Guía del usuario

AWS Tools for PowerShell

AWS Tools for PowerShell: Guía del usuario

Table of Contents

¿Qué son las AWS Tools for PowerShell?

Las AWS Tools for PowerShell son un conjunto de módulos PowerShell basados en la funcionalidad expuesta en el AWS SDK for .NET. Las AWS Tools for PowerShell le permiten realizar operaciones mediante scripts en sus recursos de AWS desde la línea de comandos de PowerShell.

Los cmdlets proporcionan una experiencia de PowerShell idiomático para especificar parámetros y administrar los resultados incluso si se implementan mediante diversas API de consulta HTTP del servicio de AWS. Por ejemplo, los cmdlets de AWS Tools for PowerShell admiten la canalización de PowerShell, es decir, puede canalizar objetos de PowerShell dentro y fuera de los cmdlets.

Las AWS Tools for PowerShell son flexibles en cuanto a la forma en que permiten controlar las credenciales, incluido el soporte para la infraestructura de AWS Identity and Access Management (IAM). Puede utilizar las herramientas con credenciales de usuario de IAM, tokens de seguridad temporales y roles de IAM.

Las AWS Tools for PowerShell admiten el mismo conjunto de servicios y regiones de AWS que el SDK. Puede instalar las AWS Tools for PowerShell en equipos que ejecuten sistemas operativos basados en Windows, Linux o macOS.

> **Note**
>
> La versión 4 de AWS Tools for PowerShell es la última versión principal y es una actualización compatible con versiones anteriores a la versión 3.3 de 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 versión 3.3 de AWS Tools for PowerShell a la versión 4.

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

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

Mantenimiento y compatibilidad de las versiones principales del SDK

Para obtener información sobre el mantenimiento y la compatibilidad con las principales versiones del SDK y sus dependencias subyacentes, consulte lo siguiente en la Guía de Referencia de SDK y herramientas de AWS:

- Política de mantenimiento de SDK y herramientas AWS
- Matriz de compatibilidad para versiones de SDK y herramientas AWS

`AWS.Tools`: una versión en módulos de las AWS Tools for PowerShell

Esta versión de las AWS Tools for PowerShell es la versión recomendada para cualquier equipo que ejecute 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 de las 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, `AWS.Tools.Common` y un módulo para cada servicio de AWS; por ejemplo: `AWS.Tools.EC2`, `AWS.Tools.IAM`, `AWS.Tools.S3`, etc.

El módulo `AWS.Tools.Installer` proporciona cmdlets que le permiten instalar, actualizar y quitar los módulos de cada uno de los servicios de AWS. 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 utilizar los cmdlets de un servicio de AWS, simplemente ejecute el comando. PowerShell importa automáticamente el módulo `AWS.Tools.Common` y el módulo del

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

Puede instalar esta versión de las AWS Tools for PowerShell en equipos que estén ejecutando:

- PowerShell Core 6.0 o posterior en Windows, Linux o macOS.
- Windows PowerShell 5.1 o versiones posteriores en Windows con .NET Framework 4.7.2 o superior.

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 de las AWS Tools for PowerShell

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

Puede instalar esta versión de las AWS Tools for PowerShell en equipos que estén ejecutando:

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

En esta guía, cuando necesitemos especificar solo esta versión, nos referiremos a ella por su nombre de módulo: AWSPowerShell.NetCore.

AWSPowerShell: una versión de un solo módulo para Windows PowerShell

Esta versión de las AWS Tools for PowerShell solo es compatible con equipos Windows que ejecutan las versiones 2.0 a 5.1 de Windows PowerShell y solo se puede instalar en estos equipos. 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 admite todos los servicios de AWS.

En esta guía, cuando necesitemos especificar solo esta versión, nos referiremos a ella por su nombre de 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 las AWS Tools for PowerShell. Incluye cómo registrarse en AWS si aún no tiene una cuenta y cómo crear un usuario de IAM que puede utilizar para ejecutar los cmdlets.

Comenzar a utilizar la AWS Tools for Windows PowerShell

> En esta sección se describen los aspectos básicos del uso de las AWS Tools for PowerShell, como, por ejemplo, la especificación de credenciales y regiones de AWS, la búsqueda de cmdlets para un determinado servicio y el uso de alias de cmdlets.

Trabajar con servicios de AWS en AWS Tools for PowerShell

> Esta sección incluye información sobre el uso de las AWS Tools for PowerShell para realizar algunas de las tareas más comunes en AWS.

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: versión de AWS Tools for PowerShell que se compone de un solo módulo heredado de gran tamaño que es específico de Windows. Todos los AWS servicios son compatibles con este módulo único y grande.

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

> **Note**
>
> Las herramientas para Windows PowerShell (AWSPowerShell módulo) se instalan de forma predeterminada en todas las Amazon Machine Images (AMI) 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`.
- Instalar cada módulo de servicio de la PowerShell Galería mediante el `Install-Module` cmdlet.

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 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 de AWS 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 desde un repositorio de `PSRepository` llamado `PSGallery` (https://www.powershellgallery.com/) y lo considera un origen 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 módulo de IAM en la carpeta `%ProgramFiles%\WindowsPowerShell\Modules` que es accesible para 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`.
- 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 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`.
- 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 ha instalado PowerShellGetla versión 3 o una versión posterior. PowerShell 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 (AMI) basadas en Windows.

Habilitar la ejecución de scripts

Para cargar los AWS Tools for PowerShell módulos, debe habilitar la ejecución de PowerShell scripts. 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 Tools for Windows actualiza automáticamente el PS ModulePath para 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 VersionTable automática $PS.

```
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 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 .

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 realizó la instalación mediante un archivo ZIP 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

esUnrestricted. 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
- Trabajar con servicios de AWS en AWS Tools for PowerShell

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

La versión 4 de AWS Tools for PowerShell es una actualización compatible con versiones anteriores a la versión 3.3 de 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

El AWSPowerShell. NetCore y AWSPowerShell los paquetes eran «monolíticos». es decir, todos los servicios de AWS se basaban en el mismo módulo, lo que lo hacía muy grande, ya que iba creciendo a medida que se agregaba cada nueva característica y servicio de AWS. El nuevo paquete `AWS.Tools` se divide en módulos más pequeños que le ofrecen flexibilidad para descargar e instalar

únicamente aquellos módulos que son necesarios para los servicios de AWS que utiliza. 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 los objetos de.NET asociados antes de llamar a un cmdlet, debe seguir llamando `Import-Module` para PowerShell informar sobre los tipos de.NET pertinentes.

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 detectar los nombres de los módulos de cada servicio de AWS entre la colección de módulos de `AWS.Tools`, puede utilizar el cmdlet `Get-AWSService`.

```
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 por usted a las API de los servicios de AWS utilizando 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 parámetro `-Select` permite especificar el contenido que se va a incluir en el objeto .NET devuelto por el cmdlet.

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

```
PS > Get-S3Object -BucketName mybucket

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

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

En la versión 4 de AWS Tools for PowerShell, puede especificar `-Select *` para obtener el objeto de respuesta .NET completo devuelto por la llamada a la API del SDK.

```
PS > Get-S3Object -BucketName mybucket -Select *
```

```
  IsTruncated    : False
  NextMarker     :
  S3Objects      : {file1.txt, file2.txt}
  Name           : mybucket
  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 mybucket -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 mybucket -Select S3Objects.Key |
>> Write-S3ObjectTagSet -Select ^Key -BucketName mybucket -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 parámetro `-MaxItems` para especificar el número máximo de objetos que se devolvían 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 la API del servicio subyacente es compatible con el parámetro `MaxItems`, seguirá estando disponible y funcionará tal y como especifica la API. 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 el resultado al cmdlet `Select-Object` y especifique el parámetro `-First` *n*, donde *n* es el número máximo de elementos que se van a incluir en la salida final.

```
PS > Get-S3ObjectV2 -BucketName BUCKET_NAME -Select S3Objects.Key | select -first 2
file1.txt
file2.txt
```

No todos los servicios de AWS admiten `-MaxItems` de la misma manera, por lo que esto elimina esa inconsistencia 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 a `byte[]` utilizando 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='myBucket'; Key='file1.txt'; PartNumber=1 } | Get-S3ObjectMetadata
```

> **Note**
>
> Algunas propiedades ya admitían esto 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`, `SecretKey`, `ProfileName` o `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 se requiere un parámetro de una API, PowerShell le solicita el parámetro del cmdlet correspondiente si no lo especificó. 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 al servicio de la API de Amazon EC2 no se ejecutará correctamente, ya que el servicio de AWS aún necesita 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 características se quedaron obsoletas en versiones anteriores de AWS Tools for PowerShell y se han quitado 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
- Configurar 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.

Configurar la identidad federada con AWS Tools for PowerShell

Para permitir que los usuarios de su organización obtengan acceso a los recursos de AWS, debe configurar un método de autenticación estándar y repetible para fines de seguridad, auditabilidad, conformidad y para permitir la separación de roles y cuentas. Aunque es habitual proporcionar a los usuarios la posibilidad de obtener acceso a las API de AWS, sin acceso federado a la API, también tendría que crear usuarios de AWS Identity and Access Management (IAM), lo que haría que utilizar la federación pierda sentido. En este tema se describe la compatibilidad de SAML (Security Assertion Markup Language) con las AWS Tools for PowerShell, que facilita la solución de acceso federado.

La compatibilidad de SAML con las AWS Tools for PowerShell le permite proporcionar a los usuarios acceso federado a servicios de AWS. SAML es un formato estándar abierto basado en XML para transmitir datos de autenticación y autorización de los usuarios entre servicios y, en concreto, entre un proveedor de identidad (como Active Directory Federation Services) y un proveedor de servicios (como AWS). Para obtener más información acerca de SAML y su funcionamiento, consulte SAML en Wikipedia o SAML Technical Specifications en el sitio web de Organization for the Advancement of Structured Information Standards (OASIS). La versión de SAML compatible con las AWS Tools for PowerShell es SAML 2.0.

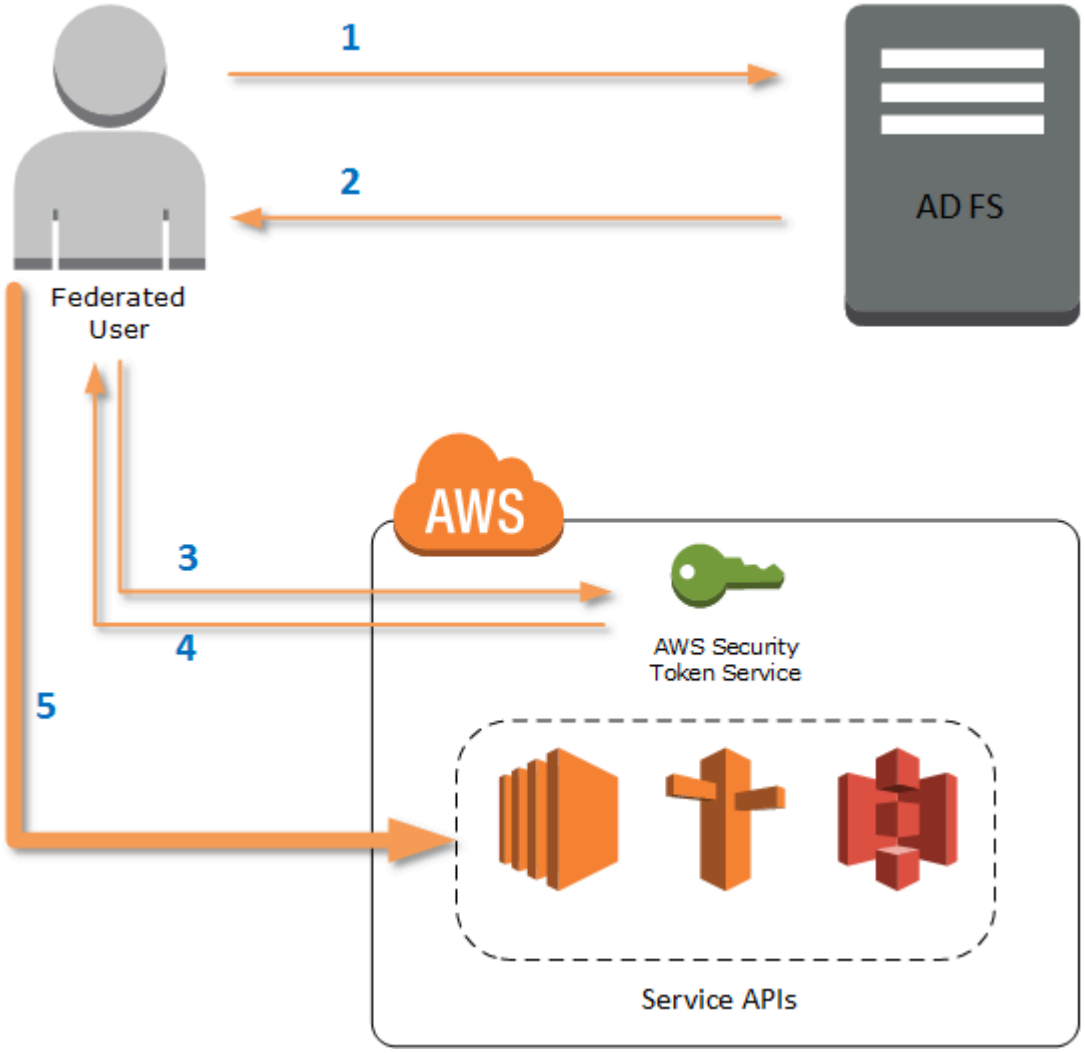
Requisitos previos

Antes de intentar usar la compatibilidad de SAML por primera vez, debe disponer de lo siguiente.

- 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 hacer esto, en concreto para Active Directory Federation Services, consulte Acerca de la federación basada en SAML 2.0 en la Guía del usuario de IAM y la entrada de blog Enabling Federation to AWS Using Windows Active Directory, AD FS, and 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.
- Version 3.1.31.0 o posterior de las AWS Tools for PowerShell instalada en su estación de trabajo local.

Cómo un usuario con identidad federada obtiene acceso federado a las API de servicios de AWS

En el siguiente proceso, se describe a grandes rasgos cómo AD FS federa a un usuario de Active Directory (AD) para obtener acceso a los recursos de AWS.

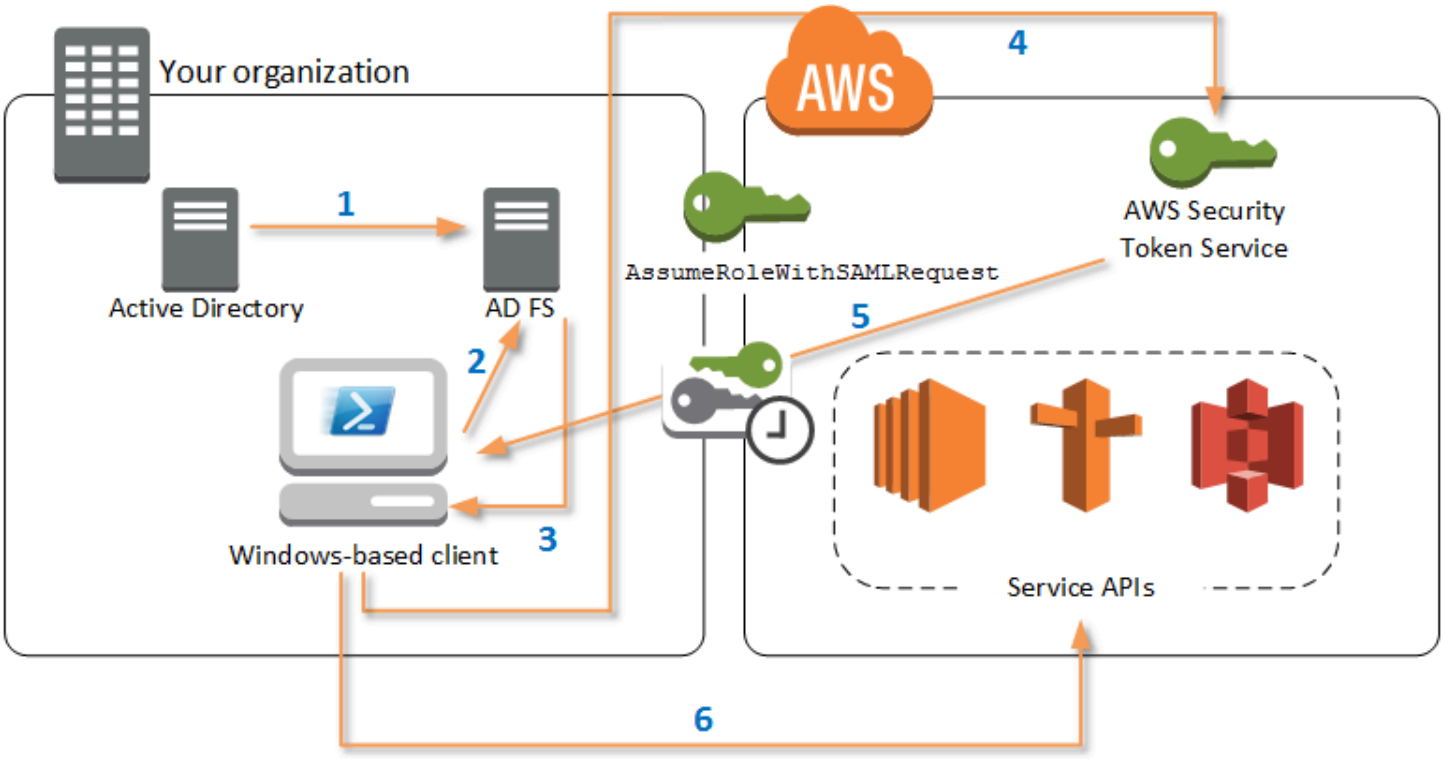


1. El cliente en el equipo del usuario federado se autentica en AD FS.
2. Si la autenticación se realiza correctamente, AD FS envía al usuario una aserción SAML.
3. El cliente del usuario envía la aserción SAML a AWS Security Token Service (STS) como parte de una solicitud de federación SAML.
4. STS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

5. El usuario accede a las API del servicio de AWS mediante la inclusión de las credenciales temporales solicitadas por las AWS Tools for PowerShell.

Cómo funciona la compatibilidad de SAML en las AWS Tools for PowerShell

En esta sección se describe cómo los cmdlets de las AWS Tools for PowerShell permiten la configuración de identidades federadas basadas en SAML para los usuarios.



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

5. Si la solicitud SAML es válida, STS devuelve una respuesta que contiene `AccessKeyId`, `SecretAccessKey` y `SessionToken` de AWS. Estas credenciales duran 3 600 segundos (1 hora).
6. El usuario ahora tiene credenciales válidas para trabajar con las API de los servicios de AWS a las que el rol de usuario tiene permiso de acceso. AWS Tools for PowerShell aplica de forma automática estas credenciales a todas las siguientes llamadas a la API de AWS y las renueva de forma automática cuando vencen.

> **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. Para las cuentas que no están unidas a un dominio, AWS Tools for PowerShell les pide a los usuarios que especifiquen sus credenciales antes de volver a autenticarse.

Cómo usar los cmdlets de configuración de SAML de PowerShell

AWS Tools for PowerShell incluye dos nuevos cmdlets que proporcionan compatibilidad con 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 credenciales de AWS, puede asignar un nombre descriptivo al perfil del rol. Puede utilizar el mismo nombre descriptivo con el cmdlet `Set-AWSCredential` o como el valor del parámetro `-ProfileName` para cualquier cmdlet que invoque API de servicios de AWS.

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

cmdlets. Si ejecuta PowerShell 2.0, debe importar el módulo manualmente. Para ello, ejecute el cmdlet Import-Module, tal y 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 sustituir los marcadores de posición de ID de cuenta y nombre de host de AD FS por sus propios ID de cuenta y nombre de host de AD-FS. 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 cmdlet `Set-AWSSamlRoleProfile` para autenticarse con el proveedor de identidad de AD FS y obtener el conjunto de roles (en la aserción SAML) que el usuario tiene permiso para realizar.

   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
```

De forma alternativa, para las cuentas que no están unidas al dominio, puede proporcionar las credenciales de Active Directory y, a continuación, seleccionar un rol de AWS al que el usuario tenga acceso, tal y 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 los roles de IAM con las 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 utilizar perfiles de rol para ejecutar cmdlets que requieren credenciales de AWS

Para ejecutar cmdlets que requieran credenciales de AWS, puede utilizar perfiles de rol definidos en el archivo de credenciales compartidas de AWS. Proporcione el nombre de un perfil de rol a `Set-AWSCredential` (o como el valor de cualquier parámetro `ProfileName` en las AWS Tools for PowerShell) para obtener de forma automática credenciales temporales de AWS 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 utilizar las credenciales temporales de AWS obtenidas con el perfil `SAMLDemoProfile` para trabajar con las API de servicios de AWS. En las secciones siguientes se muestran ejemplos de cómo utilizar los perfiles de rol.

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

Este ejemplo define un rol predeterminado para una sesión de AWS Tools for PowerShell 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 rol durante una sesión de PowerShell

En este ejemplo, se muestran todos los buckets de Amazon S3 disponibles en la cuenta de AWS del rol asociado al perfil `SAMLDemoProfile`. El ejemplo muestra que, aunque es posible que haya usado otro perfil anteriormente en su sesión de AWS Tools for PowerShell, puede cambiar de perfil especificando un valor diferente para el parámetro `-ProfileName` con cmdlets que lo admitan. Se trata de una tarea común para los administradores que administran Amazon S3 desde la línea de comandos de PowerShell.

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

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

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`.

Aunque las credenciales caduquen al cabo de una hora (un límite impuesto por STS), AWS Tools for PowerShell actualiza automáticamente las credenciales solicitando una nueva aserción SAML cuando las herramientas detectan 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. Para las cuentas de usuario no unidas a un dominio, AWS Tools for PowerShell muestra un mensaje de credenciales de PowerShell en el 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 muestran todas las instancias de Amazon EC2 en la región Asia-Pacífico (Sídney) que están asociadas con la cuenta que usa el perfil `ADFS-Production`. Este comando es útil para devolver todas las instancias de Amazon EC2 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 acerca de cómo implementar el acceso a API federado, consulte la página sobre cómo implementar una solución general para el acceso a API o CLI federado mediante SAML 2.0.

Si tiene alguna pregunta o comentario, visite los foros para desarrolladores de AWS de PowerShell Scripting o .NET Development.

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`.

Trabajar con servicios de AWS en AWS Tools for PowerShell

En esta sección se proporcionan ejemplos sobre cómo utilizar AWS Tools for PowerShell para obtener acceso a los servicios de AWS. En estos ejemplos se muestra cómo utilizar los cmdlets para realizar tareas reales de AWS. Estos ejemplos se basan en los cmdlets que proporciona Herramientas para PowerShell. Para ver qué cmdlets están disponibles, consulte la Referencia de cmdlets de AWS Tools for PowerShell.

Codificación de la concatenación de archivos de PowerShell

Algunos cmdlets de las AWS Tools for PowerShell editan los archivos o registros existentes que tiene en AWS. Un ejemplo es `Edit-R53ResourceRecordSet`, que llama a la API ChangeResourceRecordSets de Amazon Route 53.

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

Para evitar que la salida de archivos se convierta a UTF-16, puede canalizar el comando en el cmdlet `Out-File` de PowerShell y especificar la codificación UTF-8, tal y como se muestra en el siguiente ejemplo:

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

Si ejecuta comandos de la AWS CLI en la consola de PowerShell, se aplica el mismo comportamiento. Puede canalizar la salida de un comando de la AWS CLI en `Out-File` en la consola de PowerShell. 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 6.0 y versiones posteriores, incluido PowerShell Core, conserva automáticamente la codificación UTF-8 para la salida de archivos concatenados.

Objetos devueltos para herramientas de PowerShell

Para que las AWS Tools for PowerShell sean más útiles en un entorno nativo de PowerShell, el objeto devuelto por un cmdlet de AWS Tools for PowerShell es un objeto .NET, no el objeto de texto JSON que normalmente se devuelve desde la API correspondiente en el SDK de AWS. Por ejemplo, `Get-S3Bucket` emite una colección de `Buckets`, no un objeto de respuesta JSON de Amazon S3. La colección de `Buckets` se puede colocar en la canalización de PowerShell e interactuar con ella de manera apropiada. Del mismo modo, `Get-EC2Instance` emite una colección de objetos .NET `Reservation`, no un objeto resultante JSON `DescribeEC2Instances`. Este comportamiento es así por diseño con el objetivo de que la experiencia de las AWS Tools for PowerShell sea más coherente con PowerShell idiomático.

Las respuestas de servicio reales están disponibles para usted si las necesita. Se almacenan como propiedades `note` en los objetos devueltos. Para las acciones de la API que admiten paginación mediante campos `NextToken`, estas respuestas también se asocian como propiedades `note`.

Amazon EC2

En esta sección se describen los pasos necesarios para lanzar una instancia de Amazon EC2 incluidos los siguientes:

- Recuperar una lista de imágenes de máquina de Amazon (AMI)
- Cree un par de claves para la autenticación SSH.
- crear y configurar un grupo de seguridad de Amazon EC2
- 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 presenta información general breve del módulo AWS Lambda Tools for PowerShell y se describen los pasos necesarios para configurar el módulo.

Amazon SNS y Amazon SQS

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

- Cree un tema de Amazon SNS.
- Cree una cola de Amazon SQS.
- 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 sobre cómo publicar datos personalizados en CloudWatch.

- Publicar una métrica personalizada en el panel de CloudWatch

Véase también

- Comenzar a utilizar la AWS Tools for Windows PowerShell

Temas

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

- Trabajar con servicios de AWS en 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)

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

- Trabajar con servicios de AWS en 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

- Trabajar con servicios de AWS en 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

- Trabajar con servicios de AWS en 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

- Trabajar con servicios de AWS en 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 mybucket -key myobject.txt -content "file content"

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

```
>> line 2
>> line 3
>> "@
>>
PS > write-s3object mybucket -key myobject.txt -content $x
```

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
- Creación de un grupo de seguridad mediante Windows PowerShell
- Buscar una imagen de máquina de Amazon mediante Windows PowerShell
- Lanzamiento de una instancia de Amazon EC2 con Windows PowerShell

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 usar las AWS Tools for PowerShell para crear y configurar un grupo de seguridad. Cuando crea un grupo de seguridad, especifica si es para EC2-Classic o EC2-VPC. La respuesta es el ID del grupo de seguridad.

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

Temas

Requisitos previos

Necesitará la dirección IP pública de su equipo, en notación CIDR. 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 ISP o desde detrás del firewall sin una dirección IP estática, deberá encontrar el intervalo de direcciones IP que pueden utilizar los equipos cliente.

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

Creación de un grupo de seguridad para EC2-Classic

> **Warning**
>
> Vamos a retirar EC2-Classic el 15 de agosto de 2022. Le recomendamos que migre de EC2-Classic a una VPC. Para obtener más información, consulte Migración de EC2-Classic a

una VPC en la Guía del usuario de Amazon EC2 para instancias de Linux o en la Guía del usuario de Amazon EC2 para instancias de Windows. Consulte también la entrada de blog EC2-Classic Networking is Retiring – Here's How to Prepare (Se va a retirar la red EC2-Classic: cómo prepararse).

En el siguiente ejemplo se utiliza el cmdlet `New-EC2SecurityGroup` para crear un grupo de seguridad para `EC2-Classic`.

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

sg-0a346530123456789
```

Para ver la configuración inicial del grupo de seguridad, use el cmdlet `Get-EC2SecurityGroup`.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

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

Para configurar el grupo de seguridad para permitir el tráfico entrante en el puerto TCP 22 (SSH) y el puerto TCP 3389, utilice el cmdlet `Grant-EC2SecurityGroupIngress`. Por ejemplo, el siguiente script muestra cómo se podría habilitar el tráfico SSH desde una única dirección IP, `203.0.113.25/32`.

```
$cidrBlocks = New-Object 'collections.generic.list[string]'
$cidrBlocks.add("203.0.113.25/32")
$ipPermissions = New-Object Amazon.EC2.Model.IpPermission
$ipPermissions.IpProtocol = "tcp"
$ipPermissions.FromPort = 22
$ipPermissions.ToPort = 22
ipPermissions.IpRanges = $cidrBlocks
```

```
Grant-EC2SecurityGroupIngress -GroupName myPSSecurityGroup -IpPermissions
 $ipPermissions
```

Para verificar que el grupo de seguridad se ha actualizado, ejecute de nuevo el cmdlet `Get-EC2SecurityGroup`. Tenga en cuenta que no puede especificar una regla de salida para `EC2-Classic`.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup

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

Para ver la regla del grupo de seguridad, utilice la propiedad `IpPermissions`.

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

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

Creación de un grupo de seguridad para EC2-VPC

El siguiente ejemplo de `New-EC2SecurityGroup` añade un parámetro `-VpcId` para crear un grupo de seguridad para la VPC especificada.

```
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 una VPC incluye una regla que permite todo el

tráfico de salida. Tenga en cuenta que no puede hacer referencia a un grupo de seguridad de EC2-VPC por 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 para el tráfico entrante en el puerto TCP 22 (SSH) y el puerto TCP 3389, utilice el cmdlet `New-Object`. En el siguiente script de ejemplo se definen los permisos para los puertos TCP 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
```

Lanzamiento de una instancia de Amazon EC2 con Windows PowerShell

Para lanzar una instancia Amazon EC2, necesita el par de claves y el grupo de seguridad que ha creado en las secciones anteriores. También necesita el ID de una imagen de máquina de Amazon (AMI). Para obtener más información, consulte la documentación siguiente:

- Creación de un par de claves
- Creación de un grupo de seguridad mediante Windows PowerShell
- Buscar una imagen de máquina de Amazon mediante 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 EC2-Classic
- Lanzamiento de una instancia en una VPC
- Lanzamiento de una instancia de subasta en una VPC

Lanzamiento de una instancia en EC2-Classic

> **Warning**
>
> Vamos a retirar EC2-Classic el 15 de agosto de 2022. Le recomendamos que migre de EC2-Classic a una VPC. Para obtener más información, consulte Migración de EC2-Classic a una VPC en la Guía del usuario de Amazon EC2 para instancias de Linux o en la Guía del usuario de Amazon EC2 para instancias de Windows. Consulte también la entrada de blog EC2-Classic Networking is Retiring – Here's How to Prepare (Se va a retirar la red EC2-Classic: cómo prepararse).

El siguiente comando crea un sola instancia `t1.micro` y la lanza.

```
PS > New-EC2Instance -ImageId ami-c49c0dac `
```

```
    -MinCount 1 `
    -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroups myPSSecurityGroup `
    -InstanceType t1.micro

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

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-5caa4371")
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =
 "reservation-id"; Values = $reservation}
PS > (Get-EC2Instance -Filter $filter_reservation).Instances

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

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

Lanzamiento de una instancia en una VPC

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 de subasta en una 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 la 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
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yc
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

- Trabajar con servicios de AWS en 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.

Seguridad de este producto o servicio de AWS

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 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 es responsable de proteger la infraestructura que ejecuta todos los servicios ofrecidos en la nube de AWS y de proporcionar servicios que puede utilizar de forma segura. Nuestra responsabilidad en torno a la seguridad es la mayor prioridad en AWS y auditores externos prueban y verifican la eficacia de nuestra seguridad con frecuencia como parte de los programas de conformidad de AWS.

Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que usa y por otros factores, como la confidencialidad de los datos, los requisitos de la organización, y las normas y los reglamentos aplicables.

Este producto o servicio de AWS sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la página de documentación sobre la seguridad de los servicios de AWS y los servicios de AWS sujetos a las medidas de conformidad de AWS de cada programa de conformidad.

Temas

- Protección de datos de este producto o servicio de AWS
- Identity and Access Management
- Validación de la conformidad de este producto o servicio de AWS
- Aplicación de una versión mínima de TLS en las Herramientas para PowerShell

Protección de datos de este producto o servicio de AWS

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos de este producto o servicio de AWS. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la AWS Cloud. 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 las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog Modelo de responsabilidad compartida y GDPR de AWS en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de la cuenta de Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se conceden a cada usuario los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. reSe recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los Servicios de AWS.
- Utilice servicios de seguridad gestionados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de la línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

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 trabaje con este producto o servicio de AWS u otros servicios de AWS a través de la consola, la API, AWS CLI o SDK de AWS. 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, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para 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

La AWS Tools for PowerShell no guarda ningún otro datos de los clientes salvo las credenciales necesarias para interactuar con los servicios de AWS en nombre del usuario.

Si utiliza la AWS Tools for PowerShell para invocar un servicio de AWS que transmita datos del cliente al equipo local para su almacenamiento, consulte el capítulo «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 equipo cliente en el que se ejecutan los puntos de enlace de servicio de AWS Tools for PowerShell y AWS están cifrados, ya que todo se envía a través de una conexión HTTPS/TLS.

No es necesario hacer nada para habilitar el uso de HTTPS/TLS. Siempre está habilitado.

Identity and Access Management

AWS Identity and Access Management (IAM) es un servicio de AWS que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de AWS. IAM es un servicio de AWS que se puede utilizar sin cargo adicional.

Temas

Público

La forma en que utilice AWS Identity and Access Management (IAM) difiere en función del trabajo que realice en AWS.

Usuario de servicio: si utiliza servicios de AWS para realizar el trabajo, el administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de AWS 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 característica en AWS, consulte Solución de problemas de identidades y accesos en AWS o la Guía del usuario del servicio de AWS que esté usando.

Administrador de servicio: si está a cargo de los recursos de AWS en su empresa, probablemente tenga acceso completo a AWS. Su trabajo consiste en determinar a qué características y recursos de AWS deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de sus servicios. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo la empresa puede utilizar IAM con AWS, consulte la Guía del usuario del servicio de AWS que esté usando.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS. Para consultar ejemplos de políticas basadas en la identidad de AWS que puede utilizar en IAM, consulte la Guía del usuario del servicio de AWS que esté usando.

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado (haber iniciado sesión en AWS) como el Usuario raíz de la cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión en AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad de AWS IAM Identity Center. Los usuarios (del Centro de identidades de IAM), 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, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accede a AWS mediante la federación, está asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en la AWS Management Console o en el portal de acceso a AWS. Para obtener más información sobre el inicio de sesión en AWS, consulte Cómo iniciar sesión en su Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Si accede a AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de la línea de comandos (CLI) para firmar criptográficamente las solicitudes mediante el uso de las credenciales. Si no usa las herramientas de AWS, debe firmar usted mismo las solicitudes. Para obtener más información sobre la firma de solicitudes, consulte Firma de solicitudes API de AWS en la Guía del usuario de IAM.

Independientemente del método de autenticación que utilice, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, AWS le recomienda el uso de la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.

Usuario raíz de Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión que tiene acceso completo a todos los recursos y Servicios de AWS de la cuenta. Esta identidad recibe el nombre de usuario raíz de la Cuenta de AWS y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizó 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 las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, solicite que los usuarios humanos, incluidos los que requieren acceso de administrador, utilicen la federación con un proveedor de identidades para acceder a los servicios de Servicios de AWS utilizando credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidad web, el AWS Directory Service, el directorio del Identity Center, o cualquier usuario que acceda a Servicios de AWS utilizando credenciales proporcionadas a través de una fuente de identidad. Cuando identidades federadas acceden a las Cuentas de AWS, asumen roles y los roles proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el IAM Identity Center o puede conectarse y sincronizar con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus

aplicaciones y Cuentas de AWS. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad en su Cuenta de AWS que dispone de permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del Usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. 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 de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

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 permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del Usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad en su Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente un rol de IAM en la AWS Management Consolecambiando de roles. Puede asumir un rol llamando a una operación de la AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del Usuario de IAM.

Los roles de IAM 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 acerca de roles para

federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del Usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. El IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center.

- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. No obstante, con algunos Servicios de AWS se puede asociar una política directamente a un recurso (en lugar de utilizar un rol como representante). Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan características de otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común 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 a servicios.
  - Reenviar sesiones de acceso (FAS): cuando utiliza un rol o un usuario de IAM para llevar a cabo acciones en AWS, se le considera una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS o recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Rol vinculado a servicios: un rol vinculado a servicios es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su

nombre. Los roles vinculados a servicios aparecen en su Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia de EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar un rol de AWS a una instancia de EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia asociado a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para obtener más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias de Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del Usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando una entidad principal (sesión de rol, usuario o usuario raíz) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de las políticas JSON en la Guía del Usuario de IAM.

Los administradores pueden utilizar las políticas JSON de AWS 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 permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador de IAM puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen 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 dicha política puede obtener información del usuario de la AWS Management Console, la AWS CLI o la API de AWS.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política en función de identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidad 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 asociar a varios usuarios, grupos y roles de su Cuenta de AWS. Las políticas administradas incluyen las políticas administradas de AWS y las políticas administradas por el cliente. Para obtener más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM 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. Las entidades 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 se puede utilizar políticas de IAM administradas por AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de política JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para Desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. 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 característica avanzada que le permite establecer los permisos máximos que una política en función de identidad puede conceder a una entidad de IAM (usuario o rol de IAM). 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 Límites de permisos para las entidades de IAM en la Guía del Usuario de IAM.
- Políticas de control de servicio (SCP): las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias Cuentas de AWS que posea su empresa. Si habilita todas las características en una empresa, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Una SCP limita los permisos para las entidades de las cuentas de miembros, incluido cada rootlong. Para más información sobre organizaciones y las SCP, consulte Funcionamiento de las SCP en la Guía del Usuario de AWS Organizations.
- 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 identidad 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 más información, consulte Políticas de sesión en la Guía del Usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si permitir o no una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funcionan los servicios de AWS con IAM

Para obtener una perspectiva general sobre cómo funcionan los servicios de AWS con la mayoría de las características de IAM, consulte Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Para obtener información sobre cómo usar un servicio de AWS específico con IAM, consulte la sección de seguridad de la Guía del usuario del servicio correspondiente.

Solución de problemas de identidades y accesos en AWS

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con AWS e IAM.

Temas

- No tengo autorización para realizar una acción en AWS
- No tengo autorización para realizar la operación iam:PassRole
- Quiero permitir a personas externas a mi cuenta de AWS el acceso a mis recursos de AWS

No tengo autorización para realizar una 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.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `awes:`*`GetWidget`*.

```
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 su administrador de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No tengo autorización para realizar la operación iam: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.

Algunos servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en 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 su administrador de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir a personas externas a mi cuenta de AWS el acceso a mis recursos de AWS

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 admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para obtener información acerca de si AWS admite estas características, consulte Cómo funcionan los servicios de AWS con IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las cuentas de AWS de su propiedad, consulte Proporcionar acceso a un usuario de IAM a otra cuenta de AWS de la que es propietario en la Guía del usuario de IAM.

- Para obtener información sobre cómo proporcionar acceso a los recursos a Cuentas de AWS de terceros, consulte Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (federación de identidades) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del Usuario de IAM.

Validación de la conformidad de este producto o servicio de AWS

Para saber si un servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte Servicios de AWS en el ámbito del programa de conformidad y escoja el programa de conformidad que le interese. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la sensibilidad de los datos, los objetivos de cumplimiento de su empresa y la legislación y los reglamentos correspondientes. AWS proporciona los siguientes recursos para ayudar con la conformidad:

- Guías de inicio rápido de seguridad y conformidad: estas guías de implementación tratan consideraciones sobre arquitectura y ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Arquitectura para la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones aptas para HIPAA.

> **Note**
>
> No todos los Servicios de AWS son aptos para HIPAA. Para obtener más información, consulte la Referencia de servicios aptos para HIPAA.

- Recursos de conformidad de AWS: este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- Guías de cumplimiento para clientes de AWS: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad de los Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés), el Consejo de Estándares de Seguridad de la Industria de Tarjetas de Pago (PCI, por sus siglas en inglés) y la Organización Internacional de Normalización (ISO, por sus siglas en inglés)).
- Evaluación de recursos con reglas en la Guía para desarrolladores de AWS Config: el servicio AWS Config evalúa en qué medida las configuraciones de sus recursos cumplen las prácticas internas, las directrices del sector y las normativas.
- AWS Security Hub: este servicio de Servicio de AWS proporciona una visión completa de su estado de seguridad en 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.
- AWS Audit Manager: este servicio de AWS le ayuda a auditar continuamente el uso de AWS con el fin de simplificar la forma en que administra el riesgo y la conformidad con las normativas y los estándares del sector.

Este producto o servicio de AWS sigue el modelo de responsabilidad compartida a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la página de documentación sobre la seguridad de los servicios de AWS y los servicios de AWS sujetos a las medidas de conformidad de AWS de cada 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.

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
Configure la autenticación de herramientas con AWS	Se agregó información sobre la compatibilidad con el SSO en. 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 incluyen más actualizaciones de prácticas recomendadas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulte prácticas recomendadas 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 el MSI, que ha quedado obsoleto.	25 de septiembre de 2023
Actualizaciones de las prácticas recomendadas de IAM	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para	8 de septiembre de 2023

	obtener más información, consulte prácticas recomendadas de seguridad en IAM.	
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 instancia de Amazon EC2 mediante Windows PowerShell	Se han agregado 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 ha añadido información a la sección Introducción sobre los nuevos cmdlets que utilizan SAML (Security Assertion Markup Language) para permitir la configuración de identidades federadas para 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-S3Bucket admite una opción de cambio. DeleteObjects 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-EC2Instance acepta instancias Amazon.EC2.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-S3Object permite cargar contenido de texto insertado en 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.