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMPolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la política cuyo nombre ARN es**arn:aws:iam::123456789012:policy/MySamplePolicy**. Antes de eliminar la política, primero debe eliminar todas las versiones, excepto la predeterminada, mediante la ejecución de **Remove-IAMPolicyVersion**. También debe separar la política de todos IAM los usuarios, grupos o roles.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ejemplo 2: En este ejemplo, se elimina una política; en primer lugar, se eliminan todas las versiones de la política no predeterminadas, se separa de todas las IAM entidades asociadas y, por último, se elimina la propia política. La primera línea recupera el objeto de política. La segunda línea recupera todas las versiones de la política que no están marcadas como la versión predeterminada de una colección y, a continuación, elimina todas las políticas de la colección. La tercera línea recupera todos los IAM usuarios, grupos y funciones a los que está asociada la política. Las líneas cuatro a seis separan la política de cada entidad adjunta. La última línea usa este comando para eliminar la política administrada, así como la versión predeterminada restante. El ejemplo incluye el parámetro de conmutación **`-Force`** en cualquier línea que lo necesite para suprimir las solicitudes de confirmación.

```
$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
```

- Para API obtener más información, consulte la referencia DeletePolicyde AWS Tools for PowerShell cmdlets.

`Remove-IAMPolicyVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMPolicyVersion`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la versión identificada como **`v2`** de la política cuya versión ARN es**`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Ejemplo 2: en este ejemplo, se elimina una política; en primer lugar, se eliminan todas las versiones de la política que no son predeterminadas y, a continuación, se elimina la propia

política. La primera línea recupera el objeto de política. La segunda línea recupera todas las versiones de la política que no están marcadas como predeterminadas en una colección y, a continuación, utiliza este comando para eliminar cada política de la colección. La última línea elimina la política en sí, así como el resto de la versión predeterminada. Tenga en cuenta que para eliminar correctamente una política administrada, también debe separar la política de cualquier usuario, grupo o rol mediante los comandos **`Unregister-IAMUserPolicy`**, **`Unregister-IAMGroupPolicy`** y **`Unregister-IAMRolePolicy`**. Consulte el ejemplo del cmdlet **`Remove-IAMPolicy`**.

```
$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
```

- Para API obtener más información, consulte DeletePolicyVersion AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el rol nombrado **`MyNewRole`** de la IAM cuenta actual. Antes de poder eliminar el rol, primero debe usar el comando **`Unregister-IAMRolePolicy`** para separar las políticas administradas. Las políticas integradas se eliminan con el rol.

```
Remove-IAMRole -RoleName MyNewRole
```

Ejemplo 2: en este ejemplo se separan las políticas administradas del rol denominado **`MyNewRole`** y, a continuación, se elimina el rol. La primera línea recupera todas las políticas administradas asociadas al rol como una colección y, a continuación, separa cada política de la colección del rol. La segunda línea elimina el rol en sí. Las políticas integradas se eliminan junto con el rol.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Para API obtener más información, consulte DeleteRole AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMRoleFromInstanceProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMRoleFromInstanceProfile`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el rol nombrado **`MyNewRole`** del perfil de EC2 instancia denominado**`MyNewRole`**. Un perfil de instancia que se crea en la IAM consola siempre tiene el mismo nombre que el rol, como en este ejemplo. Si los crea en API oCLI, pueden tener nombres diferentes.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- Para API obtener más información, consulte RemoveRoleFromInstanceProfilela referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMRolePermissionsBoundary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMRolePermissionsBoundary`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo eliminar el límite de permisos asociado a un IAM rol.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Para API obtener más información, consulte DeleteRolePermissionsBoundary AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMRolePolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMRolePolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la política en línea **`S3AccessPolicy`** que está integrada en el IAM rol. **`S3BackupRole`**

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Para API obtener más información, consulte la referencia de DeleteRolePolicy AWS Tools for PowerShellcmdlets.

`Remove-IAMRoleTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMRoleTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta del rol denominado «MyRoleName» con la clave de etiqueta como «abac». Para eliminar varias etiquetas, proporcione una lista de claves de etiquetas separadas por comas.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Para API obtener más información, consulte UntagRole AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMSAMLProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMSAMLProvider`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el proveedor IAM SAML 2.0 cuyo ARN es**`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Para API obtener más información, consulte D eleteSAMLProvider en la referencia del AWS Tools for PowerShell cmdlet.

`Remove-IAMServerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMServerCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el certificado de servidor denominado **`MyServerCert`**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Para API obtener más información, consulte DeleteServerCertificatela referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMServiceLinkedRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMServiceLinkedRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se eliminó el rol vinculado al servicio. Tenga en cuenta que si el servicio sigue utilizando este rol, este comando generará un error.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Para API obtener más información, consulte DeleteServiceLinkedRolela referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMSigningCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMSigningCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el certificado de firma con el ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** del IAM usuario nombrado**`Bob`**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Para API obtener más información, consulte la referencia DeleteSigningCertificatedel AWS Tools for PowerShell cmdlet.

`Remove-IAMUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMUser`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el IAM usuario nombrado**Bob**.

```
Remove-IAMUser -UserName Bob
```

Ejemplo 2: en este ejemplo se elimina el nombre IAM de usuario **`Theresa`** junto con los elementos que deban eliminarse primero.

```
$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
```

- Para API obtener más información, consulte la referencia DeleteUserdel AWS Tools for PowerShell cmdlet.

`Remove-IAMUserFromGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMUserFromGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina al IAM usuario **`Bob`** del grupo**`Testers`**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Ejemplo 2: Este ejemplo busca cualquier grupo del que el IAM usuario **`Theresa`** sea miembro y, a continuación, lo elimina **`Theresa`** de esos grupos.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Ejemplo 3: En este ejemplo se muestra una forma alternativa de eliminar al IAM usuario **`Bob`** del **`Testers`** grupo.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- Para API obtener más información, consulte RemoveUserFromGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMUserPermissionsBoundary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMUserPermissionsBoundary`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo eliminar el límite de permisos asociado a un IAM usuario.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Para API obtener más información, consulte DeleteUserPermissionsBoundary AWS Tools for PowerShellCmdlet Reference.

`Remove-IAMUserPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMUserPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el nombre de la política en línea **`AccessToEC2Policy`** que está incrustada en el nombre de IAM usuario. **`Bob`**

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Ejemplo 2: Este ejemplo busca todas las políticas en línea que están integradas en el nombre de IAM usuario **`Theresa`** y, a continuación, las elimina.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Para API obtener más información, consulte Cmdlet DeleteUserPolicyReference AWS Tools for PowerShell .

`Remove-IAMUserTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMUserTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la etiqueta del usuario llamado “joe” con las claves de etiqueta “abac” y “xyzw”. Para eliminar varias etiquetas, proporcione una lista de claves de etiquetas separadas por comas.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Para API obtener más información, consulte UntagUserla referencia de AWS Tools for PowerShell cmdlets.

`Remove-IAMVirtualMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-IAMVirtualMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina el MFA dispositivo IAM virtual cuyo ARN es**`arn:aws:iam::123456789012:mfa/bob`**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Ejemplo 2: Este ejemplo comprueba si el IAM usuario Theresa tiene un MFA dispositivo asignado. Si se encuentra uno, el dispositivo está deshabilitado para el IAM usuario. Si el dispositivo es virtual, también se elimina.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- Para API obtener más información, consulte DeleteVirtualMfaDevicela Referencia de AWS Tools for PowerShell cmdlets.

`Request-IAMCredentialReport`

En el siguiente ejemplo de código se muestra cómo usarlo. `Request-IAMCredentialReport`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se solicita la generación de un nuevo informe, lo que se puede hacer cada cuatro horas. Si el último informe aún es reciente, el campo Estado indica **`COMPLETE`**. Uso de **`Get-IAMCredentialReport`** para ver el informe completo.

```
Request-IAMCredentialReport
```

Salida:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Para API obtener más información, consulte GenerateCredentialReportla referencia de AWS Tools for PowerShell cmdlets.

`Request-IAMServiceLastAccessedDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Request-IAMServiceLastAccessedDetail`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo es un cmdlet equivalente a. GenerateServiceLastAccessedDetails API Esto proporciona un identificador de trabajo que se puede usar en Get-IAMServiceLastAccessedDetail y Get- IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Para API obtener más información, consulte GenerateServiceLastAccessedDetails AWS Tools for PowerShellCmdlet Reference.

`Set-IAMDefaultPolicyVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-IAMDefaultPolicyVersion`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se establece la **v2** versión de la política **arn:aws:iam::123456789012:policy/MyPolicy** que ARN es la versión activa predeterminada.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- Para API obtener más información, consulte SetDefaultPolicyVersion AWS Tools for PowerShellCmdlet Reference.

`Set-IAMRolePermissionsBoundary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-IAMRolePermissionsBoundary`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo establecer el límite de permisos para un IAM rol. Puede establecer políticas AWS administradas o políticas personalizadas como límite de permisos.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para API obtener más información, consulte PutRolePermissionsBoundaryla referencia de AWS Tools for PowerShell cmdlets.

`Set-IAMUserPermissionsBoundary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-IAMUserPermissionsBoundary`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra cómo establecer el límite de permisos para el usuario. Puede establecer políticas AWS administradas o políticas personalizadas como límite de permisos.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para API obtener más información, consulte PutUserPermissionsBoundaryla referencia de AWS Tools for PowerShell cmdlets.

`Sync-IAMMFADevice`

En el siguiente ejemplo de código se muestra cómo usarlo. `Sync-IAMMFADevice`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo sincroniza el MFA dispositivo que está asociado al IAM usuario **`Bob`** y cuyo dispositivo ARN está asociado **`arn:aws:iam::123456789012:mfa/bob`** con un programa de autenticación que proporcionó los dos códigos de autenticación.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Ejemplo 2: Este ejemplo sincroniza el IAM MFA dispositivo que está asociado al IAM usuario **`Theresa`** con un dispositivo físico que tiene el número de serie **`ABCD12345678`** y que proporciona los dos códigos de autenticación.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Para API obtener más información, consulte ResyncMfaDevice AWS Tools for PowerShellCmdlet Reference.

`Unregister-IAMGroupPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-IAMGroupPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se separa la política de grupo administrada ARN que **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** proviene del grupo denominado**`Testers`**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Ejemplo 2: en este ejemplo se busca todas las políticas administradas que están asociadas al grupo denominado **`Testers`** y se las desvincula del grupo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Para API obtener más información, consulte la referencia DetachGroupPolicyde AWS Tools for PowerShell cmdlets.

`Unregister-IAMRolePolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-IAMRolePolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se separa la política de grupo gestionado que ARN proviene **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** de la función denominada**`FedTesterRole`**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Ejemplo 2: en este ejemplo se busca todas las políticas administradas que están asociadas al rol denominado **`FedTesterRole`** y se desvinculan del rol.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- Para API obtener más información, consulte la referencia DetachRolePolicyde AWS Tools for PowerShell cmdlets.

`Unregister-IAMUserPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-IAMUserPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se separa la política administrada ARN que **`arn:aws:iam::123456789012:policy/TesterPolicy`** proviene del IAM usuario nombrado**`Bob`**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

Ejemplo 2: Este ejemplo busca todas las políticas administradas que están asociadas al IAM usuario nombrado **`Theresa`** y las separa del usuario.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- Para API obtener más información, consulte la referencia DetachUserPolicydel AWS Tools for PowerShell cmdlet.

`Update-IAMAccessKey`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMAccessKey`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el estado de la clave **`AKIAIOSFODNN7EXAMPLE`** de acceso del IAM usuario denominado **`Bob`** a**`Inactive`**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Para API obtener más información, consulte UpdateAccessKey AWS Tools for PowerShellCmdlet Reference.

`Update-IAMAccountPasswordPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMAccountPasswordPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza la política de contraseñas de la cuenta con la configuración especificada. Tenga en cuenta que los parámetros que no estén incluidos en el comando no se dejan sin modificar. En su lugar, se restablecen a los valores predeterminados.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Para API obtener más información, consulte UpdateAccountPasswordPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMAssumeRolePolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMAssumeRolePolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza el IAM rol nombrado **`ClientRole`** con una nueva política de confianza, cuyo contenido proviene del archivo**`ClientRolePolicy.json`**. Tenga en cuenta que debe utilizar el parámetro **`-Raw`** switch para procesar correctamente el contenido del JSON archivo.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Para API obtener más información, consulte UpdateAssumeRolePolicyla referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el nombre del IAM grupo **`Testers`** a**`AppTesters`**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Ejemplo 2: En este ejemplo se cambia la ruta del IAM grupo **`AppTesters`** a**`/Org1/Org2/`**. Esto cambia ARN la del grupo a**`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Para API obtener más información, consulte UpdateGroup AWS Tools for PowerShellCmdlet Reference.

`Update-IAMLoginProfile`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMLoginProfile`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo establece una nueva contraseña temporal para el IAM usuario **`Bob`** y requiere que el usuario cambie la contraseña la próxima vez que inicie sesión.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- Para API obtener más información, consulte UpdateLoginProfile AWS Tools for PowerShellCmdlet Reference.

`Update-IAMOpenIDConnectProviderThumbprint`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMOpenIDConnectProviderThumbprint`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza la lista de huellas digitales de certificados del OIDC proveedor **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** que ARN va a utilizar una nueva huella digital. El OIDC proveedor comparte el nuevo valor cuando cambia el certificado asociado al proveedor.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Para API obtener más información, consulte UpdateOpenIdConnectProviderThumbprintla referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza la descripción del rol y el valor de duración máxima de la sesión (en segundos) para el que se puede solicitar la sesión de un rol.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Para API obtener más información, consulte UpdateRolela referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMRoleDescription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMRoleDescription`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza la descripción de un IAM rol de tu cuenta.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Para API obtener más información, consulte UpdateRoleDescription AWS Tools for PowerShellCmdlet Reference.

`Update-IAMSAMLProvider`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMSAMLProvider`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza el SAML proveedor en IAM el ARN que se encuentra **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** con un nuevo documento

de SAML metadatos del archivo**SAMLMetaData.xml**. Tenga en cuenta que debe utilizar el parámetro **-Raw** switch para procesar correctamente el contenido del JSON archivo.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Para API obtener más información, consulte UpdateSamlProviderla referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMServerCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMServerCertificate`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se cambia el nombre del certificado denominado **MyServerCertificate** a **MyRenamedServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Ejemplo 2: en este ejemplo se mueve el certificado denominado **MyServerCertificate** a la ruta /Org1/Org2/. Esto cambia el ARN para el recurso a**arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Para API obtener más información, consulte UpdateServerCertificatela referencia del AWS Tools for PowerShell cmdlet.

`Update-IAMSigningCertificate`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMSigningCertificate`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza el certificado que está asociado al IAM usuario nombrado **Bob** y cuyo identificador de certificado es **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** para marcarlo como inactivo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Para API obtener más información, consulte UpdateSigningCertificatela referencia de AWS Tools for PowerShell cmdlets.

`Update-IAMUser`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-IAMUser`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el nombre del IAM usuario **Bob** a**Robert**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Ejemplo 2: En este ejemplo se cambia la ruta del IAM usuario **Bob** a**/Org1/Org2/**, lo que, de hecho, cambia la ruta del ARN usuario a**arn:aws:iam::123456789012:user/Org1/Org2/bob**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Para API obtener más información, consulte UpdateUserla referencia del AWS Tools for PowerShell cmdlet.

`Write-IAMGroupPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-IAMGroupPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una política en línea denominada **AppTesterPolicy** y se incrusta en el IAM grupo. **AppTesters** Si ya existe una política incrustada con el mismo nombre,

se sobrescribirá. El contenido JSON de la política viene del archivo. **apptesterpolicy.json** Tenga en cuenta que debe usar el **-Raw** parámetro para procesar correctamente el contenido del JSON archivo.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Para API obtener más información, consulte PutGroupPolicy AWS Tools for PowerShellCmdlet Reference.

Write-IAMRolePolicy

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-IAMRolePolicy`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo, se crea una política en línea denominada **FedTesterRolePolicy** y se incrusta en el IAM rol. **FedTesterRole** Si ya existe una política incrustada con el mismo nombre, se sobrescribirá. El contenido JSON de la política proviene del archivo. **FedTesterPolicy.json** Tenga en cuenta que debe usar el **-Raw** parámetro para procesar correctamente el contenido del JSON archivo.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Para API obtener más información, consulte PutRolePolicy AWS Tools for PowerShellCmdlet Reference.

Write-IAMUserPolicy

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-IAMUserPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una política en línea denominada **EC2AccessPolicy** y se incrusta en el IAM usuario. **Bob** Si ya existe una política incrustada con el mismo nombre, se sobrescribirá. El contenido JSON de la política proviene del archivo. **EC2AccessPolicy.json** Tenga en cuenta que debe usar el **-Raw** parámetro para procesar correctamente el contenido del JSON archivo.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- Para API obtener más información, consulte PutUserPolicy AWS Tools for PowerShellCmdlet Reference.

Ejemplos de Kinesis con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS Tools for PowerShell uso de Kinesis.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-KINRecord`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-KINRecord`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo devolver y extraer datos de una serie de uno o más registros. El iterador suministrado a Get- KINRecord determina la posición inicial de los registros que se van a devolver, que en este ejemplo se capturan en una variable, $records. A continuación, se puede acceder a cada registro individual indexando la colección $records. Suponiendo que los datos del registro son texto codificado en formato UTF -8, el comando final muestra cómo se pueden extraer los datos del objeto y devolverlos como texto a la consola. MemoryStream

```
$records
```

```
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Salida:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Salida:

```
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())
```

Salida:

```
test data from string
```

- Para API obtener más información, consulte GetRecords AWS Tools for PowerShellCmdlet Reference.

`Get-KINShardIterator`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-KINShardIterator`

Herramientas para PowerShell

Ejemplo 1: Devuelve un iterador de fragmentos para el fragmento especificado y su posición inicial. Los detalles de los identificadores de los fragmentos y los números de secuencia se pueden obtener en el resultado del KINStream cmdlet Get-, haciendo referencia a la colección Shards del objeto de flujo devuelto. El iterador devuelto se puede usar con el KINRecord cmdlet Get- para extraer los registros de datos del fragmento.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Salida:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Para obtener API más información, consulte GetShardIteratorla referencia del cmdlet.AWS Tools for PowerShell

`Get-KINStream`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-KINStream`

Herramientas para PowerShell

Ejemplo 1: Devuelve los detalles del flujo especificado.

```
Get-KINStream -StreamName "mystream"
```

Salida:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Para API obtener más información, consulte DescribeStream AWS Tools for PowerShellCmdlet Reference.

`New-KINStream`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-KINStream`

Herramientas para PowerShell

Ejemplo 1: Crea una nueva transmisión. De forma predeterminada, este cmdlet no devuelve ningún resultado, por lo que se agrega el PassThru modificador - para devolver el valor proporcionado al StreamName parámetro - para su uso posterior.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Para API obtener más información, consulte la referencia del CreateStream AWS Tools for PowerShellcmdlet.

`Remove-KINStream`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-KINStream`

Herramientas para PowerShell

Ejemplo 1: Elimina la transmisión especificada. Se le solicitará una confirmación antes de ejecutar el comando. Para suprimir las solicitudes de confirmación, utilice el conmutador -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteStreamReference.

`Write-KINRecord`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-KINRecord`

Herramientas para PowerShell

Ejemplo 1: escribe un registro que contiene la cadena proporcionada al parámetro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

Ejemplo 2: escribe un registro que contiene los datos contenidos en el archivo especificado. El archivo se trata como una secuencia de bytes, por lo que si contiene texto, debe escribirse con la codificación necesaria antes de usarlo con este cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Para API obtener más información, consulte la referencia del PutRecord AWS Tools for PowerShellcmdlet.

Ejemplos de Lambda con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante Lambda.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-LMResourceTag`

En el siguiente ejemplo de código se muestra cómo usarloAdd-LMResourceTag.

Herramientas para PowerShell

Ejemplo 1: agrega las tres etiquetas (Washington, Oregón y California) y sus valores asociados a la función especificada identificada por ellaARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- Para API obtener más información, consulte TagResource AWS Tools for PowerShellCmdlet Reference.

`Get-LMAccountSetting`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMAccountSetting`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo, se muestra para comparar el límite de la cuenta y el uso de la cuenta

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Salida:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Para API obtener más información, consulte GetAccountSettingsla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMAlias`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo, se recuperan las ponderaciones de Routing Config para un alias de función de Lambda específico.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Salida:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Para API obtener más información, consulte GetAliasla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMFunctionConcurrency`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMFunctionConcurrency`

Herramientas para PowerShell

Ejemplo 1: este ejemplo obtiene la simultaneidad reservada para la función de Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Salida:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Para API obtener más información, consulte GetFunctionConcurrencyla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMFunctionConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMFunctionConfiguration`

Herramientas para PowerShell

Ejemplo 1: este ejemplo devuelve la configuración específica de la versión de una función de Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Salida:

```
CodeSha256                : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                  : 1426
DeadLetterConfig          : Amazon.Lambda.Model.DeadLetterConfig
Description               : Verson 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
```

- Para API obtener más información, consulte GetFunctionConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMFunctionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMFunctionList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran todas las funciones de Lambda con un tamaño de código ordenado

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Salida:

```
FunctionName                                                Runtime   Timeout
 CodeSize
------------                                                -------   -------
 --------
```

```
test                                                    python2.7       3
 243
MylambdaFunction123                                     python3.8     600
 659
myfuncpython1                                           python3.8     303
 675
```

- Para API obtener más información, consulte ListFunctionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMPolicy`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra la política de funciones de la función de Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Salida:

```
{"Version":"2012-10-17","Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:
east-1:123456789102:function:test"}]}
```

- Para API obtener más información, consulte GetPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMProvisionedConcurrencyConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMProvisionedConcurrencyConfig`

Herramientas para PowerShell

Ejemplo 1: este ejemplo obtiene la configuración de simultaneidad aprovisionada para el alias especificado de la función de Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Salida:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Para API obtener más información, consulte GetProvisionedConcurrencyConfigla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMProvisionedConcurrencyConfigList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMProvisionedConcurrencyConfigList`

Herramientas para PowerShell

Ejemplo 1: este ejemplo obtiene la configuración de simultaneidad aprovisionada para una función de Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Para API obtener más información, consulte ListProvisionedConcurrencyConfigsla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMResourceTag`

Herramientas para PowerShell

Ejemplo 1: recupera las etiquetas y sus valores actualmente configurados en la función especificada.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Salida:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Para API obtener más información, consulte ListTagsla referencia de AWS Tools for PowerShell cmdlets.

`Get-LMVersionsByFunction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-LMVersionsByFunction`

Herramientas para PowerShell

Ejemplo 1: este ejemplo devuelve la lista de configuraciones específicas de la versión para cada versión de la función de Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Salida:

```
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
```

- Para API obtener más información, consulte ListVersionsByFunctionla referencia de AWS Tools for PowerShell cmdlets.

`New-LMAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-LMAlias`

Herramientas para PowerShell

Ejemplo 1: este ejemplo crea un nuevo alias de Lambda para una versión y configuración de enrutamiento específicas a fin de especificar el porcentaje de solicitudes de invocación que recibe.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Para API obtener más información, consulte CreateAliasla referencia de AWS Tools for PowerShell cmdlets.

`Publish-LMFunction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Publish-LMFunction`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva función de C# (tiempo de ejecución de dotnetcore1.0) denominada MyFunction AWS Lambda, que proporciona los binarios compilados para la función desde un archivo zip del sistema de archivos local (se pueden utilizar rutas relativas o absolutas). Las funciones Lambda de C# especifican el controlador de la función mediante la designación: :Namespace. AssemblyName ClassName::. MethodName Debe reemplazar adecuadamente las partes del nombre del ensamblado (sin el sufijo .dll), el espacio de nombres, el nombre de la clase y el nombre del método de la especificación del controlador. La nueva función tendrá las variables de entorno 'envvar1' y 'envvar2' configuradas a partir de los valores proporcionados.

```
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" }
```

Salida:

```
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        :
```

Ejemplo 2: este ejemplo es similar al anterior, excepto que los binarios de la función se cargan primero en un bucket de Amazon S3 (que debe estar en la misma región que la función de Lambda prevista) y, a continuación, se hace referencia al objeto de S3 resultante al crear la función.

```
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" }
```

- Para API obtener más información, consulte CreateFunctionla referencia de AWS Tools for PowerShell cmdlets.

`Publish-LMVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Publish-LMVersion`

Herramientas para PowerShell

Ejemplo 1: este ejemplo crea una versión para la instantánea existente del código de función de Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Para API obtener más información, consulte PublishVersionla referencia de AWS Tools for PowerShell cmdlets.

`Remove-LMAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMAlias`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el alias de la función de Lambda mencionado en el comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Para API obtener más información, consulte DeleteAliasla referencia de AWS Tools for PowerShell cmdlets.

`Remove-LMFunction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMFunction`

Herramientas para PowerShell

Ejemplo 1: este ejemplo elimina una versión específica de una función de Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Para API obtener más información, consulte DeleteFunctionla referencia de AWS Tools for PowerShell cmdlets.

`Remove-LMFunctionConcurrency`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMFunctionConcurrency`

Herramientas para PowerShell

Ejemplo 1: este ejemplo elimina la simultaneidad de funciones de la función de Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Para API obtener más información, consulte DeleteFunctionConcurrencyla referencia de AWS Tools for PowerShell cmdlets.

`Remove-LMPermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMPermission`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la política de función para la función Lambda especificada StatementId .

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- Para API obtener más información, consulte RemovePermission AWS Tools for PowerShellCmdlet Reference.

`Remove-LMProvisionedConcurrencyConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMProvisionedConcurrencyConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la configuración de simultaneidad aprovisionada para un alias específico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- Para API obtener más información, consulte DeleteProvisionedConcurrencyConfigla referencia de AWS Tools for PowerShell cmdlets.

`Remove-LMResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-LMResourceTag`

Herramientas para PowerShell

Ejemplo 1: elimina las etiquetas suministradas de una función. El cmdlet solicitará confirmación antes de continuar, a menos que se especifique el modificador -Force. Se realiza una sola llamada al servicio para eliminar las etiquetas.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Ejemplo 2: elimina las etiquetas suministradas de una función. El cmdlet solicitará confirmación antes de continuar, a menos que se especifique el modificador -Force. Una vez realizada la llamada al servicio por etiqueta suministrada.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Para API obtener más información, consulte UntagResourcela referencia de AWS Tools for PowerShell cmdlets.

`Update-LMAlias`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-LMAlias`

Herramientas para PowerShell

Ejemplo 1: este ejemplo actualiza la configuración de un alias de función de Lambda existente. Actualiza el RoutingConfiguration valor para transferir el 60% (0.6) del tráfico a la versión 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- Para API obtener más información, consulte UpdateAliasla referencia de AWS Tools for PowerShell cmdlets.

`Update-LMFunctionCode`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-LMFunctionCode`

Herramientas para PowerShell

Ejemplo 1: actualiza la función denominada 'MyFunction' con el nuevo contenido contenido en el archivo zip especificado. Para un C#. NETFunción Lambda básica: el archivo zip debe contener el ensamblaje compilado.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Ejemplo 2: este ejemplo es similar al anterior, pero utiliza un objeto de Amazon S3 que contiene el código actualizado para actualizar la función.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key
 UpdatedCode.zip
```

- Para API obtener más información, consulte la referencia UpdateFunctionCodede AWS Tools for PowerShell cmdlets.

`Update-LMFunctionConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-LMFunctionConfiguration`

Herramientas para PowerShell

Ejemplo 1: este ejemplo actualiza la configuración de la función de Lambda existente

```
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
```

- Para API obtener más información, consulte UpdateFunctionConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Write-LMFunctionConcurrency`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-LMFunctionConcurrency`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se aplica la configuración de simultaneidad de la función en su conjunto.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Para API obtener más información, consulte PutFunctionConcurrencyla referencia de AWS Tools for PowerShell cmdlets.

`Write-LMProvisionedConcurrencyConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-LMProvisionedConcurrencyConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agrega una configuración de simultaneidad aprovisionada al alias de una función

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Para API obtener más información, consulte PutProvisionedConcurrencyConfigla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Amazon ML que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante Amazon ML.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-MLBatchPrediction`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-MLBatchPrediction`.

Herramientas para PowerShell

Ejemplo 1: Devuelve los metadatos detallados de una predicción de lote con un identificador.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Para API obtener más información, consulte GetBatchPrediction AWS Tools for PowerShellCmdlet Reference.

`Get-MLBatchPredictionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLBatchPredictionList`

Herramientas para PowerShell

Ejemplo 1: devuelve una lista de todos los registros de datos BatchPredictions y sus registros de datos asociados que coinciden con el criterio de búsqueda indicado en la solicitud.

```
Get-MLBatchPredictionList
```

Ejemplo 2: Devuelve una lista de todos BatchPredictions con un estado deCOMPLETED.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Para API obtener más información, consulte DescribeBatchPredictions AWS Tools for PowerShellCmdlet Reference.

`Get-MLDataSource`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLDataSource`

Herramientas para PowerShell

Ejemplo 1: Devuelve los metadatos, el estado y la información del archivo de datos de un DataSource con el identificador

```
Get-MLDataSource -DataSourceId ID
```

- Para API obtener más información, consulte GetDataSourcela referencia del AWS Tools for PowerShell cmdlet.

`Get-MLDataSourceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLDataSourceList`

Herramientas para PowerShell

Ejemplo 1: devuelve una lista de todos los registros de datos DataSources y sus registros de datos asociados.

```
Get-MLDataSourceList
```

Ejemplo 2: Devuelve una lista de todos DataSources con un estado deCOMPLETED.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Para API obtener más información, consulte DescribeDataSources AWS Tools for PowerShellCmdlet Reference.

`Get-MLEvaluation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLEvaluation`

Herramientas para PowerShell

Ejemplo 1: Devuelve los metadatos y el estado de una evaluación con un identificador.

```
Get-MLEvaluation -EvaluationId ID
```

- Para API obtener más información, consulte GetEvaluationla referencia del AWS Tools for PowerShell cmdlet.

`Get-MLEvaluationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLEvaluationList`

Herramientas para PowerShell

Ejemplo 1: Devuelve una lista de todos los recursos de evaluación

```
Get-MLEvaluationList
```

Ejemplo 2: Devuelve una lista de todas las evaluaciones con un estado de. COMPLETED

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeEvaluationsReference.

`Get-MLModel`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLModel`

Herramientas para PowerShell

Ejemplo 1: Devuelve la información detallada de los metadatos, el estado, el esquema y el archivo de datos de un archivo MLModel con ID.

```
Get-MLModel -ModelId ID
```

- Para API obtener más información, consulte G etMLModel en la referencia del AWS Tools for PowerShell cmdlet.

`Get-MLModelList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLModelList`

Herramientas para PowerShell

Ejemplo 1: Devuelve una lista de todos los modelos y sus registros de datos asociados.

```
Get-MLModelList
```

Ejemplo 2: Devuelve una lista de todos los modelos con un estado deCOMPLETED.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Para API obtener más información, consulte D escribeMLModels en la referencia del AWS Tools for PowerShell cmdlet.

`Get-MLPrediction`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-MLPrediction`

Herramientas para PowerShell

Ejemplo 1: enviar un registro al punto final URL de predicción en tiempo real del modelo con un ID de identificación.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Para API obtener más información, consulte Predict in AWS Tools for PowerShell Cmdlet Reference.

`New-MLBatchPrediction`

El siguiente ejemplo de código muestra cómo usarlo. `New-MLBatchPrediction`

Herramientas para PowerShell

Ejemplo 1: Cree una nueva solicitud de predicción por lotes para el modelo con un identificador y coloque la salida en la ubicación S3 especificada.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Para API obtener más información, consulte CreateBatchPredictionla referencia del AWS Tools for PowerShell cmdlet.

`New-MLDataSourceFromS3`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-MLDataSourceFromS3`

Herramientas para PowerShell

Ejemplo 1: Cree una fuente de datos con datos para una ubicación de S3, con un nombre NAME y un esquema deSCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- Para API obtener más información, consulte CreateDataSourceFromS3 en la referencia de AWS Tools for PowerShell cmdlets.

`New-MLEvaluation`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-MLEvaluation`

Herramientas para PowerShell

Ejemplo 1: Crear una evaluación para un identificador de fuente de datos y un identificador de modelo determinados

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Para API obtener más información, consulte CreateEvaluationla referencia del AWS Tools for PowerShell cmdlet.

`New-MLModel`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-MLModel`

Herramientas para PowerShell

Ejemplo 1: Crear un nuevo modelo con datos de entrenamiento.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Para API obtener más información, consulte C reateMLModel en la referencia del AWS Tools for PowerShell cmdlet.

`New-MLRealtimeEndpoint`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-MLRealtimeEndpoint`

Herramientas para PowerShell

Ejemplo 1: Cree un nuevo punto final de predicción en tiempo real para el identificador del modelo dado.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Para API obtener más información, consulte la referencia CreateRealtimeEndpointdel AWS Tools for PowerShell cmdlet.

Ejemplos de Macie que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS Tools for PowerShell uso de Macie.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde encontrará instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-MAC2FindingList`

En el siguiente ejemplo de código se muestra cómo usarlo`Get-MAC2FindingList`.

Herramientas para PowerShell

Ejemplo 1: Devuelve una lista FindingIds de hallazgos que contienen una detección de datos confidenciales del tipo "CREDIT_ CARD _NUMBER" o «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
```

- Para API obtener más información, consulte la referencia ListFindingsdel AWS Tools for PowerShell cmdlet.

AWS OpsWorks ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS OpsWorks.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`New-OPSDeployment`

En el siguiente ejemplo de código se muestra cómo usarlo`New-OPSDeployment`.

Herramientas para PowerShell

Ejemplo 1: Este comando crea una nueva implementación de aplicaciones en todas las instancias basadas en Linux de una capa de Stacks. AWS OpsWorks Incluso si especificas un ID de capa, también debes especificar un ID de pila. El comando permite que la implementación reinicie las instancias si es necesario.

```
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z" -LayerId
 "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -AppId "0f7a109c-bf68-4336-8cb9-
d37fe0b8c61d" -Command_Name deploy -Command_Arg @{Name="allow_reboot";Value="true"}
```

Ejemplo 2: Este comando despliega la **`appsetup`** receta del **`phpapp`** libro de cocina y la **`secbaseline`** receta del libro de cocina. **`testcookbook`** El objetivo de despliegue es una instancia, pero también se requieren el ID de pila y el ID de capa. El **`allow_reboot`** atributo del parámetro Command_Arg está establecido en**`true`**, lo que permite que la implementación reinicie las instancias si es necesario.

```
$commandArgs = '{ "Name":"execute_recipes", "Args"{ "recipes":
["phpapp::appsetup","testcookbook::secbaseline"] } }'
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z"
 -LayerId "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -InstanceId
 "d89a6118-0007-4ccf-a51e-59f844127021" -Command_Name $commandArgs -Command_Arg
 @{Name="allow_reboot";Value="true
```

- Para API obtener más información, consulte la referencia del CreateDeploymentcmdlet AWS Tools for PowerShell .

Lista de precios de AWS ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with Lista de precios de AWS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-PLSAttributeValue`

El siguiente ejemplo de código muestra cómo usarlo`Get-PLSAttributeValue`.

Herramientas para PowerShell

Ejemplo 1: devuelve los valores del atributo 'volumeType' de Amazon EC2 en la región us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

Salida:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Para API obtener más información, consulte la referencia del GetAttributeValues AWS Tools for PowerShellcmdlet.

`Get-PLSProduct`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-PLSProduct`

Herramientas para PowerShell

Ejemplo 1: Detalles de devolución de todos los productos de AmazonEC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Salida:

```
{"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"},...
```

Ejemplo 2: Devuelve los datos de Amazon de la EC2 región us-east-1 filtrados por tipos de volumen de «Uso general» respaldados por. SSD

```
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
```

Salida:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Para obtener API más información, consulte Cmdlet Reference. GetProductsAWS Tools for PowerShell

`Get-PLSService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-PLSService`

Herramientas para PowerShell

Ejemplo 1: devuelve los metadatos de todos los códigos de servicio disponibles en la región us-east-1.

```
Get-PLSService -Region us-east-1
```

Salida:

```
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
...
```

Ejemplo 2: Devuelve los metadatos del EC2 servicio de Amazon en la región us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Salida:

```
AttributeNames                                                                ServiceCode
--------------                                                                -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Para API obtener más información, consulte la referencia de DescribeServices AWS Tools for PowerShellcmdlets.

Ejemplos de Resource Groups que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS Tools for PowerShell with Resource Groups.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-RGResourceTag`

El siguiente ejemplo de código muestra cómo usarloAdd-RGResourceTag.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se agrega la clave de etiqueta «Instances» con el valor «workboxes» al grupo de recursos dado arn

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Salida:

```
Arn                                                          Tags
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Para API obtener más información, consulte Etiqueta en la referencia del AWS Tools for PowerShell cmdlet.

`Find-RGResource`

El siguiente ejemplo de código muestra cómo usarlo. `Find-RGResource`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un tipo de recurso de ResourceQuery for Instance con filtros de etiquetas y busca recursos.

```
$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
```

Salida:

```
ResourceArn                                                  ResourceType
-----------                                                  ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- Para API obtener más información, consulte SearchResourcesla referencia de AWS Tools for PowerShell cmdlets.

`Get-RGGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera el grupo de recursos según el nombre del grupo

```
Get-RGGroup -GroupName auto-no
```

Salida:

```
Description GroupArn                                                        Name
----------- --------                                                        ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Para API obtener más información, consulte GetGroup AWS Tools for PowerShellCmdlet Reference.

`Get-RGGroupList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGGroupList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra un grupo de recursos ya creado.

```
Get-RGGroupList
```

Salida:

```
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
```

- Para API obtener más información, consulte ListGroups AWS Tools for PowerShellCmdlet Reference.

`Get-RGGroupQuery`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGGroupQuery`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo busca la consulta de recursos para el grupo de recursos dado

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Salida:

```
Query
         Type
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- Para API obtener más información, consulte GetGroupQuery AWS Tools for PowerShellCmdlet Reference.

`Get-RGGroupResourceList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGGroupResourceList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los recursos del grupo en función de los filtrados por tipo de recurso

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Salida:

```
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
```

- Para API obtener más información, consulte ListGroupResources AWS Tools for PowerShellCmdlet Reference.

`Get-RGResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGResourceTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las etiquetas del grupo de recursos dado arn

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Salida:

```
Key       Value
---       -----
Instances workboxes
```

- Para API obtener más información, consulte la referencia GetTagsdel AWS Tools for PowerShell cmdlet.

`New-RGGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-RGGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo grupo de AWS recursos Resource Groups basado en etiquetas denominado TestPowerShellGroup. El grupo incluye las EC2 instancias de Amazon de la región actual que están etiquetadas con la clave de etiqueta «Nombre» y el valor de etiqueta «test2". El comando devuelve la consulta y el tipo de grupo, así como los resultados de la operación.

```
$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."
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-RGGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-RGGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el grupo de recursos nombrado

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-RGResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-RGResourceTag`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la etiqueta mencionada del grupo de recursos

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Salida:

```
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}
```

- Para API obtener más información, consulte Eliminar la etiqueta en la referencia del AWS Tools for PowerShell cmdlet.

`Update-RGGroup`

El siguiente ejemplo de código muestra cómo usarlo. `Update-RGGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se actualiza la descripción del grupo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Salida:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- Para API obtener más información, consulte UpdateGroup AWS Tools for PowerShellCmdlet Reference.

`Update-RGGroupQuery`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-RGGroupQuery`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un objeto de consulta y se actualiza la consulta del grupo.

```
$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
```

Salida:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Para API obtener más información, consulte UpdateGroupQuery AWS Tools for PowerShellCmdlet Reference.

Resource Groups: API ejemplos de etiquetado mediante herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS Tools for PowerShell with Resource Groups TaggingAPI.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-RGTResourceTag`

El siguiente ejemplo de código muestra cómo usarloAdd-RGTResourceTag.

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se añaden las claves de etiqueta «stage» y «version» con los valores «beta» y «preprod_test» a un bucket de Amazon S3 y a una tabla de Amazon DynamoDB. Se realiza una sola llamada al servicio para aplicar las etiquetas.

```
$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" }
```

Ejemplo 2: en este ejemplo se añaden las etiquetas y los valores especificados a un bucket de Amazon S3 y a una tabla de Amazon DynamoDB. Se realizan dos llamadas al servicio, una para cada recurso ARN canalizado al cmdlet.

```
$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" }
```

- Para API obtener más información, consulte la referencia del TagResourcescmdlet AWS Tools for PowerShell .

`Get-RGTResource`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGTResource`

Herramientas para PowerShell

Ejemplo 1: devuelve todos los recursos etiquetados de una región y las claves de etiqueta asociadas al recurso. Si no se proporciona ningún parámetro -Region al cmdlet, se intentará deducir la región a partir de los metadatos del shell o de la instancia. EC2

```
Get-RGTResource
```

Salida:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

Ejemplo 2: devuelve todos los recursos etiquetados del tipo especificado en una región. La cadena de cada nombre de servicio y tipo de recurso es la misma que la incrustada en el Amazon Resource Name (ARN) de un recurso.

```
Get-RGTResource -ResourceType "s3"
```

Salida:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

Ejemplo 3: Devuelve todos los recursos etiquetados del tipo especificado en una región. Tenga en cuenta que cuando los tipos de recursos se canalizan al cmdlet, se realiza una llamada al servicio por cada tipo de recurso suministrado.

```
"dynamodb","s3" | Get-RGTResource
```

Salida:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

Ejemplo 4: devuelve todos los recursos etiquetados que coinciden con el filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Salida:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

Ejemplo 5: Devuelve todos los recursos etiquetados que coinciden con el filtro y el tipo de recurso especificados.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Salida:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

Ejemplo 6: Devuelve todos los recursos etiquetados que coinciden con el filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Salida:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- Para API obtener más información, consulte GetResources AWS Tools for PowerShellCmdlet Reference.

`Get-RGTTagKey`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGTTagKey`

Herramientas para PowerShell

Ejemplo 1: devuelve todas las claves de etiquetas de la región especificada. Si no se especifica el parámetro -Region, el cmdlet intentará deducir la región a partir de los metadatos predeterminados de la instancia o región del shell. EC2 Tenga en cuenta que las claves de las etiquetas no se devuelven en ningún orden específico.

```
Get-RGTTagKey -region us-west-2
```

Salida:

```
version
stage
```

- Para API obtener más información, consulte GetTagKeysla referencia del AWS Tools for PowerShell cmdlet.

`Get-RGTTagValue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-RGTTagValue`

Herramientas para PowerShell

Ejemplo 1: devuelve el valor de la etiqueta especificada en una región. Si no se especifica el parámetro -Region, el cmdlet intentará deducir la región a partir de los metadatos predeterminados de la región o instancia del shell. EC2

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Salida:

```
beta
```

- Para obtener API más información, consulte GetTagValuesCmdlet Reference.AWS Tools for PowerShell

`Remove-RGTResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-RGTResourceTag`

Herramientas para PowerShell

Ejemplo 1: elimina las claves de etiqueta «stage» y «version», así como los valores asociados, de un bucket de Amazon S3 y una tabla de Amazon DynamoDB. Se realiza una sola llamada al servicio para eliminar las etiquetas. Antes de eliminar las etiquetas, el cmdlet solicitará una confirmación. Para omitir la confirmación, agregue el parámetro -Force.

```
$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"
```

Ejemplo 2: elimina las claves de etiqueta «stage» y «version», así como los valores asociados, de un bucket de Amazon S3 y una tabla de Amazon DynamoDB. Se realizan dos llamadas al servicio, una para cada recurso ARN canalizado al cmdlet. Antes de cada llamada, el cmdlet solicitará la confirmación. Para omitir la confirmación, agregue el parámetro -Force.

```
$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"
```

- Para API obtener más información, consulte UntagResources AWS Tools for PowerShellCmdlet Reference.

Ejemplos de Route 53 con herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell con Route 53.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Edit-R53ResourceRecordSet`

El siguiente ejemplo de código muestra cómo usarlo`Edit-R53ResourceRecordSet`.

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un registro A para www.example.com y se cambia el registro A de test.example.com de 192.0.2.3 a 192.0.2.1. Tenga en cuenta que los valores de los registros de tipo cambio deben estar entre comillas dobles. TXT Consulte la documentación de Amazon Route 53 para obtener más información. Puede usar el Get-R53Change cmdlet para sondear y determinar cuándo se han completado los cambios.

```
$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
```

Ejemplo 2: en este ejemplo se muestra cómo crear conjuntos de registros de recursos de alias. «Z222222222» es el ID de la zona alojada de Amazon Route 53 en la que va a crear el conjunto de registros de recursos de alias. «example.com» es el vértice de la zona para el que desea crear un alias y «www.example.com» es un subdominio para el que también desea crear un alias. «Z11111» es un ejemplo de un ID de zona alojada para el balanceador de cargas y «example-load-balancer-11.us-east-1.elb.amazonaws.com» es un ejemplo de un nombre de dominio de balanceador de carga con el que Amazon Route 53 responde a las consultas de example.com y www.example.com. Consulte la documentación de Amazon Route 53 para obtener más información. Puede usar el Get-R53Change cmdlet para sondear y determinar cuándo se han completado los cambios.

```
$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 = "Z1111111111111"
$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 = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets, one
 for the zone apex, example.com, and one for www.example.com, that both point to
 example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Ejemplo 3: en este ejemplo se crean dos registros A para www.example.com. Una cuarta parte de las veces (1/ (1+3)), Amazon Route 53 responde a las consultas de www.example.com con los dos valores del primer conjunto de registros de recursos (192.0.2.9 y 192.0.2.10). Tres cuartas partes de las veces (3/ (1+3)) Amazon Route 53 responde a las consultas de www.example.com con los dos valores del segundo conjunto de registros de recursos (192.0.2.11 y 192.0.2.12). Consulte la documentación de Amazon Route 53 para obtener más información. Puede usar el Get-R53Change cmdlet para sondear y determinar cuándo se han completado los cambios.

```
$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
```

Ejemplo 4: en este ejemplo se muestra cómo crear conjuntos de registros de recursos de alias ponderados suponiendo que example.com es el dominio para el que desea crear conjuntos de registros de recursos de alias ponderados. SetIdentifier diferencia los dos conjuntos de registros de recursos de alias ponderados entre sí. Este elemento es obligatorio porque los elementos Nombre y Tipo tienen los mismos valores para ambos conjuntos de registros de recursos. Los valores Z↻ 11111 y Z3333333333333 son ejemplos de zona hospedada para el balanceador de ELB cargas especificado IDs por el valor de. DNSName example-load-balancer-2222222222.us-east-1.elb.amazonaws.com y -4444444444.us-east-1.elb.amazonaws.com example-load-balancer son ejemplos de dominios de Elastic Load Balancing desde los que Amazon Route 53 responde a las consultas de example.com. Consulte la documentación de Amazon Route 53 para obtener más información. Puede usar el Get-R53Change cmdlet para sondear y determinar cuándo se han completado los cambios.

```
$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 = "Z1111111111111"
$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 = "Z3333333333333"
$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
```

Ejemplo 5: en este ejemplo se crean dos conjuntos de registros de recursos de alias de latencia, uno para un balanceador de ELB cargas en la región EE.UU. Oeste (Oregón) (us-west-2) y otro para un balanceador de cargas en la región Asia-Pacífico (Singapur) (ap-southeast-1). Consulte la documentación de Amazon Route 53 para obtener más información. Puede usar el Get-R53Change cmdlet para sondear y determinar cuándo se han completado los cambios.

```
$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 = "Z1111111111111"
```

```
$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 = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
 sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
 region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Para API obtener más información, consulte la referencia del [ChangeResourceRecordSets AWS Tools for PowerShell](cmdlet).

`Get-R53AccountLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53AccountLimit`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el número máximo de zonas alojadas que se pueden crear con la cuenta actual.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Salida:

```
15
```

- Para API obtener más información, consulte GetAccountLimitla referencia de AWS Tools for PowerShell cmdlets.

`Get-R53CheckerIpRanges`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53CheckerIpRanges`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve el valor de los CIDRs comprobadores de estado de Route53

```
Get-R53CheckerIpRanges
```

Salida:

```
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
```

- Para API obtener más información, consulte Cmdlet GetCheckerIpRangesReference AWS Tools for PowerShell .

`Get-R53HostedZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53HostedZone`

Herramientas para PowerShell

Ejemplo 1: Devuelve los detalles de la zona alojada con el ID PJN98FT9 Z1D633.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Para obtener API más información, consulte GetHostedZonela referencia de cmdlets.AWS Tools for PowerShell

`Get-R53HostedZoneCount`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53HostedZoneCount`

Herramientas para PowerShell

Ejemplo 1: devuelve el número total de zonas alojadas públicas y privadas de la actual Cuenta de AWS.

```
Get-R53HostedZoneCount
```

- Para API obtener más información, consulte GetHostedZoneCountla referencia de AWS Tools for PowerShell cmdlets.

`Get-R53HostedZoneLimit`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53HostedZoneLimit`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve el límite del número máximo de registros que se pueden crear en la zona alojada especificada.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Salida:

```
5
```

- Para API obtener más información, consulte GetHostedZoneLimit AWS Tools for PowerShellCmdlet Reference.

`Get-R53HostedZoneList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53HostedZoneList`

Herramientas para PowerShell

Ejemplo 1: Genera todas las zonas alojadas públicas y privadas.

```
Get-R53HostedZoneList
```

Ejemplo 2: Muestra todas las zonas alojadas que están asociadas al conjunto de delegación reutilizable que tiene el ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Para API obtener más información, consulte ListHostedZonesla referencia del AWS Tools for PowerShell cmdlet.

`Get-R53HostedZonesByName`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53HostedZonesByName`

Herramientas para PowerShell

Ejemplo 1: Devuelve todas las zonas alojadas públicas y privadas ASCII ordenadas por nombre de dominio.

```
Get-R53HostedZonesByName
```

Ejemplo 2: Devuelve las zonas alojadas públicas y privadas, ASCII ordenadas por nombre de dominio, empezando por el DNS nombre especificado.

```
Get-R53HostedZonesByName -DnsName example2.com
```

Ejemplo 3: En este ejemplo se muestra cómo enumerar manualmente las zonas alojadas. Para ello, primero se recupera un único elemento y, a continuación, se repiten de dos en dos hasta que se devuelvan todas las zonas, utilizando las propiedades de los marcadores adjuntas a la respuesta del servicio en la **`$AWSHistory`** pila después de cada llamada.

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- Para API obtener más información, consulte Cmdlet ListHostedZonesByNameReference AWS Tools for PowerShell .

`Get-R53QueryLoggingConfigList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53QueryLoggingConfigList`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo devuelve todas las configuraciones del registro de DNS consultas asociadas a la actual Cuenta de AWS.

```
Get-R53QueryLoggingConfigList
```

Salida:

```
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:*
```

- Para API obtener más información, consulte ListQueryLoggingConfigsla referencia del AWS Tools for PowerShell cmdlet.

`Get-R53ReusableDelegationSet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-R53ReusableDelegationSet`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera información sobre el conjunto de delegación especificado, incluidos los cuatro servidores de nombres que están asignados al conjunto de delegación.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Salida:

```
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}
```

- Para API obtener más información, consulte GetReusableDelegationSet AWS Tools for PowerShellCmdlet Reference.

`New-R53HostedZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-R53HostedZone`

Herramientas para PowerShell

Ejemplo 1: crea una nueva zona alojada llamada 'example.com', asociada a un conjunto de delegación reutilizable. Tenga en cuenta que debe proporcionar un valor para el CallerReference parámetro, de modo que las solicitudes se tengan que volver a intentar en caso necesario sin correr el riesgo de ejecutar la operación dos veces. Como la zona alojada se crea en unVPC, es automáticamente privada y no debe establecer el PrivateZone parámetro - HostedZoneConfig _.

```
$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
```

- Para API obtener más información, consulte CreateHostedZonela referencia de AWS Tools for PowerShell cmdlets.

`New-R53QueryLoggingConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-R53QueryLoggingConfig`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea una nueva configuración de registro de DNS consultas de Route53 para la zona alojada especificada. Amazon Route53 publicará los registros de DNS consultas en el grupo de registros de Cloudwatch especificado.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Salida:

```
QueryLoggingConfig                    Location
------------------                    --------
```

```
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- Para API obtener más información, consulte la referencia de CreateQueryLoggingConfigcmdlets AWS Tools for PowerShell .

`New-R53ReusableDelegationSet`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-R53ReusableDelegationSet`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo crea un conjunto de delegación reutilizable de 4 servidores de nombres que pueden reutilizarse en varias zonas alojadas.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Salida:

```
DelegationSet                       Location
-------------                       --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- Para API obtener más información, consulte la referencia CreateReusableDelegationSetde AWS Tools for PowerShell cmdlets.

`Register-R53VPCWithHostedZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-R53VPCWithHostedZone`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia lo especificado VPC a la zona alojada privada.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Salida:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- Para API obtener más información, consulte A ssociateVPCWith HostedZone en la referencia del AWS Tools for PowerShell cmdlet.

`Remove-R53HostedZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-R53HostedZone`

Herramientas para PowerShell

Ejemplo 1: Elimina la zona alojada con el ID especificado. Se le solicitará que lo confirme antes de continuar con el comando, a menos que añada el parámetro -Force switch.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Para API obtener más información, consulte DeleteHostedZone AWS Tools for PowerShellCmdlet Reference.

`Remove-R53QueryLoggingConfig`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-R53QueryLoggingConfig`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la configuración especificada para el registro de DNS consultas.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Para API obtener más información, consulte DeleteQueryLoggingConfig AWS Tools for PowerShellCmdlet Reference.

`Remove-R53ReusableDelegationSet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-R53ReusableDelegationSet`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el conjunto de delegación reutilizable especificado.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Para API obtener más información, consulte DeleteReusableDelegationSet AWS Tools for PowerShellCmdlet Reference.

`Unregister-R53VPCFromHostedZone`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-R53VPCFromHostedZone`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se desvincula lo especificado VPC de la zona alojada privada.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

Salida:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- Para API obtener más información, consulte D isassociateVPCFrom HostedZone en la referencia del AWS Tools for PowerShell cmdlet.

`Update-R53HostedZoneComment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-R53HostedZoneComment`

Herramientas para PowerShell

Ejemplo 1: Este comando actualiza el comentario de la zona alojada especificada.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

Salida:

```
Id                     : /hostedzone/Z385PDS6AAAAAR
Name                   : example.com.
CallerReference        : C5B55555-7147-EF04-8341-69131E805C89
Config                 : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService          :
```

- Para API obtener más información, consulte UpdateHostedZoneCommentla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Amazon S3 que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS Tools for PowerShell mediante Amazon S3.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Copy-S3Object`

El siguiente ejemplo de código muestra cómo usarlo`Copy-S3Object`.

Herramientas para PowerShell

Ejemplo 1: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Ejemplo 2: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket “backup-files”, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
 sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Ejemplo 3: este comando descarga el objeto “sample.txt” del bucket “test-files” a un archivo local con el nombre “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Ejemplo 4: descarga el objeto individual en el archivo especificado. El archivo descargado se encuentra en c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

Ejemplo 5: descarga todos los objetos que coinciden con el prefijo de clave especificado en la carpeta local. La jerarquía de claves relativa se conservará como subcarpetas en la ubicación general de descarga.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- Para API obtener más información, consulte CopyObjectla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3ACL`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3ACL`

Herramientas para PowerShell

Ejemplo 1: El comando obtiene los detalles del propietario del objeto S3.

```
Get-S3ACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 AccessControlList.Owner
```

Salida:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Get ACL in.

`Get-S3Bucket`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3Bucket`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve todos los buckets de S3.

```
Get-S3Bucket
```

Ejemplo 2: este comando devuelve un bucket denominado “test-files”

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para API obtener más información, consulte ListBucketsla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketAccelerateConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketAccelerateConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el valor Enabled si la configuración de aceleración de transferencia está habilitada para el bucket especificado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
Enabled
```

- Para API obtener más información, consulte GetBucketAccelerateConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketAnalyticsConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketAnalyticsConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve los detalles del filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para API obtener más información, consulte GetBucketAnalyticsConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketAnalyticsConfigurationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketAnalyticsConfigurationList`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de análisis del bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte ListBucketAnalyticsConfigurationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketEncryption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketEncryption`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve todas las reglas de cifrado del servidor asociadas al bucket determinado.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketEncryptionla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketInventoryConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketInventoryConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve los detalles del inventario con el nombre “testinventory” del bucket de S3 indicado.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Para API obtener más información, consulte GetBucketInventoryConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketInventoryConfigurationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketInventoryConfigurationList`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de inventario del bucket de S3 indicado.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte ListBucketInventoryConfigurationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketLocation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketLocation`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve la restricción de ubicación del bucket “s3testbucket”, si existe una restricción.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
ap-south-1
```

- Para API obtener más información, consulte GetBucketLocationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketLogging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketLogging`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el estado de registros del bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Para API obtener más información, consulte GetBucketLoggingla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketMetricsConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketMetricsConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve los detalles del filro de métricas con el nombre “testfilter” del bucket de S3 indicado.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Para API obtener más información, consulte GetBucketMetricsConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketNotification`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketNotification`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera la configuración de notificaciones del bucket en cuestión

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Salida:

```
Id   Topic
--   -----
```

```
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para API obtener más información, consulte GetBucketNotificationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketPolicy`

Herramientas para PowerShell

Ejemplo 1: este comando muestra la política de bucket asociada al bucket de S3 indicado.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketPolicyStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketPolicyStatus`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el estado de política del bucket específico de S3 e indica si el bucket es público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketPolicyStatusla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketReplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketReplication`

Herramientas para PowerShell

Ejemplo 1: devuelve la información de configuración de replicación establecida en el bucket denominado “mybucket”..

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para API obtener más información, consulte GetBucketReplicationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketRequestPayment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketRequestPayment`

Herramientas para PowerShell

Ejemplo 1: devuelve la configuración de pagos de solicitudes del bucket denominado “mybucket”.. De forma predeterminada, el propietario del bucket paga las descargas realizadas desde el bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para API obtener más información, consulte GetBucketRequestPaymentla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketTagging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketTagging`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve todas las etiquetas asociadas al bucket indicado.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketTaggingla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketVersioning`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketVersioning`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el estado del control de versiones con respecto al bucket indicado.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketVersioningla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3BucketWebsite`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3BucketWebsite`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve los detalles de las configuraciones de sitio web estáticas del bucket de S3 indicado.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetBucketWebsitela referencia de AWS Tools for PowerShell cmdlets.

`Get-S3CORSConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3CORSConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve un objeto que contiene todas las reglas de CORS configuración correspondientes al bucket de S3 en cuestión.

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 Configuration.Rules
```

Salida:

```
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 : {}
```

- Para API obtener más información, consulte G etCORSConfiguration en la referencia del AWS Tools for PowerShell cmdlet.

`Get-S3LifecycleConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3LifecycleConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo recupera la configuración del ciclo de vida del depósito.

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

Salida:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Para API obtener más información, consulte la referencia GetLifecycleConfigurationdel AWS Tools for PowerShell cmdlet.

`Get-S3Object`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3Object`

Herramientas para PowerShell

Ejemplo 1: este comando recupera la información sobre todos los elementos del bucket “test-files”.

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

Ejemplo 2: este comando recupera la información sobre el elemento “sample.txt” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 3: este comando recupera la información sobre todos los elementos con el prefijo “sample” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para API obtener más información, consulte ListObjectsla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3ObjectLockConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3ObjectLockConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve el valor “Enabled” si la configuración de bloqueo de objetos está habilitada para el bucket de S3 indicado.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Salida:

```
Value
```

```
-----
Enabled
```

- Para API obtener más información, consulte GetObjectLockConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3ObjectMetadata`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3ObjectMetadata`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los metadatos del objeto con la clave 'ListTrusts.txt' en el bucket de S3 dado.

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

Salida:

```
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                              :
```

- Para API obtener más información, consulte la referencia del GetObjectMetadata AWS Tools for PowerShellcmdlet.

`Get-S3ObjectRetention`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3ObjectRetention`

Herramientas para PowerShell

Ejemplo 1: el comando devuelve el modo y la fecha hasta que se retenga el objeto.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para API obtener más información, consulte GetObjectRetentionla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3ObjectTagSet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3ObjectTagSet`

Herramientas para PowerShell

Ejemplo 1: el ejemplo devuelve las etiquetas asociadas al objeto presentes en el bucket de S3 indicado.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Key  Value
---  -----
test value
```

- Para API obtener más información, consulte GetObjectTaggingla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3PreSignedURL`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3PreSignedURL`

Herramientas para PowerShell

Ejemplo 1: El comando devuelve una clave especificada previamente firmada URL y una fecha de caducidad.

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires
 '2023-11-16'
```

Ejemplo 2: El comando devuelve un depósito de directorio prefirmado URL con la clave especificada y una fecha de caducidad.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- Para API obtener más información, consulte GetPreSignedURL AWS Tools for PowerShellCmdlet Reference.

`Get-S3PublicAccessBlock`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3PublicAccessBlock`

Herramientas para PowerShell

Ejemplo 1: el comando devuelve la configuración de bloqueo de acceso público del bucket de S3 indicado.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte GetPublicAccessBlockla referencia de AWS Tools for PowerShell cmdlets.

`Get-S3Version`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-S3Version`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los metadatos de todas las versiones de los objetos del bucket de S3 en cuestión.

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : s3testbucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- Para API obtener más información, consulte ListVersions AWS Tools for PowerShellCmdlet Reference.

`New-S3Bucket`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-S3Bucket`

Herramientas para PowerShell

Ejemplo 1: Este comando crea un nuevo depósito privado denominado «sample-bucket».

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

Ejemplo 2: Este comando crea un nuevo depósito denominado «sample-bucket» con permisos de lectura y escritura.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

Ejemplo 3: Este comando crea un nuevo depósito denominado «sample-bucket» con permisos de solo lectura.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

Ejemplo 4: Este comando crea un nuevo bucket de directorio denominado «samplebucket--use1-az5--x-s3" con. 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
```

- Para obtener PutBucket AWS Tools for PowerShellmás información, consulte Cmdlet Reference. API

`Read-S3Object`

En el siguiente ejemplo de código se muestra cómo usarlo. `Read-S3Object`

Herramientas para PowerShell

Ejemplo 1: este comando recupera el elemento “sample.txt” del bucket “test-files” y lo guarda en un archivo denominado “local-sample.txt” en la ubicación actual. No es necesario que el archivo “local-sample.txt” exista para poder llamar a este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Ejemplo 2: Este comando recupera el directorio virtual "DIR" del depósito «test-files» y lo guarda en una carpeta denominada «Local-DIR» en la ubicación actual. No es necesario que la carpeta «Local-DIR» exista para poder ejecutar este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Ejemplo 3: descarga todos los objetos cuyas claves terminan en “.json” de los buckets con “config” en el nombre del bucket a los archivos de la carpeta especificada. Las claves de objeto se utilizan para establecer los nombres de los archivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Para API obtener más información, consulte GetObjectla Referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3Bucket`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3Bucket`

Herramientas para PowerShell

Ejemplo 1: este comando elimina todos los objetos y las versiones de los objetos del bucket “test-files” y, a continuación, elimina el bucket. El comando solicitará una confirmación antes de continuar. Añada el conmutador -Force para suprimir la confirmación. Tenga en cuenta que los buckets que no estén vacíos no se pueden eliminar.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para API obtener más información, consulte DeleteBucketla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketAnalyticsConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketAnalyticsConfiguration`

Herramientas para PowerShell

Ejemplo 1: el comando elimina el filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para API obtener más información, consulte DeleteBucketAnalyticsConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketEncryption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketEncryption`

Herramientas para PowerShell

Ejemplo 1: esto deshabilita el cifrado habilitado para el bucket de S3 proporcionado.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteBucketEncryptionla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketInventoryConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketInventoryConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este comando elimina el inventario denominado 'testInventoryName' correspondiente al depósito de S3 dado.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testInventoryName'
```

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para API obtener más información, consulte la referencia del DeleteBucketInventoryConfiguration AWS Tools for PowerShellcmdlet.

`Remove-S3BucketMetricsConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketMetricsConfiguration`

Herramientas para PowerShell

Ejemplo 1: el comando elimina el filtro de métricas con el nombre “testmetrics” en el bucket de S3 indicado.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Para API obtener más información, consulte DeleteBucketMetricsConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketPolicy`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketPolicy`

Herramientas para PowerShell

Ejemplo 1: el comando elimina la política de bucket asociada al bucket de S3 indicado.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte DeleteBucketPolicyla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketReplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketReplication`

Herramientas para PowerShell

Ejemplo 1: elimina la configuración de replicación asociada al bucket denominado “mybucket”. Tenga en cuenta que esta operación requiere permiso para la DeleteReplicationConfiguration acción s3:. Se le solicitará la confirmación antes de continuar con la operación; para suprimir la confirmación, utilice el conmutador -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para API obtener más información, consulte DeleteBucketReplicationla referencia del AWS Tools for PowerShell cmdlet.

`Remove-S3BucketTagging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketTagging`

Herramientas para PowerShell

Ejemplo 1: este comando elimina todas las etiquetas asociadas al bucket de S3 indicado.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para API obtener más información, consulte DeleteBucketTaggingla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3BucketWebsite`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3BucketWebsite`

Herramientas para PowerShell

Ejemplo 1: este comando deshabilita la propiedad de alojamiento de sitios web estáticos del bucket de S3 indicado.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para API obtener más información, consulte DeleteBucketWebsitela referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3CORSConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3CORSConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este comando elimina la CORS configuración del bucket de S3 dado.

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
 target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para API obtener más información, consulte D eleteCORSConfiguration en la referencia del AWS Tools for PowerShell cmdlet.

`Remove-S3LifecycleConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3LifecycleConfiguration`

Herramientas para PowerShell

Ejemplo 1: El comando elimina todas las reglas del ciclo de vida del bucket de S3 en cuestión.

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- Para API obtener más información, consulte DeleteLifecycleConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3MultipartUpload`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3MultipartUpload`

Herramientas para PowerShell

Ejemplo 1: este comando anula las cargas multiparte creadas hace menos de 5 días.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Ejemplo 2: este comando anula las cargas multiparte creadas antes del 2 de enero de 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
 January 02, 2014"
```

Ejemplo 3: este comando anula las cargas multiparte creadas antes del 2 de enero de 2014 a las 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02
 10:45:37"
```

- Para API obtener más información, consulte AbortMultipartUploadla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3Object`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3Object`

Herramientas para PowerShell

Ejemplo 1: este comando elimina el objeto “sample.txt” del bucket “test-files”. Antes de ejecutar el comando, se le solicitará que lo confirme; para suprimir el mensaje, utilice el conmutador -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 2: este comando elimina la versión especificada del objeto “sample.txt” del bucket “test-files”, suponiendo que el bucket se haya configurado para habilitar las versiones de los objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Ejemplo 3: este comando elimina los objetos “sample1.txt”, “sample2.txt” y “sample3.txt” del bucket “test-files” como una sola operación por lotes. La respuesta del servicio mostrará una lista de todas las claves procesadas, independientemente del estado de éxito o error de la eliminación. Para obtener únicamente los errores de las claves que el servicio no pudo procesar, añada el ReportErrorsOnly parámetro - (este parámetro también se puede especificar con el alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Ejemplo 4: En este ejemplo se utiliza una expresión en línea con el KeyCollection parámetro - para obtener las claves de los objetos que se van a eliminar. Get-S3Objectdevuelve una colección de instancias de Amazon.S3.Model.S3Object, cada una de las cuales tiene un elemento clave del tipo string que identifica el objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Ejemplo 5: este ejemplo obtiene todos los objetos que tienen un prefijo de clave “prefijo/subprefijo” en el bucket y los elimina. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, considere la posibilidad de pasar la colección al parámetro - InputObject (alias -S3ObjectCollection) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Ejemplo 6: en este ejemplo, se canaliza al cmdlet una colección de ObjectVersion instancias de Amazon.S3.Model.S3 que representan marcadores de eliminación para su eliminación. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, considere la posibilidad de pasarla al parámetro - InputObject (alias -S3ObjectCollection) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Ejemplo 7: Este script muestra cómo realizar una eliminación por lotes de un conjunto de objetos (en este caso, eliminar marcadores) mediante la construcción de una matriz de objetos para utilizarlos con el parámetro -. KeyAndVersionCollection

```
$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
```

- Para API obtener más información, consulte DeleteObjects AWS Tools for PowerShellCmdlet Reference.

`Remove-S3ObjectTagSet`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3ObjectTagSet`

Herramientas para PowerShell

Ejemplo 1: este comando elimina todas las etiquetas asociadas con el objeto con la clave “testfile.txt” en el bucket de S3 indicado.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
 '^Key'
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteObjectTaggingla referencia de AWS Tools for PowerShell cmdlets.

`Remove-S3PublicAccessBlock`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-S3PublicAccessBlock`

Herramientas para PowerShell

Ejemplo 1: este comando desactiva la configuración de bloqueo de acceso público para el bucket en cuestión.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Salida:

```
s3testbucket
```

- Para API obtener más información, consulte DeletePublicAccessBlockla referencia de AWS Tools for PowerShell cmdlets.

`Set-S3BucketEncryption`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-S3BucketEncryption`

Herramientas para PowerShell

Ejemplo 1: Este comando habilita el cifrado predeterminado AES256 del lado del servidor con claves administradas de Amazon SSE S3 (-S3) en el bucket dado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para API obtener más información, consulte la referencia PutBucketEncryptionde AWS Tools for PowerShell cmdlets.

`Test-S3Bucket`

En el siguiente ejemplo de código se muestra cómo usarlo. `Test-S3Bucket`

Herramientas para PowerShell

Ejemplo 1: este comando devuelve True si el bucket existe y False en caso contrario. El comando devuelve True incluso si el depósito no pertenece al usuario.

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para API obtener más información, consulte Test-S3Bucket AWS Tools for PowerShellCmdlet Reference.

`Write-S3BucketAccelerateConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketAccelerateConfiguration`

Herramientas para PowerShell

Ejemplo 1: este comando habilita la aceleración de transferencia para el bucket de S3 indicado.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Para API obtener más información, consulte PutBucketAccelerateConfigurationla referencia de AWS Tools for PowerShell cmdlets.

`Write-S3BucketNotification`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketNotification`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se configura la configuración del SNS tema para el evento de S3 ObjectRemovedDelete y se habilita la notificación para el bucket de s3 en cuestión

```
$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
```

Ejemplo 2: En este ejemplo, se habilitan las notificaciones del ObjectCreatedAll bucket en cuestión y se envían a la función Lambda.

```
$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
```

Ejemplo 3: este ejemplo crea 2 configuraciones de Lambda diferentes sobre la base de un sufijo clave diferente y las configura en un solo comando.

```
#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
```

- Para API obtener más información, consulte PutBucketNotification AWS Tools for PowerShellCmdlet Reference.

`Write-S3BucketReplication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketReplication`

Ejemplo 1: Este ejemplo establece una configuración de replicación con una sola regla que permite replicar en el depósito «exampletargetbucket» cualquier objeto nuevo creado con el prefijo de nombre clave «» en el depósito «examplebucket»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
```

Ejemplo 2: En este ejemplo se establece una configuración de replicación con varias reglas que permiten replicar en el bucket «exampletargetbucket» cualquier objeto nuevo que se cree con el prefijo del nombre de clave «» o «». TaxDocs OtherDocs Los prefijos de claves no deben superponerse.

```
$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
```

Ejemplo 3: En este ejemplo, se actualiza la configuración de replicación en el depósito especificado para inhabilitar la regla que controla la replicación de objetos con el prefijo de nombre clave "" en el depósito «exampletargetbucket'. TaxDocs

```
$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
```

- PutBucketReplicationPara AWS Tools for PowerShell obtener más información, consulte Cmdlet Reference. API

`Write-S3BucketRequestPayment`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketRequestPayment`

Herramientas para PowerShell

Ejemplo 1: actualiza la configuración de pago de solicitud del bucket denominado “mybucket”, de modo que se cobre la descarga a la persona que solicita las descargas del bucket. De forma predeterminada, el propietario del bucket paga las descargas. Para volver a establecer el pago de la solicitud como predeterminado, utilice 'BucketOwner' para el parámetro RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet PutBucketRequestPaymentReference.

`Write-S3BucketTagging`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketTagging`

Herramientas para PowerShell

Ejemplo 1: este comando aplica dos etiquetas a un bucket denominado **`cloudtrail-test-2018`**, una etiqueta con una clave de Stage y un valor de Test, y una etiqueta con una clave de Environment y un valor de Alpha. Para comprobar que las etiquetas se han añadido al bucket, ejecute **`Get-S3BucketTagging -BucketName bucket_name`**. Los resultados deben mostrar las etiquetas que ha aplicado al bucket en el primer comando. Tenga en cuenta que **`Write-S3BucketTagging`** sobrescribe todo el conjunto de etiquetas existente en un bucket. Para agregar o eliminar etiquetas individuales, ejecute los API cmdlets Resource Groups y Tagging y. **`Add-RGTResourceTag`** **`Remove-RGTResourceTag`** Como alternativa, puede utilizar el editor de etiquetas de la consola de AWS administración para gestionar las etiquetas de bucket de S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Ejemplo 2: este comando canaliza un bucket denominado **`cloudtrail-test-2018`** al cmdlet de **`Write-S3BucketTagging`**. Aplica las etiquetas Stage:Production y Department:Finance al bucket. Tenga en cuenta que **`Write-S3BucketTagging`** sobrescribe todo el conjunto de etiquetas existente en un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Para API obtener más información, consulte PutBucketTaggingla referencia de AWS Tools for PowerShell cmdlets.

`Write-S3BucketVersioning`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketVersioning`

Herramientas para PowerShell

Ejemplo 1: el comando habilita el control de versiones para el bucket de S3 indicado.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status
 Enabled
```

- Para API obtener más información, consulte PutBucketVersioningla referencia de AWS Tools for PowerShell cmdlets.

`Write-S3BucketWebsite`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3BucketWebsite`

Herramientas para PowerShell

Ejemplo 1: el comando habilita el alojamiento de sitios web para el bucket indicado con el documento de índice como “index.html” y el documento de error como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para API obtener más información, consulte PutBucketWebsitela referencia de AWS Tools for PowerShell cmdlets.

`Write-S3LifecycleConfiguration`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3LifecycleConfiguration`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo escribe o reemplaza la configuración proporcionada en el $NewRule. Esta configuración garantiza limitar los objetos del ámbito con valores de prefijo y etiqueta determinados.

```
$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
```

Ejemplo 2: Este ejemplo establece varias reglas con filtrado. $ ArchiveRule establece que los objetos se archivarán en 30 días en Glacier y en 120 días DeepArchive. $ ExpireRule caduca tanto en la versión actual como en la anterior en 150 días para los objetos con el prefijo «py» y el prefijo tag:key «archivado» establecido en «sí».

```
$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
```

- Para obtener API más información, consulte la referencia de cmdlets. PutLifecycleConfigurationAWS Tools for PowerShell

`Write-S3Object`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3Object`

Herramientas para PowerShell

Ejemplo 1: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Ejemplo 2: este comando carga el archivo único “sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”. Si no se proporciona el parámetro -Key, el nombre del archivo se utiliza como clave de objeto de S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Ejemplo 3: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “prefix/to/sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

Ejemplo 4: Este comando carga todos los archivos del subdirectorio «Scripts» al bucket «test-files» y aplica el key prefijo común «» a cada objeto. SampleScripts Cada archivo cargado tendrá una clave de «SampleScripts/filename», donde la palabra «filename» variará.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Ejemplo 5: Este comando carga todos los archivos*.ps1 del directorio local «Scripts» al depósito «archivos de prueba» y aplica el prefijo clave común «» a cada objeto. SampleScripts Cada archivo cargado tendrá la clave "SampleScripts/nombrearchivo.ps1", donde el nombre de archivo variará.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Ejemplo 6: este comando crea un nuevo objeto S3 que contiene la cadena de contenido especificada con la clave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Ejemplo 7: este comando carga el archivo especificado (el nombre del archivo se usa como clave) y aplica las etiquetas especificadas al nuevo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Ejemplo 8: este comando carga de forma recursiva la carpeta especificada y aplica las etiquetas especificadas a todos los objetos nuevos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obtener API más información, consulte la referencia de cmdlets. PutObjectAWS Tools for PowerShell

`Write-S3ObjectRetention`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-S3ObjectRetention`

Herramientas para PowerShell

Ejemplo 1: el comando habilita el modo de retención de gobierno hasta la fecha “31 de diciembre de 2019 a las 00:00:00” para el objeto “testfile.txt” del bucket de S3 indicado.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para API obtener más información, consulte PutObjectRetentionla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de S3 Glacier que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell con S3 Glacier.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Get-GLCJob`

El siguiente ejemplo de código muestra cómo usarlo`Get-GLCJob`.

Herramientas para PowerShell

Ejemplo 1: Devuelve los detalles del trabajo especificado. Cuando el trabajo se completa correctamente, el GCJobOutput cmdlet Read- se puede utilizar para recuperar el contenido del trabajo (un archivo o una lista de inventario) en el sistema de archivos local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Salida:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...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
```

- Para API obtener más información, consulte la referencia del DescribeJob AWS Tools for PowerShellcmdlet.

`New-GLCVault`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-GLCVault`

Herramientas para PowerShell

Ejemplo 1: crea una nueva bóveda para la cuenta del usuario. Como no se ha proporcionado ningún valor al AccountId parámetro -, los cmdlets utilizan el valor predeterminado «-» para indicar la cuenta actual.

```
New-GLCVault -VaultName myvault
```

Salida:

```
/01234567812/vaults/myvault
```

- Para API obtener más información, consulte la referencia del CreateVault AWS Tools for PowerShellcmdlet.

`Read-GLCJobOutput`

En el siguiente ejemplo de código se muestra cómo usarlo. `Read-GLCJobOutput`

Herramientas para PowerShell

Ejemplo 1: descarga el contenido del archivo cuya recuperación estaba programada en el trabajo especificado y lo almacena en un archivo en el disco. La descarga valida la suma de comprobación, si hay alguna disponible. Si es necesario, la suma de comprobación se puede obtener del historial de respuestas del servicio de la siguiente manera (suponiendo que este cmdlet se ejecutó por última vez): **`$AWSHistory.LastServiceResponse`** Si el cmdlet no fue el que se ejecutó más recientemente, inspeccione la **`$AWSHistory.Commands`** colección para obtener la respuesta del servicio correspondiente.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- Para API obtener más información, consulte la referencia del GetJobOutput AWS Tools for PowerShellcmdlet.

`Start-GLCJob`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-GLCJob`

Herramientas para PowerShell

Ejemplo 1: inicia un trabajo para recuperar un archivo del almacén especificado propiedad del usuario. El estado del trabajo se puede comprobar mediante el GLCJob cmdlet Get-. Cuando el trabajo se complete correctamente, el GCJobOutput cmdlet Read- se puede utilizar para recuperar el contenido del archivo en el sistema de archivos local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Salida:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- Para API obtener más información, consulte la referencia del InitiateJob AWS Tools for PowerShellcmdlet.

`Write-GLCArchive`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-GLCArchive`

Herramientas para PowerShell

Ejemplo 1: carga un único archivo en el almacén especificado y devuelve el identificador del archivo y la suma de comprobación calculada.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Salida:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Ejemplo 2: carga el contenido de una jerarquía de carpetas al almacén especificado de la cuenta del usuario. Para cada archivo cargado, el cmdlet emite el nombre del archivo, el identificador de archivo correspondiente y la suma de comprobación calculada del archivo.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Salida:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin         qXAfOdSG...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-XfOPA 7469e...3e86f1
```

- Para obtener API más información, consulte UploadArchivela referencia del cmdlet.AWS Tools for PowerShell

SESEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonSES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Get-SESIdentity`

El siguiente ejemplo de código muestra cómo usarlo`Get-SESIdentity`.

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve una lista que contiene todas las identidades (direcciones de correo electrónico y dominios) de una AWS cuenta específica, independientemente del estado de verificación.

```
Get-SESIdentity
```

- Para API obtener más información, consulte ListIdentities AWS Tools for PowerShellCmdlet Reference.

`Get-SESSendQuota`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SESSendQuota`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve los límites de envío actuales del usuario.

```
Get-SESSendQuota
```

- Para API obtener más información, consulte GetSendQuota AWS Tools for PowerShellCmdlet Reference.

`Get-SESSendStatistic`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SESSendStatistic`

Herramientas para PowerShell

Ejemplo 1: Este comando devuelve las estadísticas de envío del usuario. El resultado es una lista de puntos de datos que representan las dos últimas semanas de actividad de envío. Cada punto de datos de la lista contiene estadísticas para un intervalo de 15 minutos.

```
Get-SESSendStatistic
```

- Para API obtener más información, consulte GetSendStatisticsla referencia de AWS Tools for PowerShell cmdlets.

SNSEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonSNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Publish-SNSMessage`

El siguiente ejemplo de código muestra cómo usarlo`Publish-SNSMessage`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra la publicación de un mensaje con una sola línea MessageAttribute declarada.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

Ejemplo 2: En este ejemplo se muestra la publicación de un mensaje con varios MessageAttributes declarados de antemano.

```
$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
```

- Para API obtener más información, consulte Publish in AWS Tools for PowerShell Cmdlet Reference.

SQSEjemplos de Amazon que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante AmazonSQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`Add-SQSPermission`

El siguiente ejemplo de código muestra cómo usarlo`Add-SQSPermission`.

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite Cuenta de AWS al especificado enviar mensajes desde la cola especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para API obtener más información, consulte AddPermission AWS Tools for PowerShellCmdlet Reference.

`Clear-SQSQueue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Clear-SQSQueue`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se eliminan todos los mensajes de la cola especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet PurgeQueueReference.

`Edit-SQSMessageVisibility`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-SQSMessageVisibility`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo cambia el tiempo de espera de visibilidad del mensaje con el identificador de recibo especificado en la cola especificada a 10 horas (10 horas * 60 minutos * 60 segundos = 36 000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Para API obtener más información, consulte Cmdlet ChangeMessageVisibilityReference AWS Tools for PowerShell .

`Edit-SQSMessageVisibilityBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-SQSMessageVisibilityBatch`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo cambia el tiempo de espera de visibilidad de 2 mensajes con los identificadores de recepción especificados en la cola especificada. El tiempo de espera de visibilidad del primer mensaje se cambia a 10 horas (10 horas * 60 minutos * 60 segundos = 36000 segundos). El tiempo de espera de visibilidad del segundo mensaje se cambia a 5 horas (5 horas * 60 minutos * 60 segundos = 18000 segundos).

```
$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
```

Salida:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ChangeMessageVisibilityBatchReference.

`Get-SQSDeadLetterSourceQueue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SQSDeadLetterSourceQueue`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todas URLs las colas que se basan en la cola especificada como cola de cartas muertas.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Salida:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para API obtener más información, consulte Cmdlet ListDeadLetterSourceQueuesReference AWS Tools for PowerShell .

`Get-SQSQueue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SQSQueue`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todas las colas.

```
Get-SQSQueue
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se enumeran todas las colas que comiencen con el nombre especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Salida:

```
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
```

- Para API obtener más información, consulte ListQueues AWS Tools for PowerShellCmdlet Reference.

`Get-SQSQueueAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SQSQueueAttribute`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los atributos de la cola especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Salida:

```
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":"2008-10-17","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":

 "SendMessagesFromMyQueue","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]...}
```

Ejemplo 2: En este ejemplo se enumeran por separado solo los atributos especificados para la cola especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Salida:

```
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":"2008-10-17","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":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                       arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Para API obtener más información, consulte GetQueueAttributes AWS Tools for PowerShellCmdlet Reference.

`Get-SQSQueueUrl`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SQSQueueUrl`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra una lista URL de la cola con el nombre especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Salida:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte GetQueueUrl AWS Tools for PowerShellCmdlet Reference.

`New-SQSQueue`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SQSQueue`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea una cola con el nombre especificado.

```
New-SQSQueue -QueueName MyQueue
```

Salida:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte CreateQueue AWS Tools for PowerShellCmdlet Reference.

`Receive-SQSMessage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Receive-SQSMessage`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra la información de los 10 mensajes siguientes que se van a recibir en la cola especificada. La información contendrá valores para los atributos de mensaje especificados, si existen.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Salida:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
```

```
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Para API obtener más información, consulte ReceiveMessage AWS Tools for PowerShellCmdlet Reference.

`Remove-SQSMessage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SQSMessage`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina el mensaje con el identificador de recibo especificado de la cola especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteMessageReference.

`Remove-SQSMessageBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SQSMessageBatch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se eliminan 2 mensajes con los identificadores de recepción especificados de la cola especificada.

```
$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 = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Salida:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteMessageBatchReference.

`Remove-SQSPermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SQSPermission`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo elimina la configuración de permisos con la etiqueta especificada de la cola especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte RemovePermission AWS Tools for PowerShellCmdlet Reference.

`Remove-SQSQueue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SQSQueue`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la cola especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DeleteQueueReference.

`Send-SQSMessage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Send-SQSMessage`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo envía un mensaje con los atributos y el cuerpo del mensaje especificados a la cola especificada y la entrega del mensaje se retrasa 10 segundos.

```
$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
```

Salida:

```
MD5OfMessageAttributes             MD5OfMessageBody                    MessageId

----------------------             ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE   51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Para API obtener más información, consulte SendMessage AWS Tools for PowerShellCmdlet Reference.

`Send-SQSMessageBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Send-SQSMessageBatch`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo envía 2 mensajes con los atributos y cuerpos de mensaje especificados a la cola especificada. La entrega se retrasa 15 segundos para el primer mensaje y 10 segundos para el segundo mensaje.

```
$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
```

Salida:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Para API obtener más información, consulte SendMessageBatchla Referencia de AWS Tools for PowerShell cmdlets.

`Set-SQSQueueAttribute`

En el siguiente ejemplo de código se muestra cómo usarlo. `Set-SQSQueueAttribute`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestra cómo establecer una política para suscribir una cola a un SNS tema. Cuando se publica un mensaje en el tema, se envía un mensaje a la cola de suscriptores.

```
# 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
$policy = @"
{
  "Version": "2008-10-17",
  "Id": "$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 }
```

Ejemplo 2: en este ejemplo se establecen los atributos especificados para la cola especificada.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para API obtener más información, consulte SetQueueAttributes AWS Tools for PowerShellCmdlet Reference.

AWS STS ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS STS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Convert-STSAuthorizationMessage`

El siguiente ejemplo de código muestra cómo usarlo`Convert-STSAuthorizationMessage`.

Herramientas para PowerShell

Ejemplo 1: decodifica la información adicional incluida en el contenido del mensaje codificado suministrado que se devolvió en respuesta a una solicitud. La información adicional está codificada porque los detalles del estado de la autorización pueden constituir información privilegiada que el usuario que solicitó la acción no debería ver.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Para API obtener más información, consulte DecodeAuthorizationMessagela referencia de AWS Tools for PowerShell cmdlets.

`Get-STSFederationToken`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-STSFederationToken`

Herramientas para PowerShell

Ejemplo 1: solicita un token federado válido durante una hora con “Bob” como nombre del usuario federado. Este nombre se puede usar para hacer referencia al nombre de usuario federado en una política basada en recursos (como una política de bucket de Amazon S3). La IAM política proporcionada, en JSON formato, se utiliza para determinar los permisos disponibles para el IAM usuario. La política proporcionada no puede conceder más permisos que los concedidos al usuario solicitante. Los permisos finales para el usuario federado son los más restrictivos, en función de la intersección de la política aprobada y la política de IAM usuario.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Para API obtener más información, consulte GetFederationToken AWS Tools for PowerShellCmdlet Reference.

`Get-STSSessionToken`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-STSSessionToken`

Herramientas para PowerShell

Ejemplo 1: devuelve una instancia **`Amazon.RuntimeAWSCredentials`** que contiene credenciales temporales válidas durante un período de tiempo determinado. Las credenciales utilizadas para solicitar credenciales temporales se deducen de los valores predeterminados actuales del intérprete de comandos. Para especificar otras credenciales, utilice los SecretKey parámetros - ProfileName o - AccessKey /-.

```
Get-STSSessionToken
```

Salida:

```
AccessKeyId                          Expiration
 SecretAccessKey                     SessionToken
-----------                          ----------
 ---------------                     ------------
EXAMPLEACCESSKEYID                   2/16/2015 9:12:28 PM
 examplesecretaccesskey...           SamPleTokeN.....
```

Ejemplo 2: devuelve una instancia de **`Amazon.RuntimeAWSCredentials`** que contiene credenciales temporales válidas durante una hora. Las credenciales utilizadas para realizar la solicitud se obtienen del perfil especificado.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Salida:

```
AccessKeyId                          Expiration
 SecretAccessKey                     SessionToken
-----------                          ----------
 ---------------                     ------------
EXAMPLEACCESSKEYID                   2/16/2015 9:12:28 PM
 examplesecretaccesskey...           SamPleTokeN.....
```

Ejemplo 3: Devuelve una **`Amazon.RuntimeAWSCredentials`** instancia que contiene credenciales temporales válidas durante una hora utilizando el número de identificación del MFA

dispositivo asociado a la cuenta cuyas credenciales se especifican en el perfil «myprofilename» y el valor proporcionado por el dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Salida:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- Para API obtener más información, consulte Cmdlet GetSessionTokenReference AWS Tools for PowerShell .

`Use-STSRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Use-STSRole`

Herramientas para PowerShell

Ejemplo 1: devuelve un conjunto de credenciales temporales (clave de acceso, clave secreta y token de sesión) que se pueden usar durante una hora para acceder a AWS los recursos a los que el usuario solicitante normalmente no tendría acceso. Las credenciales devueltas tienen los permisos permitidos por la política de acceso del rol que se está asumiendo y por la política proporcionada (no se puede usar la política proporcionada para conceder permisos superiores a los definidos en la política de acceso del rol que se está asumiendo).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Ejemplo 2: devuelve un conjunto de credenciales temporales, válidas durante una hora, que tienen los mismos permisos que se definen en la política de acceso del rol que se está asumiendo.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Ejemplo 3: devuelve un conjunto de credenciales temporales con el número de serie y el token generado a partir de una de las credenciales de usuario MFA asociadas a las que se utilizó para ejecutar el cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Ejemplo 4: devuelve un conjunto de credenciales temporales que han asumido un rol definido en la cuenta de un cliente. Para cada función que el tercero pueda asumir, la cuenta del cliente debe crear una función con un identificador que se debe introducir en el ExternalId parámetro - cada vez que se asuma la función.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Para API obtener más información, consulte AssumeRole AWS Tools for PowerShellCmdlet Reference.

`Use-STSWebIdentityRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Use-STSWebIdentityRole`

Herramientas para PowerShell

Ejemplo 1: devuelve un conjunto temporal de credenciales, válido durante una hora, para un usuario que se ha autenticado con el proveedor de identidad Inicio de sesión con Amazon. Las credenciales asumen la política de acceso asociada al rol identificado por el rolARN. Si lo desea, puede pasar una JSON política al parámetro -Policy que restrinja aún más los permisos de acceso (no puede conceder más permisos de los disponibles en los permisos asociados al rol). El valor proporcionado a - WebIdentityToken es el identificador de usuario único que devolvió el proveedor de identidades.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Para API obtener más información, consulte AssumeRoleWithWebIdentityla referencia del AWS Tools for PowerShell cmdlet.

AWS Support ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS Support.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-ASACommunicationToCase`

El siguiente ejemplo de código muestra cómo usarlo`Add-ASACommunicationToCase`.

Herramientas para PowerShell

Ejemplo 1: agrega el cuerpo de una comunicación de correo electrónico al caso especificado.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Ejemplo 2: agrega el cuerpo de una comunicación de correo electrónico a las mayúsculas y minúsculas especificadas más una o más direcciones de correo electrónico contenidas en la línea CC del correo electrónico.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Para API obtener más información, consulte AddCommunicationToCase AWS Tools for PowerShellCmdlet Reference.

`Get-ASACase`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASACase`

Herramientas para PowerShell

Ejemplo 1: Devuelve los detalles de todos los casos de soporte.

```
Get-ASACase
```

Ejemplo 2: Devuelve los detalles de todos los casos de soporte desde la fecha y hora especificadas.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Ejemplo 3: Devuelve los detalles de los primeros 10 casos de soporte, incluidos los que se han resuelto.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Ejemplo 4: devuelve los detalles del único caso de soporte especificado.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Ejemplo 5: Devuelve los detalles de los casos de soporte especificados.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

Ejemplo 6: devuelve todos los casos de soporte mediante la paginación manual. Los estuches se recuperan en lotes de 20.

```
$nextToken = $null
do {
  Get-ASACase -NextToken $nextToken -MaxResult 20
```

```
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte DescribeCases AWS Tools for PowerShellCmdlet Reference.

`Get-ASACommunication`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASACommunication`

Herramientas para PowerShell

Ejemplo 1: devuelve todas las comunicaciones del caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Ejemplo 2: devuelve todas las comunicaciones desde la medianoche UTC del 1 de enero de 2012 para el caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

Ejemplo 3: devuelve todas las comunicaciones desde la medianoche UTC del 1 de enero de 2012 para el caso especificado, mediante la búsqueda manual. Las comunicaciones se recuperan en lotes de 20.

```
$nextToken = $null
do {
  Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken
 $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para API obtener más información, consulte la referencia DescribeCommunicationsdel AWS Tools for PowerShell cmdlet.

`Get-ASAService`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASAService`

Herramientas para PowerShell

Ejemplo 1: Devuelve todos los códigos, nombres y categorías de servicio disponibles.

```
Get-ASAService
```

Ejemplo 2: devuelve el nombre y las categorías del servicio con el código especificado.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Ejemplo 3: Devuelve el nombre y las categorías de los códigos de servicio especificados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Ejemplo 4: Devuelve el nombre y las categorías (en japonés) de los códigos de servicio especificados. Actualmente, se admiten los códigos de idioma inglés («en») y japonés («ja»).

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Para API obtener más información, consulte DescribeServicesla Referencia de AWS Tools for PowerShell cmdlets.

`Get-ASASeverityLevel`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASASeverityLevel`

Herramientas para PowerShell

Ejemplo 1: Devuelve la lista de niveles de gravedad que se pueden asignar a un caso de AWS Support.

```
Get-ASASeverityLevel
```

Ejemplo 2: Devuelve la lista de niveles de gravedad que se pueden asignar a un caso de AWS Support. Los nombres de los niveles se devuelven en japonés.

```
Get-ASASeverityLevel -Language "ja"
```

- Para API obtener más información, consulte DescribeSeverityLevels AWS Tools for PowerShellCmdlet Reference.

`Get-ASATrustedAdvisorCheck`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASATrustedAdvisorCheck`

Herramientas para PowerShell

Ejemplo 1: Devuelve la colección de cheques de Trusted Advisor. Debe especificar el parámetro de idioma, que puede aceptar «en» para la salida en inglés o «ja» para la salida en japonés.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Para API obtener más información, consulte DescribeTrustedAdvisorChecksla referencia del AWS Tools for PowerShell cmdlet.

`Get-ASATrustedAdvisorCheckRefreshStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASATrustedAdvisorCheckRefreshStatus`

Herramientas para PowerShell

Ejemplo 1: Devuelve el estado actual de las solicitudes de actualización para las comprobaciones especificadas. Solicitud: se ASATrustedAdvisorCheckRefresh puede utilizar para solicitar que se actualice la información de estado de las comprobaciones.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Para API obtener más información, consulte la referencia DescribeTrustedAdvisorCheckRefreshStatusesdel AWS Tools for PowerShell cmdlet.

`Get-ASATrustedAdvisorCheckResult`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASATrustedAdvisorCheckResult`

Herramientas para PowerShell

Ejemplo 1: Devuelve los resultados de una comprobación de Trusted Advisor. La lista de comprobaciones de Trusted Advisor disponibles se puede obtener mediante Get-ASATrustedAdvisorChecks. El resultado es el estado general de la comprobación, la fecha y hora en la que se ejecutó por última vez y el identificador único de la comprobación específica. Para que los resultados se muestren en japonés, añada el parámetro «ja» de -Language.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Para API obtener más información, consulte la referencia DescribeTrustedAdvisorCheckResultde AWS Tools for PowerShell cmdlets.

`Get-ASATrustedAdvisorCheckSummary`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-ASATrustedAdvisorCheckSummary`

Herramientas para PowerShell

Ejemplo 1: Devuelve el resumen más reciente de la comprobación de Trusted Advisor especificada.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Ejemplo 2: Devuelve los resúmenes más recientes de las comprobaciones de Trusted Advisor especificadas.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Para API obtener más información, consulte DescribeTrustedAdvisorCheckSummaries AWS Tools for PowerShellCmdlet Reference.

`New-ASACase`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-ASACase`

Herramientas para PowerShell

Ejemplo 1: Crea un nuevo caso en el AWS Support Center. Los valores de los CategoryCode parámetros - ServiceCode y - se pueden obtener mediante el ASAService cmdlet Get-. El valor del SeverityCode parámetro - se puede obtener mediante el cmdlet Get-. ASASeverityLevel El valor del IssueType parámetro - puede ser «servicio al cliente» o «técnico». Si tiene éxito, se AWS mostrará el número de caso de Support. De forma predeterminada, las mayúsculas y minúsculas se gestionarán en inglés. Para usar japonés, añada el parámetro «ja» en el idioma. Los CommunicationBody parámetros -ServiceCode, -CategoryCode, -Subject y - son obligatorios.

```
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"
```

- Para API obtener más información, consulte la referencia CreateCasedel AWS Tools for PowerShell cmdlet.

`Request-ASATrustedAdvisorCheckRefresh`

En el siguiente ejemplo de código se muestra cómo usarlo. `Request-ASATrustedAdvisorCheckRefresh`

Herramientas para PowerShell

Ejemplo 1: Solicita una actualización para la comprobación de Trusted Advisor especificada.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Para API obtener más información, consulte RefreshTrustedAdvisorCheck AWS Tools for PowerShellCmdlet Reference.

`Resolve-ASACase`

En el siguiente ejemplo de código se muestra cómo usarlo. `Resolve-ASACase`

Herramientas para PowerShell

Ejemplo 1: devuelve el estado inicial del caso especificado y el estado actual una vez finalizada la llamada para resolverlo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Para API obtener más información, consulte ResolveCasela referencia del AWS Tools for PowerShell cmdlet.

Ejemplos de Systems Manager que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante Systems Manager.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Add-SSMResourceTag`

El siguiente ejemplo de código muestra cómo usarloAdd-SSMResourceTag.

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza un periodo de mantenimiento con etiquetas nuevas. No se obtienen resultados si el comando se ejecuta correctamente. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear cada etiqueta. No se obtienen resultados si el comando se ejecuta correctamente.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet AddTagsToResourceReference.

`Edit-SSMDocumentPermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-SSMDocumentPermission`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agregan permisos para “compartir” a todas las cuentas de un documento. No se obtienen resultados si el comando se ejecuta correctamente.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Ejemplo 2: en este ejemplo se agregan permisos para “compartir” a una cuenta específica de un documento. No se obtienen resultados si el comando se ejecuta correctamente.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Para API obtener más información, consulte ModifyDocumentPermissionla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMActivation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMActivation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se proporcionan detalles sobre las activaciones de su cuenta.

```
Get-SSMActivation
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeActivationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAssociation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describe la asociación entre una instancia y un documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeAssociationla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAssociationExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAssociationExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelven las ejecuciones del ID de asociación proporcionado

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeAssociationExecutionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAssociationExecutionTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAssociationExecutionTarget`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra el ID del recurso y su estado de ejecución que forman parte de los destinos de ejecución de la asociación

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Salida:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Ejemplo 2: este comando comprueba la ejecución concreta de una automatización concreta desde ayer, a la que está asociado un documento de comandos. Además, también comprueba si se ha producido un error en la ejecución de la asociación y, de ser así, mostrará los detalles de la invocación del comando junto con el ID de la instancia

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.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
    }
}
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeAssociationExecutionTargetsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAssociationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAssociationList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las asociaciones de una instancia. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Salida:

```
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}
```

Ejemplo 2: en este ejemplo se enumeran todas las asociaciones de un documento de configuración. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Salida:

```
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}
```

Ejemplo 3: Con la PowerShell versión 2, debe usar New-Object para crear cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Salida:

```
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}
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet ListAssociationsReference.

`Get-SSMAssociationVersionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAssociationVersionList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recuperan todas las versiones de la asociación proporcionada.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Salida:

```
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}
```

- Para API obtener más información, consulte ListAssociationVersionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAutomationExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAutomationExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles de una ejecución de Automatización.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Salida:

```
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_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU
                            gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUGOAZG-xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (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...}
```

Ejemplo 2: en este ejemplo se enumeran los detalles de los pasos del identificador de ejecución de automatización indicado

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Salida:

```
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    {}
```

- Para API obtener más información, consulte GetAutomationExecutionla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAutomationExecutionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAutomationExecutionList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen todas las ejecuciones de Automatización activas y finalizadas asociadas a su cuenta.

```
Get-SSMAutomationExecutionList
```

Salida:

```
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]]}
```

Ejemplo 2: En este ejemplo, se muestra el identificador de ejecución, el documento y la fecha de inicio y finalización de la ejecución en el caso de las ejecuciones cuyo valor no sea «Éxito» AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Salida:

```
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                    Failed
                   4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Para obtener API más información, consulte Cmdlet Reference. DescribeAutomationExecutionsAWS Tools for PowerShell

`Get-SSMAutomationStepExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAutomationStepExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra información acerca de todas las ejecuciones de pasos activas y finalizadas en un flujo de trabajo de Automatización.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeAutomationStepExecutionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMAvailablePatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMAvailablePatch`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran todos los parches disponibles para Windows Server 2012 que tienen una MSRC gravedad crítica. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
```

```
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Salida:

```
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
...
```

Ejemplo 2: Con la PowerShell versión 2, debe usar New-Object para crear cada filtro.

```
$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
```

Ejemplo 3: En este ejemplo se muestran todas las actualizaciones publicadas en los últimos 20 días y aplicables a los productos correspondientes a 2019 WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

Salida:

```
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)
```

- Para API obtener más información, consulte la referencia DescribeAvailablePatchesde AWS Tools for PowerShell cmdlets.

`Get-SSMCommand`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMCommand`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los comandos solicitados.

```
Get-SSMCommand
```

Salida:

```
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            : {}
```

Ejemplo 2: en este ejemplo se obtiene el estado de un comando específico.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Ejemplo 3: Este ejemplo recupera todos los SSM comandos invocados después de 2019-04-01T 00:00:00 Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

Salida:

```
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
```

- Para obtener más información, consulte Cmdlet Reference. API ListCommandsAWS Tools for PowerShell

`Get-SSMCommandInvocation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMCommandInvocation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las invocaciones de un comando.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

Salida:

```
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        :
```

Ejemplo 2: En este ejemplo se muestra la invocación del identificador de comando CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Salida:

```
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/KUMO.png to ..\..\programdata\KUMO.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
```

- Para obtener ListCommandInvocations más AWS Tools for PowerShell información, consulte la referencia del cmdlet. API

`Get-SSMCommandInvocationDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMCommandInvocationDetail`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles de un comando ejecutado en una instancia.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Salida:

```
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
```

- Para API obtener más información, consulte GetCommandInvocationla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMComplianceItemList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMComplianceItemList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran los elementos de conformidad para el identificador y el tipo de recurso indicados, filtrando el tipo de conformidad por “Asociación”

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Salida:

```
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            :
```

- Para API obtener más información, consulte ListComplianceItemsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMComplianceSummaryList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMComplianceSummaryList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve un recuento resumido de los recursos conformes y no conformes correspondientes a todos los tipos de conformidad.

```
Get-SSMComplianceSummaryList
```

Salida:

```
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
```

- Para API obtener más información, consulte ListComplianceSummariesla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMConnectionStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMConnectionStatus`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se recupera el estado de conexión del Administrador de sesiones de una instancia para determinar si está conectada y lista para recibir las conexiones del Administrador de sesiones.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Salida:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Para API obtener más información, consulte GetConnectionStatusla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDefaultPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDefaultPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra la línea de base de revisiones predeterminada.

```
Get-SSMDefaultPatchBaseline
```

Salida:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Para API obtener más información, consulte GetDefaultPatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDeployablePatchSnapshotForInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDeployablePatchSnapshotForInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra la instantánea actual de la línea de base de revisiones que usa una instancia. Este comando debe ejecutarse desde la instancia con las credenciales de la instancia. Para garantizar que utiliza las credenciales de la instancia, en el ejemplo pasa un objeto **`Amazon.Runtime.InstanceProfileAWSCredentials`** al parámetro Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Salida:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Ejemplo 2: En este ejemplo se muestra cómo obtener el contenido completo SnapshotDownloadUrl. Este comando debe ejecutarse desde la instancia con las credenciales de la instancia. Para garantizar que utiliza las credenciales de la instancia, en el ejemplo se configura la PowerShell sesión para que utilice un **`Amazon.Runtime.InstanceProfileAWSCredentials`** objeto.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Salida:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Para API obtener más información, consulte la referencia GetDeployablePatchSnapshotForInstancede AWS Tools for PowerShell cmdlets.

`Get-SSMDocument`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDocument`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve el contenido de un documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Salida:

```
Content
-------
{...
```

Ejemplo 2: en este ejemplo se muestra el contenido completo de un documento.

```
(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 }}"
         }
      }
   ]
}
```

- Para API obtener más información, consulte GetDocumentla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDocumentDescription`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDocumentDescription`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve información sobre un documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeDocumentla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDocumentList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDocumentList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran todos los documentos de configuración de su cuenta.

```
Get-SSMDocumentList
```

Salida:

```
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
...
```

Ejemplo 2: en este ejemplo se recuperan todos los documentos de automatización cuyo nombre coincida con “Platform”

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

Salida:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Para API obtener más información, consulte ListDocumentsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDocumentPermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDocumentPermission`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las versiones de un documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Salida:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Para API obtener más información, consulte DescribeDocumentPermissionla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMDocumentVersionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMDocumentVersionList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se devuelve la lista de permisos de un documento.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

Salida:

```
all
```

- Para API obtener más información, consulte ListDocumentVersionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMEffectiveInstanceAssociationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMEffectiveInstanceAssociationList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se describen las asociaciones efectivas de una instancia.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

Salida:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Ejemplo 2: en este ejemplo se muestra el contenido de las asociaciones efectivas de una instancia.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Salida:

```
{
    "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 }}"
                }
            ]
        }
    }
}
```

- Para API obtener más información, consulte DescribeEffectiveInstanceAssociationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMEffectivePatchesForPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMEffectivePatchesForPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las líneas de base de revisiones con una lista de resultados máxima de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Salida:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Ejemplo 2: en este ejemplo se muestran los estados de revisión de todas las líneas de base de revisiones con una lista de resultados máxima de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Salida:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Para API obtener más información, consulte DescribeEffectivePatchesForPatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInstanceAssociationsStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInstanceAssociationsStatus`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles de las asociaciones de una instancia.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se comprueba el estado de la asociación de instancias para el ID de instancia dado y, además, se muestra el estado de ejecución de esas asociaciones

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeInstanceAssociationsStatusla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInstanceInformation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInstanceInformation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles de cada una de las instancias.

```
Get-SSMInstanceInformation
```

Salida:

```
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
```

Ejemplo 2: En este ejemplo se muestra cómo utilizar el parámetro -Filter para filtrar los resultados solo a las instancias de AWS Systems Manager de la región **`us-east-1`** con un **`AgentVersion`** de**`2.2.800.0`**. Encontrará una lista de valores clave de -Filter válidos en el tema de InstanceInformation API referencia (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html #systemsmanager -Type- -). InstanceInformation ActivationId

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Salida:

```
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
```

Ejemplo 3: En este ejemplo se muestra cómo utilizar el InstanceInformationFilterList parámetro - para filtrar los resultados solo a las instancias de AWS Systems Manager **`PlatformTypes`** de la región **`us-east-1`** con **`Windows`** o**`Linux`**. Puedes encontrar una lista de valores InstanceInformationFilterList clave válidos en el tema de InstanceInformationFilter API referencia (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

Salida:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
```

```
IPAddress                                : 10.0.0.27
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
```

Ejemplo 4: en este ejemplo se enumeran las instancias gestionadas por SSM y las exportaciones InstanceId PingStatus, LastPingDateTime y PlatformName en un archivo csv.

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- Para API obtener más información, consulte la referencia DescribeInstanceInformationde AWS Tools for PowerShell cmdlets.

`Get-SSMInstancePatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInstancePatch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen los detalles de conformidad de las revisiones de una instancia.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Para API obtener más información, consulte DescribeInstancePatchesla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInstancePatchState`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInstancePatchState`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen los estados resumidos de las revisiones en una instancia.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Ejemplo 2: en este ejemplo se obtienen los estados resumidos de las revisiones en dos instancias.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Para API obtener más información, consulte DescribeInstancePatchStateslа referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInstancePatchStatesForPatchGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInstancePatchStatesForPatchGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen los estados resumidos de las revisiones por instancia en un grupo de revisiones.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Para API obtener más información, consulte DescribeInstancePatchStatesForPatchGroupla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInventory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInventory`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen los metadatos personalizados del inventario.

```
Get-SSMInventory
```

Salida:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Para API obtener más información, consulte GetInventoryla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInventoryEntriesList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInventoryEntriesList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las entradas de inventario personalizadas de una instancia.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se enumeran los detalles.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Salida:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Para API obtener más información, consulte ListInventoryEntriesla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMInventoryEntryList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInventoryEntryList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se recuperan las entradas **`AWS:Network`** de inventario de tipos de la instancia.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Get- SSMInventoryEntryList in.

`Get-SSMInventorySchema`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMInventorySchema`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo devuelve una lista con los nombres de los tipos de inventario de la cuenta.

```
Get-SSMInventorySchema
```

- Para API obtener más información, consulte GetInventorySchemala referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMLatestEC2Image`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMLatestEC2Image`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran todas las versiones más recientes de WindowsAMIs.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Salida:

```
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
...
```

Ejemplo 2: Este ejemplo recupera el AMI identificador de una imagen específica de Amazon Linux para la región us-west-2.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

Salida:

```
ami-09b92cd132204c704
```

Ejemplo 3: en este ejemplo se enumeran las últimas ventanas que AMIs coinciden con la expresión comodín especificada.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Salida:

```
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
...
```

- Para API obtener más información, consulte la referencia del AWS Tools for PowerShell cmdlet Get- SSMLatestEC2Image in.

`Get-SSMMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtienen los detalles sobre un periodo de mantenimiento.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Salida:

```
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
```

- Para API obtener más información, consulte GetMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumera la información sobre una tarea ejecutada como parte de una ejecución de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Salida:

```
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
```

- Para API obtener más información, consulte GetMaintenanceWindowExecutionla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowExecutionList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowExecutionList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las ejecuciones de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Salida:

```
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
```

Ejemplo 2: en este ejemplo se enumeran todas las ejecuciones de un periodo de mantenimiento antes de una fecha especificada.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Ejemplo 3: en este ejemplo se enumeran todas las ejecuciones de un periodo de mantenimiento después de una fecha especificada.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Para API obtener más información, consulte DescribeMaintenanceWindowExecutionsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowExecutionTask`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowExecutionTask`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumera la información sobre una tarea ejecutada que formaba parte de una ejecución de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Salida:

```
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.SimpleSystemsMana
                   meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para API obtener más información, consulte GetMaintenanceWindowExecutionTaskla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowExecutionTaskInvocationList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran las invocaciones de una tarea ejecutada como parte de una ejecución de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Salida:

```
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    :
```

- Para API obtener más información, consulte DescribeMaintenanceWindowExecutionTaskInvocationsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowExecutionTaskList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowExecutionTaskList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran las tareas asociadas a la ejecución de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeMaintenanceWindowExecutionTasksla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los periodos de mantenimiento de su cuenta.

```
Get-SSMMaintenanceWindowList
```

Salida:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Para API obtener más información, consulte DescribeMaintenanceWindowsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowTarget`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowTarget`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los destinos de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeMaintenanceWindowTargetsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMMaintenanceWindowTaskList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMMaintenanceWindowTaskList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las tareas de un periodo de mantenimiento.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeMaintenanceWindowTasksla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMParameterHistory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMParameterHistory`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumera el historial de valores de un parámetro.

```
Get-SSMParameterHistory -Name "Welcome"
```

Salida:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Para API obtener más información, consulte GetParameterHistoryla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMParameterList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMParameterList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todos los parámetros.

```
Get-SSMParameterList
```

Salida:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Para API obtener más información, consulte DescribeParametersla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMParameterValue`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMParameterValue`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran los valores de un parámetro.

```
Get-SSMParameterValue -Name "Welcome"
```

Salida:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Ejemplo 2: en este ejemplo se enumeran los detalles del valor.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Salida:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Para API obtener más información, consulte GetParametersla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran todas las líneas de base de revisiones.

```
Get-SSMPatchBaseline
```

Salida:

```
BaselineDescription                                                 BaselineId
                                                 BaselineName
-------------------                                                 ----------
                                                 ------------
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...
```

Ejemplo 2: En este ejemplo se enumeran todas las líneas base de los parches proporcionadas por. AWS La sintaxis utilizada en este ejemplo requiere la PowerShell versión 3 o posterior.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Salida:

```
Get-SSMPatchBaseline -Filter $filter1
```

Ejemplo 3: en este ejemplo se enumeran todas las líneas de base de revisiones en las que usted es propietario. La sintaxis utilizada en este ejemplo requiere PowerShell la versión 3 o posterior.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Salida:

```
Get-SSMPatchBaseline -Filter $filter1
```

Ejemplo 4: Con la PowerShell versión 2, debe usar New-Object para crear cada etiqueta.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Salida:

```
BaselineDescription                     BaselineId
                          BaselineName             DefaultBaselin

                                                   e
-------------------                     ----------
                          ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribePatchBaselinesReference.

`Get-SSMPatchBaselineDetail`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMPatchBaselineDetail`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestran los detalles de una línea de base de revisiones.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Salida:

```
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 : {}
```

- Para API obtener más información, consulte GetPatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMPatchBaselineForPatchGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMPatchBaselineForPatchGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se muestra la línea de base de revisiones para un grupo de revisiones.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Salida:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para API obtener más información, consulte GetPatchBaselineForPatchGroupla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMPatchGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMPatchGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran los registros de los grupos de revisiones.

```
Get-SSMPatchGroup
```

Salida:

```
BaselineIdentity                                              PatchGroup
----------------                                              ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Para API obtener más información, consulte DescribePatchGroupsla referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMPatchGroupState`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMPatchGroupState`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene un resumen general de conformidad de las revisiones de un grupo de revisiones.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Salida:

```
Instances                       : 4
InstancesWithFailedPatches      : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches   : 3
InstancesWithMissingPatches     : 0
InstancesWithNotApplicablePatches  : 0
```

- Para API obtener más información, consulte DescribePatchGroupStatela referencia de AWS Tools for PowerShell cmdlets.

`Get-SSMResourceComplianceSummaryList`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMResourceComplianceSummaryList`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene un recuento resumido para cada recurso. En el resumen se incluye información sobre los estados conformes y no conformes y recuentos detallados de la gravedad de los elementos de conformidad de los productos que coinciden con “Windows10”.

Como el valor MaxResult predeterminado es 100 si no se especifica el parámetro y este valor no es válido, se agrega el MaxResult parámetro y el valor se establece en 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Para API obtener más información, consulte ListResourceComplianceSummaries AWS Tools for PowerShellCmdlet Reference.

`Get-SSMResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-SSMResourceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se enumeran las etiquetas de un periodo de mantenimiento.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Salida:

```
Key   Value
---   -----
Stack Production
```

- Para API obtener más información, consulte ListTagsForResourcela referencia de AWS Tools for PowerShell cmdlets.

`New-SSMActivation`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMActivation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una instancia administrada.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

Salida:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Para API obtener más información, consulte CreateActivationla referencia de AWS Tools for PowerShell cmdlets.

`New-SSMAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMAssociation`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo asocia un documento de configuración a una instancia mediante instanceIDs.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Salida:

```
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 :
```

Ejemplo 2: en este ejemplo se asocia un documento de configuración a una instancia mediante destinos.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Salida:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Ejemplo 3: en este ejemplo se asocia un documento de configuración a una instancia mediante destinos y parámetros.

```
$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
```

Salida:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Ejemplo 4: en este ejemplo se crea una asociación a todas las instancias de la región con **`AWS-GatherSoftwareInventory`**. También se proporcionan archivos personalizados y ubicaciones de registro en los parámetros que se van a recopilar

```
$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)"
```

Salida:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para API obtener más información, consulte CreateAssociation AWS Tools for PowerShellCmdlet Reference.

`New-SSMAssociationFromBatch`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMAssociationFromBatch`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se asocia un documento de configuración a varias instancias. El resultado devuelve una lista de operaciones correctas y con errores, si corresponde.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Salida:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Ejemplo 2: en este ejemplo se muestran todos los detalles de una operación correcta.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Para API obtener más información, consulte CreateAssociationBatchla referencia de AWS Tools for PowerShell cmdlets.

`New-SSMDocument`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMDocument`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un documento en su cuenta. El documento debe estar en JSON formato. Para obtener más información sobre cómo escribir un documento de configuración, consulte el documento de configuración en la SSM API referencia.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Salida:

```
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
```

- Para API obtener más información, consulte CreateDocumentla referencia de AWS Tools for PowerShell cmdlets.

`New-SSMMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un nuevo periodo de mantenimiento con el nombre especificado que se ejecuta a las 16:00 h todos los martes durante 4 horas, con un límite de 1 hora y que permite la asignación de destinos no asociados.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Salida:

```
mw-03eb53e1ea7383998
```

- Para API obtener más información, consulte CreateMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`New-SSMPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-SSMPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una línea de base de revisiones que aprueba las revisiones siete días después de que Microsoft las publique, para las instancias administradas que ejecutan Windows Server 2019 en un entorno de producción.

```
$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
```

Salida:

```
pb-0z4z6221c4296b23z
```

- Para API obtener más información, consulte CreatePatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Register-SSMDefaultPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-SSMDefaultPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra una línea de base de revisiones como la línea de base de revisiones predeterminada.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Salida:

```
pb-03da896ca3b68b639
```

- Para API obtener más información, consulte RegisterDefaultPatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Register-SSMPatchBaselineForPatchGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-SSMPatchBaselineForPatchGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra una línea de base de revisiones para un grupo de revisiones.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Salida:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Para API obtener más información, consulte RegisterPatchBaselineForPatchGroupla referencia de AWS Tools for PowerShell cmdlets.

`Register-SSMTargetWithMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-SSMTargetWithMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra una instancia con un periodo de mantenimiento.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Salida:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Ejemplo 2: en este ejemplo se registran varias instancias con un periodo de mantenimiento.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Salida:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Ejemplo 3: En este ejemplo se registra una instancia con una ventana de mantenimiento mediante EC2 etiquetas.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Salida:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Para API obtener más información, consulte RegisterTargetWithMaintenanceWindow AWS Tools for PowerShellCmdlet Reference.

`Register-SSMTaskWithMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-SSMTaskWithMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se registra una tarea con un periodo de mantenimiento mediante un ID de instancia. El resultado es el ID de la tarea.

```
$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
```

Salida:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Ejemplo 2: en este ejemplo se registra una tarea con un periodo de mantenimiento mediante un ID de destino. El resultado es el ID de la tarea.

```
$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
```

Salida:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Ejemplo 3: en este ejemplo se crea un objeto de parámetro para el documento de ejecución de comandos **`AWS-RunPowerShellScript`** y se crea una tarea con un periodo de mantenimiento determinado mediante el ID de destino. El resultado devuelto es el ID de la tarea.

```
$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
```

Salida:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Ejemplo 4: En este ejemplo se registra una tarea de automatización de AWS Systems Manager mediante un documento denominado**`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"
```

- Para API obtener más información, consulte RegisterTaskWithMaintenanceWindow AWS Tools for PowerShellCmdlet Reference.

`Remove-SSMActivation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMActivation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina una activación. No se obtienen resultados si el comando se ejecuta correctamente.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Para API obtener más información, consulte DeleteActivationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMAssociation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina la asociación entre una instancia y un documento. No se obtienen resultados si el comando se ejecuta correctamente.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Para API obtener más información, consulte DeleteAssociationla referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMDocument`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMDocument`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina un documento. No se obtienen resultados si el comando se ejecuta correctamente.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Para API obtener más información, consulte DeleteDocumentla referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina un periodo de mantenimiento.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Salida:

```
mw-06d59c1a07c022145
```

- Para API obtener más información, consulte DeleteMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMParameter`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMParameter`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina un parámetro. No se obtienen resultados si el comando se ejecuta correctamente.

```
Remove-SSMParameter -Name "helloWorld"
```

- Para API obtener más información, consulte DeleteParameterla referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina una línea de base de revisiones.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Salida:

```
pb-045f10b4f382baeda
```

- Para API obtener más información, consulte DeletePatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Remove-SSMResourceTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-SSMResourceTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina una etiqueta de un periodo de mantenimiento. No se obtienen resultados si el comando se ejecuta correctamente.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Para API obtener más información, consulte RemoveTagsFromResourcela referencia de AWS Tools for PowerShell cmdlets.

`Send-SSMCommand`

En el siguiente ejemplo de código se muestra cómo usarlo. `Send-SSMCommand`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se ejecuta un comando echo en una instancia de destino.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Salida:

```
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}
```

Ejemplo 2: en este ejemplo se muestra cómo ejecutar un comando que acepte parámetros anidados.

```
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
```

- Para API obtener más información, consulte SendCommandla referencia de AWS Tools for PowerShell cmdlets.

`Start-SSMAutomationExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-SSMAutomationExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se ejecuta un documento que especifica un rol de automatización, un ID de AMI origen y un rol de EC2 instancia de Amazon.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Salida:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Para API obtener más información, consulta StartAutomationExecutionla referencia de AWS Tools for PowerShell cmdlets.

`Start-SSMSession`

En el siguiente ejemplo de código se muestra cómo usarlo. `Start-SSMSession`

Herramientas para PowerShell

Ejemplo 1: este ejemplo inicia una conexión a un destino para una sesión de Session Manager, lo que habilita el reenvío de puertos.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

Salida:

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- Para API obtener más información, consulte StartSessionla referencia de AWS Tools for PowerShell cmdlets.

`Stop-SSMAutomationExecution`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-SSMAutomationExecution`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se detiene una ejecución de Automatización. No se obtienen resultados si el comando se ejecuta correctamente.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Para API obtener más información, consulte StopAutomationExecutionla referencia de AWS Tools for PowerShell cmdlets.

`Stop-SSMCommand`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-SSMCommand`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se intenta cancelar un comando. No se obtienen resultados si la operación se ejecuta correctamente.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Para API obtener más información, consulte CancelCommandla referencia de AWS Tools for PowerShell cmdlets.

`Unregister-SSMManagedInstance`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-SSMManagedInstance`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se anula el registro de una instancia administrada. No se obtienen resultados si el comando se ejecuta correctamente.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Para API obtener más información, consulte DeregisterManagedInstancela referencia de AWS Tools for PowerShell cmdlets.

`Unregister-SSMPatchBaselineForPatchGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-SSMPatchBaselineForPatchGroup`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se anula el registro de un grupo de revisiones de una línea de base de revisiones.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Salida:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para API obtener más información, consulte DeregisterPatchBaselineForPatchGroupla referencia de AWS Tools for PowerShell cmdlets.

`Unregister-SSMTargetFromMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-SSMTargetFromMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina un destino de un periodo de mantenimiento.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Salida:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Para API obtener más información, consulte DeregisterTargetFromMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`Unregister-SSMTaskFromMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-SSMTaskFromMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se elimina una tarea de un periodo de mantenimiento.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Salida:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Para API obtener más información, consulte DeregisterTaskFromMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`Update-SSMAssociation`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMAssociation`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza una asociación con una nueva versión del documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Salida:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para API obtener más información, consulte UpdateAssociationla referencia de AWS Tools for PowerShell cmdlets.

`Update-SSMAssociationStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMAssociationStatus`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza el estado de la asociación entre una instancia y un documento de configuración.

```
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"
```

Salida:

```
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
```

- Para API obtener más información, consulte UpdateAssociationStatusla referencia de AWS Tools for PowerShell cmdlets.

`Update-SSMDocument`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMDocument`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea una nueva versión de un documento con el contenido actualizado del archivo JSON que especifique. El documento debe estar en JSON formato. Puede obtener la versión del documento con el cmdlet «Get-SSMDocumentVersionList».

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

Salida:

```
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
```

- Para API obtener más información, consulte la referencia del UpdateDocument AWS Tools for PowerShellcmdlet.

`Update-SSMDocumentDefaultVersion`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMDocumentDefaultVersion`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza la versión predeterminada de un documento. Puede obtener las versiones de los documentos disponibles con el cmdlet «Get-SSMDocumentVersionList».

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Salida:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Para API obtener más información, consulte la referencia del UpdateDocumentDefaultVersion AWS Tools for PowerShellcmdlet.

`Update-SSMMaintenanceWindow`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMMaintenanceWindow`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza el nombre de un periodo de mantenimiento.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Salida:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Ejemplo 2: en este ejemplo se activa un periodo de mantenimiento.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Salida:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Ejemplo 3: en este ejemplo se desactiva un periodo de mantenimiento.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Salida:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para API obtener más información, consulte UpdateMaintenanceWindowla referencia de AWS Tools for PowerShell cmdlets.

`Update-SSMManagedInstanceRole`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMManagedInstanceRole`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se actualiza el rol de una instancia administrada. No se obtienen resultados si el comando se ejecuta correctamente.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Para API obtener más información, consulte UpdateManagedInstanceRolela referencia de AWS Tools for PowerShell cmdlets.

`Update-SSMPatchBaseline`

En el siguiente ejemplo de código se muestra cómo usarlo. `Update-SSMPatchBaseline`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se agregan dos revisiones como rechazadas y una revisión como aprobada a una línea de base de revisiones existente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Salida:

```
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}
```

- Para API obtener más información, consulte UpdatePatchBaselinela referencia de AWS Tools for PowerShell cmdlets.

`Write-SSMComplianceItem`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-SSMComplianceItem`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se escribe un elemento de conformidad personalizado para la instancia administrada especificada

```
$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-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Para API obtener más información, consulte PutComplianceItemsla referencia de AWS Tools for PowerShell cmdlets.

`Write-SSMInventory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-SSMInventory`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se asigna información de ubicación de bastidores a una instancia. No se obtienen resultados si el comando se ejecuta correctamente.

```
$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
```

- Para API obtener más información, consulte PutInventoryla referencia de AWS Tools for PowerShell cmdlets.

`Write-SSMParameter`

En el siguiente ejemplo de código se muestra cómo usarlo. `Write-SSMParameter`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se crea un parámetro. No se obtienen resultados si el comando se ejecuta correctamente.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Ejemplo 2: en este ejemplo se modifica un parámetro. No se obtienen resultados si el comando se ejecuta correctamente.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Para API obtener más información, consulte PutParameterla referencia de AWS Tools for PowerShell cmdlets.

Ejemplos de Amazon Translate que utilizan herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS Tools for PowerShell mediante Amazon Translate.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`ConvertTo-TRNTargetLanguage`

El siguiente ejemplo de código muestra cómo usarlo`ConvertTo-TRNTargetLanguage`.

Herramientas para PowerShell

Ejemplo 1: convierte el texto en inglés especificado en francés. El texto que se va a convertir también se puede pasar como parámetro -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Para API obtener más información, consulte TranslateText AWS Tools for PowerShellCmdlet Reference.

AWS WAFV2 ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with AWS WAFV2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

Acciones

`New-WAF2WebACL`

El siguiente ejemplo de código muestra cómo usarlo`New-WAF2WebACL`.

Herramientas para PowerShell

Ejemplo 1: Este comando crea una nueva web ACL llamada «waf-test». Tenga en cuenta que, según la API documentación del servicio, 'DefaultAction' es una propiedad obligatoria. Por lo tanto, debe especificarse el valor de '- DefaultAction _Allow' y/o '- DefaultAction _Block'. Como '- DefaultAction _Allow' y '- DefaultAction _Block' no son las propiedades obligatorias, el valor '@ {} 'podría usarse como marcador de posición, como se muestra en el ejemplo anterior.

```
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 @{}
```

Salida:

```
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
```

- Para obtener API más información, consulte la referencia del cmdlet. CreateWebAclAWS Tools for PowerShell

WorkSpaces ejemplos de uso de herramientas para PowerShell

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS Tools for PowerShell with WorkSpaces.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados.

Cada ejemplo incluye un enlace al código fuente completo, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código en su contexto.

Temas

- Acciones

Acciones

`Approve-WKSIpRule`

El siguiente ejemplo de código muestra cómo usarlo`Approve-WKSIpRule`.

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se añaden reglas a un grupo de IP existente

```
$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
```

- Para API obtener más información, consulte AuthorizeIpRules AWS Tools for PowerShellCmdlet Reference.

`Copy-WKSWorkspaceImage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Copy-WKSWorkspaceImage`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se copia la imagen del espacio de trabajo con el identificador especificado de us-west-2 a la región actual con el nombre "» CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

Salida:

```
wsi-456abaqfe
```

- Para API obtener más información, consulte la referencia del CopyWorkspaceImage AWS Tools for PowerShellcmdlet.

`Edit-WKSClientProperty`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSClientProperty`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite la reconexión del cliente de Workspaces

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- Para API obtener más información, consulte la referencia de ModifyClientProperties AWS Tools for PowerShellcmdlets.

`Edit-WKSSelfServicePermission`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSSelfServicePermission`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo habilita los permisos de autoservicio para cambiar el tipo de procesamiento y aumentar el tamaño del volumen del directorio especificado

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Para API obtener más información, consulte ModifySelfservicePermissionsla referencia de AWS Tools for PowerShell cmdlets.

`Edit-WKSWorkspaceAccessProperty`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSWorkspaceAccessProperty`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo permite el acceso a Workspace en Android y Chrome OS para el directorio especificado

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Para API obtener más información, consulte ModifyWorkspaceAccessPropertiesla referencia de AWS Tools for PowerShell cmdlets.

`Edit-WKSWorkspaceCreationProperty`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSWorkspaceCreationProperty`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se activa el acceso a Internet y el modo de mantenimiento como valores predeterminados al crear un espacio de trabajo

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Para API obtener más información, consulte ModifyWorkspaceCreationProperties AWS Tools for PowerShellCmdlet Reference.

`Edit-WKSWorkspaceProperty`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSWorkspaceProperty`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo cambia la propiedad del modo de ejecución del espacio de trabajo a Parada automática para el espacio de trabajo especificado

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Para API obtener más información, consulte ModifyWorkspaceProperties AWS Tools for PowerShellCmdlet Reference.

`Edit-WKSWorkspaceState`

En el siguiente ejemplo de código se muestra cómo usarlo. `Edit-WKSWorkspaceState`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se cambia el estado del espacio de trabajo especificado a Disponible

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- Para API obtener más información, consulte ModifyWorkspaceState AWS Tools for PowerShellCmdlet Reference.

`Get-WKSClientProperty`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSClientProperty`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen las propiedades del cliente de Workspace para el directorio especificado

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- Para API obtener más información, consulte DescribeClientPropertiesla referencia de AWS Tools for PowerShell cmdlets.

`Get-WKSIpGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSIpGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen los detalles del grupo de IP especificado en la región especificada

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Salida:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- Para API obtener más información, consulte DescribeIpGroups AWS Tools for PowerShellCmdlet Reference.

`Get-WKSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSTag`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo busca la etiqueta del espacio de trabajo dado

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Salida:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Para API obtener más información, consulte la referencia DescribeTagsdel AWS Tools for PowerShell cmdlet.

`Get-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: Recupera los detalles de todos los que llevas WorkSpaces a la canalización.

```
Get-WKSWorkspace
```

Salida:

```
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
```

Ejemplo 2: Este comando muestra los valores de las propiedades secundarias de un espacio **`WorkSpaceProperties`** de trabajo de la **`us-west-2`** región. Para obtener más información sobre las propiedades secundarias de**`WorkSpaceProperties`**, consulte https:// docs.aws.amazon.com/workspaces/ API_W orkspaceProperties latest/api/ .html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

Salida:

```
        ComputeTypeName                    : STANDARD
```

```
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Ejemplo 3: Este comando muestra el valor de la propiedad secundaria de un espacio **`RootVolumeSizeGib`** de trabajo **`WorkSpaceProperties`** de la región. **`us-west-2`** El tamaño del volumen de la raíz, en GiB, es 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Salida:

```
        80
```

- Para API obtener más información, consulte la referencia DescribeWorkspacesde AWS Tools for PowerShell cmdlets.

`Get-WKSWorkspaceBundle`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspaceBundle`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se obtienen detalles de todos los paquetes de Workspace de la región actual

```
Get-WKSWorkspaceBundle
```

Salida:

```
BundleId        : wsb-sfhdgv342
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
```

- Para API obtener más información, consulte la referencia de DescribeWorkspaceBundles AWS Tools for PowerShellcmdlets.

`Get-WKSWorkspaceDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspaceDirectory`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se enumeran los detalles del directorio de los directorios registrados

```
Get-WKSWorkspaceDirectory
```

Salida:

```
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
```

- Para API obtener más información, consulte DescribeWorkspaceDirectories AWS Tools for PowerShellCmdlet Reference.

`Get-WKSWorkspaceImage`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspaceImage`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se muestran todos los detalles de todas las imágenes de la región

```
Get-WKSWorkspaceImage
```

Salida:

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- Para API obtener más información, consulte DescribeWorkspaceImages AWS Tools for PowerShellCmdlet Reference.

`Get-WKSWorkspaceSnapshot`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspaceSnapshot`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo muestra la marca de tiempo de la instantánea más reciente creada para el espacio de trabajo especificado

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Salida:

```
RebuildSnapshots                  RestoreSnapshots
----------------                  ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Para API obtener más información, consulte AWS Tools for PowerShell Cmdlet DescribeWorkspaceSnapshotsReference.

`Get-WKSWorkspacesConnectionStatus`

En el siguiente ejemplo de código se muestra cómo usarlo. `Get-WKSWorkspacesConnectionStatus`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se obtiene el estado de conexión del espacio de trabajo especificado

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Para API obtener más información, consulte DescribeWorkspacesConnectionStatus AWS Tools for PowerShellCmdlet Reference.

`New-WKSIpGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-WKSIpGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un grupo de IP vacío denominado FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Salida:

```
wsipg-w45rty4ty
```

- Para API obtener más información, consulte CreateIpGroup AWS Tools for PowerShellCmdlet Reference.

`New-WKSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-WKSTag`

Herramientas para PowerShell

Ejemplo 1: en este ejemplo se añade una etiqueta nueva a un espacio de trabajo denominado**`ws-wsname`**. La etiqueta tiene la clave «Nombre» y el valor clave de**`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
```

Ejemplo 2: en este ejemplo se añaden varias etiquetas a un espacio de trabajo denominado**ws-wsname**. Una etiqueta tiene una clave de «Nombre» y un valor clave de**AWS_Workspace**; la otra etiqueta tiene una clave de etiqueta de «Etapa» y un valor clave de «Prueba».

```
$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
```

- Para API obtener más información, consulte CreateTags AWS Tools for PowerShellCmdlet Reference.

`New-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `New-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: cree un WorkSpace para el paquete, el directorio y el usuario proporcionados.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Ejemplo 2: En este ejemplo se crean varios WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Para API obtener más información, consulte CreateWorkspacesla referencia del AWS Tools for PowerShell cmdlet.

`Register-WKSIpGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-WKSIpGroup`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra el grupo de IP especificado en el directorio especificado

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Para API obtener más información, consulte AssociateIpGroups AWS Tools for PowerShellCmdlet Reference.

`Register-WKSWorkspaceDirectory`

En el siguiente ejemplo de código se muestra cómo usarlo. `Register-WKSWorkspaceDirectory`

Herramientas para PowerShell

Ejemplo 1: Este ejemplo registra el directorio especificado para Workspaces Service

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Para API obtener más información, consulte la referencia RegisterWorkspaceDirectoryde AWS Tools for PowerShell cmdlets.

`Remove-WKSIpGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-WKSIpGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina el grupo de IP especificado

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteIpGroup AWS Tools for PowerShellCmdlet Reference.

`Remove-WKSTag`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-WKSTag`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se elimina la etiqueta asociada al espacio de trabajo

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Salida:

```
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
```

- Para API obtener más información, consulte DeleteTags AWS Tools for PowerShellCmdlet Reference.

`Remove-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `Remove-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: Termina varios WorkSpaces. El uso del modificador -Force impide que el cmdlet solicite confirmación.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Ejemplo 2: recupera la colección de todos los suyos WorkSpaces y los canaliza IDs al WorkSpaceId parámetro - de Remove-WKSWorkspace, finalizando todos los. WorkSpaces El cmdlet avisará antes de que finalice cada uno de ellos. WorkSpace Para suprimir la solicitud de confirmación, agregue el modificador -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Ejemplo 3: En este ejemplo se muestra cómo pasar los TerminateRequest objetos que definen lo WorkSpaces que se va a terminar. El cmdlet solicitará confirmación antes de continuar, a menos que también se especifique el parámetro -Force switch.

```
$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
```

- Para API obtener más información, consulte la referencia del TerminateWorkspacescmdlet AWS Tools for PowerShell .

`Reset-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `Reset-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: Reconstruye lo especificado WorkSpace.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Ejemplo 2: recupera la colección de todos los suyos WorkSpaces y los IDs canaliza al WorkSpaceId parámetro de restablecimientoWKSWorkspace, lo que provoca su WorkSpaces reconstrucción.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Para API obtener más información, consulte RebuildWorkspaces AWS Tools for PowerShellCmdlet Reference.

`Restart-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `Restart-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: Reinicia el especificado WorkSpace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Ejemplo 2: Reinicia varios. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Ejemplo 3: recupera la colección de todos los suyos WorkSpaces y los canaliza IDs al WorkSpaceId parámetro - de Restart-WKSWorkspace, lo que provoca que se WorkSpaces reinicie.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Para API obtener más información, consulte Cmdlet RebootWorkspacesReference AWS Tools for PowerShell .

`Stop-WKSWorkspace`

En el siguiente ejemplo de código se muestra cómo usarlo. `Stop-WKSWorkspace`

Herramientas para PowerShell

Ejemplo 1: detiene varios WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Ejemplo 2: recupera la colección de todos sus datos WorkSpaces y los canaliza IDs al WorkSpaceId parámetro Stop, WKSWorkspace lo que provoca WorkSpaces que se detenga.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Ejemplo 3: En este ejemplo se muestra cómo pasar StopRequest objetos que definen lo WorkSpaces que se debe detener.

```
$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
```

- Para API obtener más información, consulte StopWorkspaces AWS Tools for PowerShellCmdlet Reference.

`Unregister-WKSIpGroup`

En el siguiente ejemplo de código se muestra cómo usarlo. `Unregister-WKSIpGroup`

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se anula el registro del grupo de IP especificado del directorio especificado

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Para API obtener más información, consulte DisassociateIpGroups AWS Tools for PowerShellCmdlet Reference.

Seguridad de este AWS producto o servicio

La seguridad en la nube de Amazon Web Services (AWS) es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre usted AWS y usted. En el modelo de responsabilidad compartida, se habla de “seguridad de la nube” y “seguridad en la nube”:

Seguridad de la nube: AWS se encarga de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la AWS nube y de proporcionarle servicios que pueda utilizar de forma segura. Nuestra responsabilidad en materia de seguridad es nuestra máxima prioridad AWS, y auditores externos comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS conformidad.

Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice y otros factores, como la confidencialidad de sus datos, los requisitos de su organización y las leyes y reglamentos aplicables.

Este AWS producto o servicio sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la página de documentación sobre la seguridad del AWS servicio y AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad.

Temas

- Protección de datos en este AWS producto o servicio
- Identity and Access Management
- Validación de conformidad para este AWS producto o servicio
- Aplicación de una versión mínima de TLS en las Herramientas para PowerShell
- Consideraciones de seguridad adicionales para las herramientas para PowerShell

Protección de datos en este AWS producto o servicio

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en este AWS producto o servicio. Como se describe en este modelo, AWS es responsable de proteger

la infraestructura global en la que se ejecutan todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte la sección Privacidad de datos FAQ. Para obtener información sobre la protección de datos en Europa, consulte el modelo de responsabilidad AWS compartida y la entrada del GDPR blog sobre AWS seguridad.

Para proteger los datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactorial (MFA) con cada cuenta.
- UseSSL/TLSpara comunicarse con AWS los recursos. Necesitamos TLS 1.2 y recomendamos TLS 1.3.
- Configure API y registre la actividad del usuario con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita entre FIPS 140 y 3 módulos criptográficos validados para acceder a AWS través de una interfaz de línea de comandos o unaAPI, utilice un FIPS terminal. Para obtener más información sobre los FIPS puntos finales disponibles, consulte la Norma federal de procesamiento de información () FIPS 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con este AWS producto o servicio u otro Servicios de AWS mediante la consola, API AWS CLI, o. AWS SDKs Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, le recomendamos encarecidamente que no incluya información sobre las credenciales URL para validar su solicitud a ese servidor.

Cifrado de datos

Una característica clave de cualquier servicio seguro es que la información se cifre cuando no se está utilizando activamente.

Cifrado en reposo

Por sí mismo, no almacena ningún dato del cliente aparte de las credenciales que necesita para interactuar con los AWS servicios en nombre del usuario. AWS Tools for PowerShell

Si las utiliza AWS Tools for PowerShell para invocar un AWS servicio que transmite los datos de los clientes a su ordenador local para su almacenamiento, consulte el capítulo sobre seguridad y conformidad de la Guía del usuario de ese servicio para obtener información sobre cómo se almacenan, protegen y cifran esos datos.

Cifrado en tránsito

De forma predeterminada, todos los datos transmitidos desde el ordenador cliente en el que se ejecutan los puntos finales AWS Tools for PowerShell y el AWS servicio se cifran enviándolos a través de una conexiónHTTPS/TLS.

No necesita hacer nada para habilitar el uso deHTTPS/TLS. Siempre está habilitado.

Identity and Access Management

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. IAMlos administradores controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar AWS los recursos. IAMes una Servicio de AWS que puede utilizar sin coste adicional.

Temas

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice AWS.

Usuario del servicio: si Servicios de AWS solía hacer su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más AWS funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una función de AWS, consulte Solución de problemas AWS de identidad y acceso o consulte la guía del usuario de la Servicio de AWS que está utilizando.

Administrador de servicios: si está a cargo de AWS los recursos de su empresa, probablemente tenga acceso total a ellos AWS. Su trabajo consiste en determinar a qué AWS funciones y recursos deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes a su IAM administrador para cambiar los permisos de los usuarios del servicio. Revise la información de esta página para comprender los conceptos básicos deIAM. Para obtener más información sobre cómo puede usarlo IAM su empresa AWS, consulte la guía del usuario del Servicio de AWS que está utilizando.

IAMadministrador: si es IAM administrador, puede que desee obtener más información sobre cómo puede redactar políticas para administrar el acceso AWS. Para ver ejemplos de políticas AWS basadas en la identidad que puede utilizarIAM, consulte la guía del usuario de la Servicio de AWS que está utilizando.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como IAM usuario o asumiendo un IAM rol. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (IAMIdentity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, el administrador configuró previamente la federación de identidades mediante roles. IAM Cuando accede AWS mediante la federación, asume indirectamente un rol.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS incluye un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar AWS API las solicitudes en la Guía del IAM usuario.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactorial (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactorial en la Guía del AWS IAM Identity Center usuario y Uso de la autenticación multifactorial (MFA) AWS en la Guía del IAM usuario.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de tareas que requieren que inicie sesión como usuario root, consulte Tareas que requieren credenciales de usuario root en la Guía del IAM usuario.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en IAM Identity Center, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus aplicaciones Cuentas de AWS . Para obtener información sobre IAM Identity Center, consulte ¿Qué es IAM Identity Center? en la Guía AWS IAM Identity Center del usuario.

Usuarios y grupos de IAM

Un IAMusuario es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos utilizar credenciales temporales en lugar de crear IAM usuarios con credenciales de larga duración, como contraseñas y claves de acceso. Sin embargo, si tiene casos de uso específicos que requieren credenciales a largo plazo con IAM los usuarios, le recomendamos que rote las claves de acceso. Para obtener más información, consulte Rotar las claves de acceso con regularidad para los casos de uso que requieran credenciales de larga duración en la Guía del IAM usuario.

Un IAMgrupo es una identidad que especifica un conjunto de IAM usuarios. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar IAM los recursos.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Cuándo crear un IAM usuario (en lugar de un rol) en la Guía del IAM usuario.

IAMroles

Un IAMrol es una identidad dentro de ti Cuenta de AWS que tiene permisos específicos. Es similar a un IAM usuario, pero no está asociado a una persona específica. Puede asumir temporalmente un IAM rol en el AWS Management Console cambiando de rol. Puede asumir un rol llamando a una AWS API operación AWS CLI o o utilizando una operación personalizadaURL. Para obtener más información sobre los métodos de uso de roles, consulte Uso de IAM roles en la Guía del IAM usuario.

IAMlos roles con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información sobre los roles para la federación, consulte Creación de un rol para un proveedor de identidad externo en la Guía del IAM usuario. Si usa IAM Identity Center, configura un conjunto de permisos. Para controlar a qué pueden acceder sus identidades después de autenticarse, IAM Identity Center correlaciona el conjunto de permisos con un rol en. IAM Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos IAM de usuario temporales: un IAM usuario o rol puede asumir un IAM rol para asumir temporalmente diferentes permisos para una tarea específica.
- Acceso multicuenta: puedes usar un IAM rol para permitir que alguien (un responsable de confianza) de una cuenta diferente acceda a los recursos de tu cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso multicuenta, consulta el tema sobre el acceso a los recursos entre cuentas IAM en la Guía del IAM usuario.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros. Servicios de AWS Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un IAM usuario o un rol para realizar acciones en AWS ellas, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FASutiliza los permisos del principal que llama a an Servicio de AWS, junto con los que solicitan, Servicio de AWS para realizar solicitudes a los servicios descendentes. FASlas solicitudes solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener detalles sobre la política a la hora de realizar FAS solicitudes, consulte Reenviar sesiones de acceso.
  - Función de servicio: una función de servicio es una IAMfunción que un servicio asume para realizar acciones en su nombre. Un IAM administrador puede crear, modificar y eliminar un rol de servicio desde dentroIAM. Para obtener más información, consulte Crear un rol para delegar permisos Servicio de AWS en un rol en el IAMManual del usuario.

  - Función vinculada a un servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un IAM administrador puede ver los permisos de los roles vinculados al servicio, pero no editarlos.
- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un IAM rol para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y que realizan AWS CLI o AWS API solicitan. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Uso de un IAM rol para conceder permisos a aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del IAM usuario.

Para saber si se deben usar IAM roles o IAM usuarios, consulte Cuándo crear un IAM rol (en lugar de un usuario) en la Guía del IAM usuario.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como JSON documentos. Para obtener más información sobre la estructura y el contenido de los documentos de JSON políticas, consulte Descripción general de JSON las políticas en la Guía del IAM usuario.

Los administradores pueden usar AWS JSON políticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder a los usuarios permiso para realizar acciones en los recursos que necesitan, un IAM administrador puede crear IAM políticas. A continuación, el administrador puede añadir las IAM políticas a las funciones y los usuarios pueden asumir las funciones.

IAMlas políticas definen los permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la

acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de AWS Management Console AWS CLI, el o el AWS API.

Políticas basadas en identidad

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener información sobre cómo crear una política basada en la identidad, consulte Creación de IAM políticas en la Guía del usuario. IAM

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y funciones de su empresa. Cuenta de AWS Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para saber cómo elegir entre una política gestionada o una política integrada, consulte Elegir entre políticas gestionadas y políticas integradas en la Guía del IAM usuario.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de JSON política que se adjuntan a un recurso. Algunos ejemplos de políticas basadas en recursos son las políticas de confianza de IAM roles y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puede usar políticas AWS administradas desde una política IAM basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLsson similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de JSON políticas.

Amazon S3 AWS WAF y Amazon VPC son ejemplos de servicios compatiblesACLs. Para obtener más informaciónACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una función avanzada en la que se establecen los permisos máximos que una política basada en la identidad puede conceder a una IAM entidad (IAMusuario o rol). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte los límites de los permisos para IAM las entidades en la Guía del IAMusuario.
- Políticas de control de servicios (SCPs): SCPs son JSON políticas que especifican los permisos máximos para una organización o unidad organizativa (OU) AWS Organizations. AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. SCPLimita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una Usuario raíz de la cuenta de AWS. Para obtener más información sobre OrganizationsSCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información, consulte las políticas de sesión en la Guía del IAM usuario.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud

cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del IAM usuario.

¿Cómo Servicios de AWS trabajar con IAM

Para obtener una visión general de cómo Servicios de AWS funciona con la mayoría de las IAM funciones, consulte AWS los servicios con los que funcionan IAM en la Guía del IAM usuario.

Para obtener información sobre cómo usar una función específica Servicio de AWS IAM, consulta la sección de seguridad de la Guía del usuario del servicio correspondiente.

Solución de problemas AWS de identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con AWS yIAM.

Temas

- No estoy autorizado a realizar ninguna acción en AWS
- No estoy autorizado a realizar tareas como: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS recursos

No estoy autorizado a realizar ninguna acción en AWS

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

El siguiente ejemplo de error se produce cuando el `mateojackson` IAM usuario intenta usar la consola para ver detalles sobre un *`my-example-widget`* recurso ficticio pero no tiene los `awes:`*`GetWidget`* permisos ficticios.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `awes:`*`GetWidget`*.

Si necesita ayuda, póngase en contacto con AWS el administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

El siguiente ejemplo de error se produce cuando un IAM usuario denominado `marymajor` intenta utilizar la consola para realizar una acción en ella. AWS Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con AWS el administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS recursos

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan políticas basadas en recursos o listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para más información, consulte lo siguiente:

- Para saber si AWS es compatible con estas funciones, consulte. ¿Cómo Servicios de AWS trabajar con IAM
- Para obtener información sobre cómo proporcionar acceso a los recursos de su propiedad, consulte Proporcionar acceso a un IAM usuario en otro Cuenta de AWS de su propiedad en la Guía del IAM usuario. Cuentas de AWS

- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo permitir el acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del IAM usuario.
- Para obtener información sobre cómo proporcionar acceso mediante la federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (federación de identidades) en la Guía del IAM usuario.
- Para saber la diferencia entre el uso de roles y políticas basadas en recursos para el acceso entre cuentas, consulte el acceso a recursos entre cuentas IAM en la Guía del usuario. IAM

Validación de conformidad para este AWS producto o servicio

Para saber si un programa de cumplimiento Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseñando una arquitectura basada en la HIPAA seguridad y el cumplimiento en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar las empresas AWS para crear HIPAA aplicaciones aptas.

> **Note**
>
> No todos son aptos. Servicios de AWS HIPAA Para obtener más información, consulta la Referencia de servicios HIPAA aptos.

- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.

- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. En las guías se resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y se orientan a los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, por ejemplo PCIDSS, cumpliendo con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS consumo para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Este AWS producto o servicio sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la página de documentación sobre la seguridad del AWS servicio y AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad.

Aplicación de una versión mínima de TLS en las Herramientas para PowerShell

Para aumentar la seguridad al comunicarse con los servicios de AWS, debe configurar las Herramientas para PowerShell para utilizar la versión de TLS adecuada. Para obtener información

sobre cómo hacerlo, consulte Aplicar una versión mínima de TLS en la Guía para desarrolladores de AWS SDK for .NET.

Consideraciones de seguridad adicionales para las herramientas para PowerShell

Este tema contiene consideraciones de seguridad además de los temas de seguridad tratados en las secciones anteriores.

Registro de información confidencial

Algunas operaciones de esta herramienta pueden devolver información que podría considerarse confidencial, incluida la información de las variables de entorno. La exposición de esta información puede representar un riesgo para la seguridad en algunos escenarios; por ejemplo, la información podría incluirse en los registros de integración y despliegue continuos (CI/CD). Por lo tanto, es importante que revise cuándo incluye dichos resultados como parte de sus registros y los suprima cuando no los necesite. Para obtener información adicional sobre la protección de datos confidenciales, consulteProtección de datos en este AWS producto o servicio.

Tenga en cuenta las siguientes prácticas recomendadas:

- No utilice variables de entorno para almacenar valores confidenciales para sus recursos sin servidor. En su lugar, haga que el código sin servidor recupere mediante programación el secreto de un almacén de secretos (por ejemplo,). AWS Secrets Manager
- Revisa el contenido de tus registros de compilación para asegurarte de que no contengan información confidencial. Considera enfoques como canalizar el archivo a /dev/null o capturar el resultado como un bash o una PowerShell variable para suprimir los resultados de los comandos.
- Tenga en cuenta el acceso a sus registros y limite el acceso de forma adecuada a su caso de uso.

Referencia de cmdlet para las Herramientas para PowerShell

Herramientas para PowerShell proporciona cmdlets que puede usar para acceder a los servicios de AWS. Para ver qué cmdlets están disponibles, consulte la Referencia de cmdlets de AWS Tools for PowerShell.

Historial de documentos

En este tema se describen los cambios importantes realizados en la documentación de AWS Tools for PowerShell.

También actualizamos la documentación periódicamente en respuesta a los comentarios de los clientes. Para enviar sus comentarios acerca de un tema, utilice los botones de comentarios que aparecen junto a “¿Le ha servido de ayuda esta página?” en la parte inferior de cada página.

Para obtener información adicional sobre los cambios y actualizaciones de AWS Tools for PowerShell, consulte las notas de la versión.

Cambio	Descripción	Fecha
Observabilidad	Se agregó información preliminar sobre la observabilidad en el AWS Tools for PowerShell, que permite recopilar datos de telemetría.	13 de septiembre de 2024
Instalando el en Windows AWS Tools for PowerShell	Se agregó información sobre el desbloqueo de ZIP archivos antes de extraer el contenido.	5 de agosto de 2024
Información sobre -Classic EC2	Se ha eliminado la información sobre EC2 -Classic, que ha sido retirada.	1 de agosto de 2024
Ejemplos de código	Se incluyó un capítulo con ejemplos de cmdlets.	17 de abril de 2024
Consideraciones de seguridad adicionales	Se incluye información sobre el posible registro de datos confidenciales.	16 de abril de 2024
Configure la autenticación de herramientas con AWS	Se agregó información sobre SSO el soporte para AWS Tools for PowerShell.	15 de marzo de 2024

Referencia de cmdlets para las herramientas para PowerShell	Se agregó una sección con un enlace a la referencia del PowerShell cmdlet Tools for.	17 de noviembre de 2023
Se incluyeron más actualizaciones de IAM prácticas recomendadas	Guía actualizada para adaptarla a las IAM mejores prácticas. Para obtener más información, consulte las mejores prácticas de seguridad en IAM.	12 de octubre de 2023
Instalación en Windows	Se ha eliminado la información sobre la instalación de las herramientas para Windows PowerShell mediante elMSI, que ha quedado obsoleto.	25 de septiembre de 2023
IAMprácticas recomendadas, actualizaciones	Guía actualizada para alinearla con las IAM mejores prácticas. Para obtener más información, consulte las mejores prácticas de seguridad en IAM.	8 de septiembre de 2023
Canalización y $ AWSHistory	Se ha añadido el parámetro `IncludeSensitiveData` al cmdlet `Set-AWSHistoryConfiguration` .	9 de marzo de 2023
Uso del ClientConfig parámetro en los cmdlets	Se agregó información sobre la compatibilidad con el ClientConfig parámetro.	28 de octubre de 2022
Lance una EC2 instancia de Amazon con Windows PowerShell	Se agregaron notas sobre la retirada de EC2 -Classic.	26 de julio de 2022

AWS Tools for PowerShell Versión 4	Se ha añadido información sobre la versión 4, incluidas instrucciones de instalación para Windows y Linux/macOS, así como un tema sobre migración en el que se explican las diferencias con respecto a la versión 3 y se presentan las nuevas características.	21 de noviembre de 2019
AWS Tools for PowerShell 3.3.563	Se ha agregado información acerca de cómo instalar y utilizar la versión preliminar del módulo `AWS.Tools.Common` . Este nuevo módulo divide el paquete monolítico anterior en un módulo compartido y un módulo por servicio. AWS	18 de octubre de 2019
AWS Tools for PowerShell 3.3.343.0	Se ha añadido información a la AWS Tools for PowerShell sección Uso de la AWS Lambda herramienta PowerShell para que los desarrolladores PowerShell principales puedan crear funciones. AWS Lambda	11 de septiembre de 2018

AWS Tools for Windows PowerShell 3.1.31.0	Se agregó información a la sección de introducción sobre los nuevos cmdlets que utilizan Security Assertion Markup Language (SAML) para permitir la configuración de la identidad federada de los usuarios.	1 de diciembre de 2015
AWS Tools for Windows PowerShell 2.3.19	Se agregó información sobre el nuevo cmdlet a la sección Descubrimiento y alias de los cmdlets, que puede ayudar a los usuarios a encontrar más fácilmente los `Get-AWSCmdletName` cmdlets que desean. AWS	5 de febrero de 2015

AWS Tools for Windows PowerShell 1.1.1.0	Los resultados de la recopilación de los cmdlets siempre se enumeran en la canalización. PowerShell Compatibilidad automática con llamadas a servicios paginables. La nueva variable $ AWSHistory shell recopila las respuestas de servicio y, opcionalmente, las solicitudes de servicio. AWSRegionlas instancias utilizan el campo Región en lugar de SystemName para facilitar la canalización. Remove-S3Bucketadmite la opción de DeleteObjects cambio -. Se ha corregido un problema de usabilidad con Set-AWSCredentials. Inicializar: AWSDefaults informa de dónde obtuvo las credenciales y los datos de la región. Stop-EC2Instanceacepta Amazon. EC2Instancias de .Model.Reservation como entrada. Los tipos de parámetros List<T> genéricos se han reemplazado por tipos de matriz (T[]). Los cmdlets que eliminan o terminan recursos piden confirmación antes de eliminarlos. Write-S3Objectadmite la carga de contenido de texto en línea a Amazon S3.	15 de mayo de 2013

AWS Tools for Windows PowerShell 1.0.1.0	La ubicación de instalación del PowerShell módulo Herramientas para Windows ha cambiado para que los entornos que utilizan la PowerShell versión 3 de Windows puedan aprovechar la carga automática. El módulo y los archivos auxiliares ahora se instalan en una subcarpeta de `AWSPowerShell` en `AWS ToolsPowerShell`. El instalador elimina de forma automática los archivos de las versiones anteriores que existen en la carpeta de `AWS ToolsPowerShell`. El módulo `PSModulePath` para Windows PowerShell (todas las versiones) se ha actualizado en esta versión para incluir la carpeta principal del módulo (`AWS ToolsPowerShell`). Para los sistemas con la PowerShell versión 2 de Windows, el acceso directo del menú Inicio se actualiza para importar el módulo desde la nueva ubicación y, a continuación, ejecutar `Initialize-AWSDefaults`. Para los sistemas con la PowerShell versión 3 de Windows, el acceso directo del menú Inicio se actualiza para eliminar el `Import-`	21 de diciembre de 2012

	`Module` comando y dejar solo`Initialize-AWSDefaults` . Si ha editado su PowerShell perfil para ejecutar una `Import-Module` parte del `AWSPowerShell.psd1` archivo, tendrá que actualizarlo para que apunte a la nueva ubicación del archivo (o, si utiliza la PowerShell versión 3, eliminar la `Import-Module` sentencia, ya que ya no es necesaria). Como resultado de estos cambios, el PowerShell módulo Herramientas para Windows ahora aparece como módulo disponible durante la ejecución`Get-Module -ListAvailable` . Además, para los usuarios de la PowerShell versión 3 de Windows, la ejecución de cualquier cmdlet exportado por el módulo cargará automáticamente el módulo en el PowerShell shell actual sin necesidad de usarlo `Import-Module` primero. Esto permite el uso interactivo de los cmdlets en un sistema con una política de ejecución que no permite la ejecución de scripts.	
AWS Tools for Windows PowerShell 1.0.0.0	Versión inicial	6 de diciembre de 2012

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.