Guia do usuário

AWS Tools for PowerShell

AWS Tools for PowerShell: Guia do usuário

Table of Contents

O que é o AWS Tools for PowerShell?

O AWS Tools for PowerShell é um conjunto de módulos do PowerShell criados com base na funcionalidade exposta pelo AWS SDK for .NET. O AWS Tools for PowerShell permite que você faça script de operações em seus recursos da AWS a partir da linha de comando do PowerShell.

Os cmdlets fornecem uma experiência idiomática do PowerShell para especificar parâmetros e lidar com os resultados, mesmo que eles sejam implementados usando as várias APIs de consulta HTTP do produto da AWS. Por exemplo, os cmdlets para o AWS Tools for PowerShell oferecem suporte ao pipeline do PowerShell ou seja, você pode estruturar objetos do PowerShell nos cmdlets e fora deles.

O AWS Tools for PowerShell é flexível na forma como permite a você manipular credenciais, incluindo suporte à infraestrutura do AWS Identity and Access Management (IAM). É possível usar as ferramentas com credenciais de usuário do IAM, tokens de segurança temporários e funções do IAM.

O AWS Tools for PowerShell oferece suporte ao mesmo conjunto de serviços e regiões da AWS compatíveis com o SDK. É possível instalar o AWS Tools for PowerShell em computadores com sistemas operacionais Windows, Linux ou macOS.

> **Note**
>
> O AWS Tools for PowerShell versão 4 é a versão principal mais recente e é uma atualização compatível com versões anteriores do AWS Tools for PowerShell versão 3.3. Ele adiciona melhorias significativas enquanto mantém o comportamento existente do cmdlet. Os scripts existentes devem continuar a funcionar após a atualização para a nova versão, mas recomendamos que você os teste minuciosamente antes de atualizar. Para obter mais informações sobre as alterações na versão 4, consulte Migrar do AWS Tools for PowerShell versão 3.3 para a versão 4.

O AWS Tools for PowerShell está disponível como os três pacotes distintos a seguir:

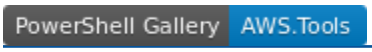
- `AWS.Tools`
- AWSPowerShell.NetCore
- AWSPowerShell

Manutenção e suporte para as versões principais de SDK

Para obter informações sobre manutenção e suporte para versões principais do SDK e suas dependências subjacentes, consulte o seguinte no Guia de referência de AWS SDKs e ferramentas:

- Política de manutenção de ferramentas e SDKs da AWS
- Matriz de suporte a versões de ferramentas e SDKs da AWS

`AWS.Tools`: uma versão modularizada do AWS Tools for PowerShell



Esta versão do AWS Tools for PowerShell é a versão recomendada para qualquer computador executando o PowerShell em um ambiente de produção. Como ele é modularizado, é necessário fazer download e carregar somente os módulos dos serviços que deseja usar. Isso reduz o tempo de download, o uso de memória e, na maioria dos casos, habilita a importação automática de cmdlets das `AWS.Tools` sem a necessidade de chamar `Import-Module` manualmente primeiro.

Esta é a versão mais recente do AWS Tools for PowerShell e é executada em todos os sistemas operacionais com suporte, incluindo Windows, Linux e macOS. Esse pacote fornece um módulo de instalação, `AWS.Tools.Installer`, um módulo comum, `AWS.Tools.Common` e um módulo para cada serviço da AWS, por exemplo, `AWS.Tools.EC2`, `AWS.Tools.IAM`, `AWS.Tools.S3` e assim por diante.

O módulo `AWS.Tools.Installer` fornece cmdlets que permitem instalar, atualizar e remover os módulos para cada um dos serviços da AWS. Os cmdlets desse módulo garantem automaticamente que você tenha todos os módulos dependentes necessários para oferecer suporte aos módulos que você deseja usar.

O módulo `AWS.Tools.Common` fornece cmdlets para configuração e autenticação que não são específicos do produto. Para usar os cmdlets para um produto da AWS, basta executar o comando. O PowerShell importa automaticamente o módulo `AWS.Tools.Common` e o módulo para o produto da AWS cujo cmdlet você deseja executar. Esse módulo é instalado automaticamente se você usar o módulo `AWS.Tools.Installer` para instalar os módulos do serviço.

É possível instalar esta versão do AWS Tools for PowerShell em computadores que estejam executando:

- PowerShell Core 6.0 ou posterior no Windows, Linux ou macOS.
- Windows PowerShell 5.1 ou posterior no Windows com o .NET Framework 4.7.2 ou posterior.

Ao longo deste guia, quando precisamos especificar somente esta versão, nos referimos a ela pelo nome do módulo: *`AWS.Tools`*.

AWSPowerShell.NetCore: uma versão de módulo único do AWS Tools for PowerShell

Ela consiste em um único módulo grande que contém suporte a todos os produtos da AWS. Antes de usar este módulo, você deverá importá-lo manualmente.

É possível instalar esta versão do AWS Tools for PowerShell em computadores que estejam executando:

- PowerShell Core 6.0 ou posterior no Windows, Linux ou macOS.
- Windows PowerShell 3.0 ou posterior no Windows com o .NET Framework 4.7.2 ou posterior.

Ao longo deste guia, sempre que precisarmos especificar somente esta versão, faremos referência a ela pelo nome do módulo: AWSPowerShell.NetCore.

AWSPowerShell: uma versão de módulo único para o Windows PowerShell

Esta versão do AWS Tools for PowerShell é compatível com e pode ser instalada somente em computadores Windows que executam as versões 2.0 a 5.1 do Windows PowerShell. Ela não é

compatível com o PowerShell Core 6.0 ou posterior, ou qualquer outro sistema operacional (Linux ou macOS). Ela consiste em um único módulo grande que contém suporte a todos os produtos da AWS.

Ao longo deste guia, sempre que precisarmos especificar somente esta versão, faremos referência a ela pelo nome do módulo: AWSPowerShell.

Como usar este guia

O guia é dividido nas seções principais a seguir.

Instalar o AWS Tools for PowerShell

Esta seção explica como instalar o AWS Tools for PowerShell. Ele inclui informações sobre como se cadastrar na AWS se você ainda não tiver uma conta e como criar um usuário do IAM que pode ser usado para executar os cmdlets.

Conceitos básicos da AWS Tools for Windows PowerShell

Esta seção descreve os conceitos básicos do uso do AWS Tools for PowerShell, como especificar credenciais e regiões da AWS, encontrar cmdlets para um determinado produto e o uso de aliases para cmdlets.

Trabalhar com serviços da AWS no AWS Tools for PowerShell

Esta seção inclui informações sobre como usar o AWS Tools for PowerShell para executar algumas das tarefas mais comuns da AWS.

Instalar o AWS Tools for PowerShell

Para instalar e usar os cmdlets do AWS Tools for PowerShell com êxito, consulte as etapas nos tópicos a seguir.

Tópicos

Instalando o AWS Tools for PowerShell no Windows

Um computador baseado em Windows pode executar qualquer uma das opções de pacote: AWS Tools for PowerShell

- **`AWS.Tools`**- A versão modularizada do. AWS Tools for PowerShell Cada AWS serviço é suportado por seu próprio módulo pequeno e individual, com módulos de suporte compartilhados `AWS.Tools.Common` `AWS.Tools.Installer` e.
- AWSPowerShell. NetCore- A versão única e de módulo grande do. AWS Tools for PowerShell Todos os AWS serviços são suportados por esse módulo único e grande.

  > **Note**
  >
  > Esteja ciente de que o módulo único pode ser muito grande para ser usado com funções AWS Lambda. Em vez disso, use a versão modularizada mostrada acima.

- AWSPowerShell – a versão única de módulo grande legada específica do Windows do AWS Tools for PowerShell. Todos os AWS serviços são suportados por esse módulo único e grande.

O pacote escolhido depende da versão e edição do Windows sendo executado.

> **Note**
>
> As Ferramentas para Windows PowerShell (AWSPowerShell módulo) são instaladas por padrão em todas as Amazon Machine Images (AMIs) baseadas em Windows.

A configuração do AWS Tools for PowerShell envolve as seguintes tarefas de alto nível, descritas em detalhes neste tópico.

1. Instale a opção de AWS Tools for PowerShell pacote apropriada para seu ambiente.
2. Verifique se a execução do script está habilitada ao executar o cmdlet `Get-ExecutionPolicy`.
3. Importe o AWS Tools for PowerShell módulo para sua PowerShell sessão.

Pré-requisitos

Versões mais recentes do PowerShell, incluindo o PowerShell Core, estão disponíveis como downloads na Microsoft em Instalando várias versões do PowerShell no site da Microsoft.

Instalar o **`AWS.Tools`** no Windows

Você pode instalar a versão modularizada do AWS Tools for PowerShell em computadores que executam o Windows com o Windows PowerShell 5.1 ou PowerShell Core 6.0 ou posterior. Para obter informações sobre como instalar o PowerShell Core, consulte Instalando várias versões do PowerShell no site da Microsoft.

Você pode instalar o `AWS.Tools` de três maneiras:

- Usando os cmdlets do módulo `AWS.Tools.Installer`. Esse módulo simplifica a instalação e a atualização de outros `AWS.Tools` módulos. `AWS.Tools.Installer`requer `PowerShellGet` e baixa e instala automaticamente uma versão atualizada do mesmo. `AWS.Tools.Installer`mantém automaticamente as versões do seu módulo sincronizadas. Quando você instala ou atualiza para uma versão mais recente de um módulo, os cmdlets atualizam `AWS.Tools.Installer` automaticamente todos os outros `AWS.Tools` módulos para a mesma versão.

  Esse método é descrito no procedimento a seguir.
- Baixando os módulos de `AWS.Tools.zip` e extração em uma das pastas do módulo. É possível descobrir as pastas de módulo exibindo o valor da variável de ambiente `PSModulePath`.
- Instalando cada módulo de serviço da PowerShell Galeria usando o `Install-Module` cmdlet.

Para instalar **`AWS.Tools`** no Windows usando o **`AWS.Tools.Installer`** módulo

1. Inicie uma PowerShell sessão.

> **Note**
>
> Recomendamos que você não execute PowerShell como administrador com permissões elevadas, exceto quando exigido pela tarefa em questão. Isso se deve ao risco potencial de segurança e é consistente com o princípio do privilégio mínimo.

2. Para instalar o pacote do `AWS.Tools` modularizado, execute o comando a seguir.

```
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
```

   Se for notificado de que o repositório é "não confiável", você será solicitado a confirmar se deseja instalar de qualquer maneira. Digite **y** para permitir PowerShell a instalação do módulo. Para evitar o prompt e instalar o módulo sem confiar no repositório, você pode executar o comando com o parâmetro `-Force`.

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

3. Agora você pode instalar o módulo para cada AWS serviço que você deseja usar usando o `Install-AWSToolsModule` cmdlet. Por exemplo, o comando a seguir instala os módulos do Amazon EC2 e do Amazon S3. Esse comando também instala todos os módulos dependentes que são exigidos para que o módulo especificado funcione. Por exemplo, ao instalar o primeiro módulo do serviço `AWS.Tools`, ele também instala `AWS.Tools.Common`. Esse é um módulo compartilhado exigido por todos os módulos AWS de serviço. Ele também remove as versões mais antigas dos módulos e atualiza outros módulos para a mesma versão mais recente.

```
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**
>
> O cmdlet `Install-AWSToolsModule` faz download de todos os módulos solicitados do `PSRepository` chamado `PSGallery` (https://www.powershellgallery.com/) e o considera como uma fonte confiável. Use o comando `Get-PSRepository -Name PSGallery` para obter mais informações sobre esse `PSRepository`.

Por padrão, o comando anterior instala módulos na pasta `%USERPROFILE%\Documents\WindowsPowerShell\Modules`. Para instalar o AWS Tools for PowerShell para todos os usuários de um computador, você deve executar o comando a seguir em uma PowerShell sessão iniciada como administrador. Por exemplo, o comando a seguir instala o módulo do IAM na pasta `%ProgramFiles%\WindowsPowerShell\Modules` que pode ser acessada por todos os usuários.

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

Para instalar outros módulos, execute comandos semelhantes com os nomes de módulo apropriados, conforme encontrado na PowerShell Galeria.

Instalar AWSPowerShell. NetCore no Windows

Você pode instalar AWSPowerShell o. NetCore em computadores que executam o Windows com a PowerShell versão 3 a 5.1 ou PowerShell Core 6.0 ou posterior. Para obter informações sobre como instalar o PowerShell Core, consulte Instalando várias versões do PowerShell no PowerShell site da Microsoft.

Você pode instalar AWSPowerShell. NetCore em uma das duas maneiras

- Baixando o módulo de AWSPowerShell. NetCore.zip e extraindo-o em um dos diretórios do módulo. É possível descobrir os diretórios do módulo exibindo o valor da variável de ambiente `PSModulePath`.
- Instalando a partir da PowerShell Galeria usando o `Install-Module` cmdlet, conforme descrito no procedimento a seguir.

Para instalar AWSPowerShell. NetCore da PowerShell Galeria usando o cmdlet Install-Module

Para instalar AWSPowerShell o. NetCore da PowerShell Galeria, seu computador deve estar executando a PowerShell versão 5.0 ou posterior ou a versão PowerShell 3 ou posterior. PowerShellGet Execute o seguinte comando .

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

Se você estiver executando PowerShell como administrador, o comando anterior será instalado AWS Tools for PowerShell para todos os usuários no computador. Se você estiver executando PowerShell como um usuário padrão sem permissões de administrador, esse mesmo comando será instalado somente AWS Tools for PowerShell para o usuário atual.

Para instalar apenas para o usuário atual quando esse usuário tiver permissões de administrador, execute o comando com o conjunto de parâmetros do `-Scope CurrentUser` conforme indicado a seguir.

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

Embora as versões PowerShell 3.0 e posteriores normalmente carreguem módulos em sua PowerShell sessão na primeira vez que você executa um cmdlet no módulo, a. AWSPowerShell NetCore o módulo é muito grande para suportar essa funcionalidade. Em vez disso, você deve

carregar explicitamente o. AWSPowerShell NetCore Módulo principal em sua PowerShell sessão executando o comando a seguir.

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

Para carregar AWSPowerShell o. NetCore módulo em uma PowerShell sessão automaticamente, adicione esse comando ao seu PowerShell perfil. Para obter mais informações sobre como editar seu PowerShell perfil, consulte Sobre perfis na PowerShell documentação.

Instalar AWSPowerShell no Windows PowerShell

Você pode instalar o AWS Tools for Windows PowerShell de duas maneiras:

- Baixando o módulo do AWSPowerShellarquivo.zip e extraindo-o em um dos diretórios do módulo. É possível descobrir os diretórios do módulo exibindo o valor da variável de ambiente `PSModulePath`.
- Instalando a partir da PowerShell Galeria usando o `Install-Module` cmdlet conforme descrito no procedimento a seguir.

Para instalar a AWSPowerShell partir da PowerShell Galeria usando o cmdlet Install-Module

Você pode instalar o a AWSPowerShell partir da PowerShell Galeria se estiver executando a PowerShell versão 5.0 ou posterior ou se tiver PowerShellGetinstalado a versão PowerShell 3 ou posterior. Você pode instalar e atualizar AWSPowerShell a partir da PowerShellGaleria da Microsoft executando o seguinte comando.

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

Para carregar o AWSPowerShell módulo em uma PowerShell sessão automaticamente, adicione o `import-module` cmdlet anterior ao seu PowerShell perfil. Para obter mais informações sobre como editar seu PowerShell perfil, consulte Sobre perfis na PowerShell documentação.

> **Note**
>
> As Ferramentas para Windows PowerShell são instaladas por padrão em todas as Amazon Machine Images (AMIs) baseadas em Windows.

Ativar a execução do script

Para carregar os AWS Tools for PowerShell módulos, você deve habilitar a execução do PowerShell script. Para habilitar a execução do script, execute o cmdlet `Set-ExecutionPolicy` para definir uma política de `RemoteSigned`. Para obter mais informações, consulte About Execution Policies no site da Microsoft Technet.

> **Note**
>
> Este é um requisito apenas para computadores que executam o Windows. A restrição de segurança `ExecutionPolicy` não está presente em outros sistemas operacionais.

Para ativar a execução do script

1. São necessários direitos de administrador para definir a política de execução. Se você não estiver logado como usuário com direitos de administrador, abra uma PowerShell sessão como administrador. Escolha Start (Iniciar) e selecione All Programs (Todos os programas). Escolha Acessórios e, em seguida, escolha Windows PowerShell. Clique com o botão direito do mouse em Windows e PowerShell, no menu de contexto, escolha Executar como administrador.
2. No prompt de comando, digite o seguinte.

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

> **Note**
>
> Em um sistema de 64 bits, você deve fazer isso separadamente para a versão de 32 bits do PowerShell Windows PowerShell (x86).

Se você não tiver a política de execução definida corretamente, PowerShell mostrará o erro a seguir sempre que você tentar executar um script, como seu 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
```

O PowerShell instalador do Tools for Windows atualiza automaticamente o PS ModulePath para incluir a localização do diretório que contém o `AWSPowerShell` módulo.

Como `PSModulePath` inclui a localização do diretório do AWS módulo, o `Get-Module -ListAvailable` cmdlet mostra o módulo.

```
PS > Get-Module -ListAvailable

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

Versionamento

AWS lança novas versões do AWS Tools for PowerShell periodicamente para oferecer suporte a novos AWS serviços e recursos. Para determinar a versão das Ferramentas que você instalou, execute o 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]
```

Você também pode adicionar o `-ListServiceVersionInfo` parâmetro a um AWSPowerShellVersion comando Get- para ver uma lista dos AWS serviços que são suportados na versão atual das ferramentas. Se você usar a opção modularizada `AWS.Tools.*`, somente os módulos importados atualmente serão exibidos.

```
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 a versão PowerShell que você está executando, insira `$PSVersionTable` para visualizar o conteúdo da variável 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
```

Atualizando o AWS Tools for PowerShell no Windows

Periodicamente, à medida que as versões atualizadas do AWS Tools for PowerShell são lançadas, você deve atualizar a versão que está executando localmente.

Atualize os módulos modularizados **`AWS.Tools`**

Para atualizar seus `AWS.Tools` módulos para a versão mais recente, execute o seguinte comando:

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

Esse comando atualiza todos os `AWS.Tools` módulos instalados no momento e, após uma atualização bem-sucedida, remove outras versões instaladas.

> **Note**
>
> O cmdlet `Update-AWSToolsModule` faz download de todos os módulos do `PSRepository` chamado `PSGallery` (https://www.powershellgallery.com/) e o considera como uma fonte confiável. Use o comando: `Get-PSRepository -Name PSGallery` para obter mais informações sobre esse `PSRepository`.

Atualize as ferramentas do PowerShell Core

Execute o `Get-AWSPowerShellVersion` cmdlet para determinar a versão que você está executando e compare-a com a versão do Tools for Windows PowerShell que está disponível no site da PowerShell Galeria. Sugerimos verificar a cada duas ou três semanas. Support para novos comandos e AWS serviços está disponível somente após a atualização para uma versão com esse suporte.

Antes de instalar uma versão mais recente do. AWSPowerShell NetCore, desinstale o módulo existente. Feche todas PowerShell as sessões abertas antes de desinstalar o pacote existente. Execute o seguinte comando para desinstalar o pacote.

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

Depois de desinstalar o pacote, instale o módulo atualizado executando o comando a seguir.

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

Após a instalação, execute o comando `Import-Module AWSPowerShell.NetCore` para carregar os cmdlets atualizados em sua PowerShell sessão.

Atualize as ferramentas para Windows PowerShell

Execute o `Get-AWSPowerShellVersion` cmdlet para determinar a versão que você está executando e compare-a com a versão do Tools for Windows PowerShell que está disponível no site da PowerShell Galeria. Sugerimos verificar a cada duas ou três semanas. Support para novos comandos e AWS serviços está disponível somente após a atualização para uma versão com esse suporte.

- Se você instalou usando o cmdlet `Install-Module`, execute os comandos a seguir.

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

- Se você instalou usando um arquivo ZIP baixado:

  1. Baixe a versão mais recente do PowerShell site Tools for. Compare o número da versão do pacote no nome do arquivo baixado com o número da versão obtido ao executar o cmdlet `Get-AWSPowerShellVersion`.

2. Se a versão de download for maior do que a versão que você instalou, feche todos os PowerShell consoles do Tools for Windows.
3. Instale a versão mais recente do Tools for Windows PowerShell.

Após a instalação, execute `Import-Module AWSPowerShell` para carregar os cmdlets atualizados em sua PowerShell sessão. Ou execute o AWS Tools for PowerShell console personalizado no menu Iniciar.

Instalando AWS Tools for PowerShell no Linux ou macOS

Este tópico fornece instruções sobre como instalar o AWS Tools for PowerShell no Linux ou no macOS.

Visão geral da configuração

Para instalar AWS Tools for PowerShell em um computador Linux ou macOS, você pode escolher entre duas opções de pacote:

- **`AWS.Tools`**— A versão modularizada do. AWS Tools for PowerShell Cada AWS serviço é suportado por seu próprio módulo pequeno e individual, com módulos de suporte compartilhadosAWS.Tools.Common.
- AWSPowerShell. NetCore— A versão única e de módulo grande do. AWS Tools for PowerShell Todos os AWS serviços são suportados por esse módulo único e grande.

> **Note**
>
> Esteja ciente de que o módulo único pode ser muito grande para ser usado com funções AWS Lambda. Em vez disso, use a versão modularizada mostrada acima.

A respectiva configuração em um computador com Linux ou macOS envolve as seguintes tarefas, descritas em detalhes posteriormente nesse tópico:

1. Instale o PowerShell Core 6.0 ou posterior em um sistema compatível.
2. Depois de instalar o PowerShell Core, comece PowerShell executando `pwsh` no shell do sistema.
3. Instale um `AWS.Tools` ou AWSPowerShell. NetCore.

4. Execute o `Import-Module` cmdlet apropriado para importar o módulo para sua PowerShell sessão.
5. Execute o AWSDefaultConfiguration cmdlet Initialize- para fornecer suas credenciais. AWS

Pré-requisitos

Para executar o AWS Tools for PowerShell Core, seu computador deve estar executando o PowerShell Core 6.0 ou posterior.

- Para obter uma lista das versões suportadas da plataforma Linux e obter informações sobre como instalar a versão mais recente do PowerShell em um computador baseado em Linux, consulte Instalando PowerShell no Linux no site da Microsoft. Alguns sistemas operacionais baseados em Linux, como Arch, Kali e Raspbian, não são oficialmente compatíveis, mas têm níveis variáveis de suporte da comunidade.
- Para obter informações sobre as versões compatíveis do macOS e sobre como instalar a versão mais recente do no PowerShell macOS, consulte Instalação no PowerShell macOS no site da Microsoft.

Instalar o **`AWS.Tools`** no Linux ou no macOS

Você pode instalar a versão modularizada do AWS Tools for PowerShell em computadores que estejam executando o PowerShell Core 6.0 ou posterior. Para obter informações sobre como instalar o PowerShell Core, consulte Instalando várias versões do PowerShell no PowerShell site da Microsoft.

Você pode instalar o `AWS.Tools` de três maneiras:

- Usando os cmdlets do módulo `AWS.Tools.Installer`. Esse módulo simplifica a instalação e a atualização de outros `AWS.Tools` módulos. `AWS.Tools.Installer`requer `PowerShellGet` e baixa e instala automaticamente uma versão atualizada do mesmo. `AWS.Tools.Installer`mantém automaticamente as versões do seu módulo sincronizadas. Quando você instala ou atualiza para uma versão mais recente de um módulo, os cmdlets atualizam `AWS.Tools.Installer` automaticamente todos os outros `AWS.Tools` módulos para a mesma versão.

  Esse método é descrito no procedimento a seguir.

- Download dos módulos de AWS.Tools.zip e extração em um dos diretórios do módulo. Você pode descobrir seus diretórios de módulo imprimindo o valor da variável $Env:PSModulePath.
- Instalando cada módulo de serviço da PowerShell Galeria usando o Install-Module cmdlet.

Para instalar **AWS.Tools** no Linux ou no macOS usando o módulo **AWS.Tools.Installer**

1. Inicie uma sessão PowerShell principal executando o comando a seguir.

```
$ pwsh
```

> **Note**
>
> Recomendamos que você não execute PowerShell como administrador com permissões elevadas, exceto quando exigido pela tarefa em questão. Isso se deve ao risco potencial de segurança e é consistente com o princípio do privilégio mínimo.

2. Para instalar o pacote AWS.Tools modularizado usando o módulo AWS.Tools.Installer, execute o comando a seguir.

```
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
```

Se você for notificado de que o repositório é "não confiável", você será solicitado a confirmar se deseja prosseguir com a instalação. Digite **y** para permitir PowerShell a instalação do módulo. Para evitar o prompt e instalar o módulo sem confiar no repositório, você pode executar o comando a seguir.

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

3. Agora você pode instalar o módulo para cada serviço que deseja usar. Por exemplo, o comando a seguir instala os módulos do Amazon EC2 e do Amazon S3. Esse comando também instala

todos os módulos dependentes que são exigidos para que o módulo especificado funcione. Por exemplo, ao instalar o primeiro módulo do serviço `AWS.Tools`, ele também instala `AWS.Tools.Common`. Esse é um módulo compartilhado exigido por todos os módulos AWS de serviço. Ele também remove as versões mais antigas dos módulos e atualiza outros módulos para a mesma versão mais recente.

```
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**
>
> O cmdlet `Install-AWSToolsModule` faz download de todos os módulos solicitados do `PSRepository` chamado `PSGallery` (https://www.powershellgallery.com/) e considera o repositório como uma fonte confiável. Use o comando `Get-PSRepository -Name PSGallery` para obter mais informações sobre esse `PSRepository`.

O comando anterior instala os módulos nos diretórios-padrão do seu sistema. Os diretórios reais dependem da distribuição e da versão do sistema operacional e da PowerShell versão instalada. Por exemplo, se você instalou o PowerShell 7 em um sistema semelhante ao RHEL, os módulos padrão provavelmente estão localizados em `/opt/microsoft/powershell/7/Modules` (ou`$PSHOME/Modules`) e os módulos de usuário provavelmente estão localizados

em. `~/.local/share/powershell/Modules` Para obter mais informações, consulte Instalar PowerShell no Linux no PowerShell site da Microsoft. Para ver onde os módulos estão instalados, execute o seguinte comando:

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

Para instalar outros módulos, execute comandos semelhantes com os nomes de módulo apropriados, conforme encontrado na PowerShell Galeria.

Instalar AWSPowerShell. NetCore no Linux ou macOS

Para atualizar para uma versão mais recente do. AWSPowerShell NetCore, siga as instruções emAtualizando o AWS Tools for PowerShell no Linux ou no macOS. Desinstale as versões anteriores do AWSPowerShell. NetCore primeiro.

Você pode instalar AWSPowerShell. NetCore em uma das duas formas:

- Download do módulo de `AWSPowerShell.NetCore.zip` e extração em um dos diretórios do módulo. Você pode descobrir seus diretórios de módulo imprimindo o valor da variável `$Env:PSModulePath`.
- Instalando a partir da PowerShell Galeria usando o `Install-Module` cmdlet conforme descrito no procedimento a seguir.

Para instalar AWSPowerShell. NetCore no Linux ou macOS usando o cmdlet Install-Module

Inicie uma sessão PowerShell principal executando o comando a seguir.

```
$ pwsh
```

> **Note**
>
> Recomendamos que você não comece PowerShell correndo `sudo pwsh` para executar PowerShell com direitos de administrador elevados. Isso se deve ao risco potencial de segurança e é consistente com o princípio do privilégio mínimo.

Para instalar AWSPowerShell o. NetCore pacote de módulo único da PowerShell Galeria, execute o seguinte 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
```

Se você for notificado de que o repositório é "não confiável", você será solicitado a confirmar se deseja prosseguir com a instalação. Digite **y** para permitir PowerShell a instalação do módulo. Para evitar o prompt sem confiar no repositório, você pode executar o comando a seguir.

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

Você não precisa executar esse comando como root, a menos que queira instalá-lo AWS Tools for PowerShell para todos os usuários de um computador. Para fazer isso, execute o comando a seguir em uma PowerShell sessão com a qual você começou`sudo pwsh`.

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

Execução do script

O comando `Set-ExecutionPolicy` não está disponível em sistemas que não sejam Windows. Você pode executar`Get-ExecutionPolicy`, o que mostra que a configuração padrão da política de execução no PowerShell Core em execução em sistemas não Windows é`Unrestricted`. Para obter mais informações, consulte About Execution Policies no site da Microsoft Technet.

Como `PSModulePath` inclui a localização do diretório do AWS módulo, o `Get-Module -ListAvailable` cmdlet mostra o módulo que você instalou.

`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 um PowerShell console para usar o AWS Tools for PowerShell Core (AWSPowerShell. NetCore Somente)

PowerShell O Core normalmente carrega módulos automaticamente sempre que você executa um cmdlet no módulo. Mas isso não funciona para AWSPowerShell. NetCore por causa de seu grande tamanho. Para começar a correr AWSPowerShell. NetCore cmdlets, você deve primeiro executar o `Import-Module AWSPowerShell.NetCore` comando. Isso não é necessário para cmdlets nos módulos do `AWS.Tools`.

Inicialize sua sessão PowerShell

Ao iniciar PowerShell em um sistema baseado em Linux ou macOS depois de instalar o AWS Tools for PowerShell, você deve executar Initialize- AWSDefaultConfiguration para especificar qual chave de acesso usar. AWS Para obter mais informações sobre o `Initialize-AWSDefaultConfiguration`, consulte Usar credenciais da AWS.

> **Note**
>
> Nas versões anteriores (antes da 3.3.96.0) do AWS Tools for PowerShell, esse cmdlet foi nomeado. `Initialize-AWSDefaults`

Versionamento

AWS lança novas versões do AWS Tools for PowerShell periodicamente para oferecer suporte a novos AWS serviços e recursos. Para determinar a versão do AWS Tools for PowerShell que você instalou, execute o 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 uma lista dos AWS serviços suportados na versão atual das ferramentas, adicione o `-ListServiceVersionInfo` parâmetro a um AWSPowerShellVersion cmdlet Get-.

Para determinar a versão PowerShell que você está executando, insira `$PSVersionTable` para visualizar o conteúdo da variável `$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
```

Atualizando o AWS Tools for PowerShell no Linux ou no macOS

Periodicamente, à medida que as versões atualizadas do AWS Tools for PowerShell são lançadas, você deve atualizar a versão que está executando localmente.

Atualize os módulos modularizados **`AWS.Tools`**

Para atualizar seus `AWS.Tools` módulos para a versão mais recente, execute o seguinte comando:

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

Esse comando atualiza todos os módulos `AWS.Tools` instalados atualmente e, para os módulos que foram atualizados com êxito, remove as versões anteriores.

> **Note**
>
> O cmdlet `Update-AWSToolsModule` faz download de todos os módulos do `PSRepository` chamado `PSGallery` (https://www.powershellgallery.com/) e o considera como uma fonte confiável. Use o comando `Get-PSRepository -Name PSGallery` para obter mais informações sobre esse `PSRepository`.

Atualize as ferramentas do PowerShell Core

Execute o `Get-AWSPowerShellVersion` cmdlet para determinar a versão que você está executando e compare-a com a versão do Tools for Windows PowerShell que está disponível no site da PowerShell Galeria. Sugerimos verificar a cada duas ou três semanas. Support para novos comandos e AWS serviços está disponível somente após a atualização para uma versão com esse suporte.

Antes de instalar uma versão mais recente do. AWSPowerShell NetCore, desinstale o módulo existente. Feche todas PowerShell as sessões abertas antes de desinstalar o pacote existente. Execute o seguinte comando para desinstalar o pacote.

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

Depois de desinstalar o pacote, instale o módulo atualizado executando o comando a seguir.

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

Após a instalação, execute o comando `Import-Module AWSPowerShell.NetCore` para carregar os cmdlets atualizados em sua PowerShell sessão.

Informações relacionadas

- Conceitos básicos da AWS Tools for Windows PowerShell
- Trabalhar com serviços da AWS no AWS Tools for PowerShell

Migrar do AWS Tools for PowerShell versão 3.3 para a versão 4

O AWS Tools for PowerShell versão 4 é uma atualização compatível com versões anteriores para o AWS Tools for PowerShell versão 3.3. Ele adiciona melhorias significativas enquanto mantém o comportamento existente do cmdlet.

Os scripts existentes devem continuar a funcionar após a atualização para a nova versão, mas recomendamos que você os teste minuciosamente antes de atualizar os ambientes de produção.

Esta seção descreve as alterações e explica como elas podem impactar os scripts.

Nova versão totalmente modularizada do **`AWS.Tools`**

AWSPowerShellA. NetCore e AWSPowerShell os pacotes eram “monolíticos”. Isso significava que todos os serviços da AWS eram compatíveis no mesmo módulo, tornando-o muito grande e aumentando a cada vez que um novo recurso e serviço da AWS era adicionado. O novo pacote `AWS.Tools` está dividido em módulos menores que oferecem a flexibilidade de fazer download e instalar somente aqueles exigidos para os serviços da AWS que você usa. O pacote inclui um módulo `AWS.Tools.Common` compartilhado que é exigido por todos os outros módulos e um módulo `AWS.Tools.Installer` que simplifica a instalação, a atualização e a remoção de módulos conforme necessário.

Isso também permite importar os cmdlets automaticamente na primeira chamada, sem precisar chamar primeiro `Import-module`. No entanto, para interagir com os objetos.NET associados antes de chamar um cmdlet, você ainda deve ligar `Import-Module` para PowerShell informar sobre os tipos.NET relevantes.

Por exemplo, o comando a seguir tem uma referência a `Amazon.EC2.Model.Filter`. Esse tipo de referência não pode acionar a importação automática, portanto, é necessário chamar `Import-Module` primeiro ou o comando falhará.

```
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
```

Novo cmdlet `Get-AWSService`

Para ajudar a descobrir os nomes dos módulos para cada serviço da AWS na coleção de módulos do `AWS.Tools`, é possível usar o 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
  ...
```

Novo parâmetro `-Select` para controlar o objeto retornado por um cmdlet

A maioria dos cmdlets na versão 4 oferecem suporte a um novo parâmetro `-Select`. Cada cmdlet chama as APIs de serviço da AWS para você usando o AWS SDK for .NET. Em seguida, o AWS Tools for PowerShell cliente converte a resposta em um objeto que você pode usar em seus PowerShell scripts e direcionar para outros comandos. Às vezes, o PowerShell objeto final tem mais campos ou propriedades na resposta original do que você precisa, e outras vezes você pode querer que o objeto inclua campos ou propriedades da resposta que não estão lá por padrão. O parâmetro `-Select` permite que você especifique o que está incluído no objeto .NET retornado pelo cmdlet.

Por exemplo, o cmdlet Get-S3Object invoca a operação do Amazon S3 SDK. ListObjects Essa operação retorna um ListObjectsResponseobjeto. No entanto, por padrão, o `Get-S3Object` cmdlet

retorna somente o `S3Objects` elemento da resposta do SDK para o usuário. PowerShell No exemplo a seguir, esse objeto é uma matriz com dois 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
```

No AWS Tools for PowerShell versão 4, é possível especificar `-Select *` para retornar o objeto de resposta .NET completo retornado pela chamada de API do SDK.

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

Também é possível especificar o caminho para a propriedade aninhada específica que você deseja. O exemplo a seguir retorna somente a propriedade `Key` de cada elemento na matriz `S3Objects`.

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

Em determinadas situações, pode ser útil retornar um parâmetro de cmdlet. É possível fazer isso com `-Select ^ParameterName`. Esse recurso substitui o parâmetro `-PassThru`, que ainda está disponível, mas defasado.

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

O tópico de referência para cada cmdlet identifica se ele é compatível com o parâmetro `-Select`.

Limitação mais consistente do número de itens na saída

As versões anteriores do AWS Tools for PowerShell permitiam usar o parâmetro `-MaxItems` para especificar o número máximo de objetos retornados na saída final.

Esse comportamento foi removido do `AWS.Tools`.

Esse comportamento está obsoleto em. AWSPowerShell NetCore e AWSPowerShell, e serão removidos dessas versões em uma versão futura.

Se a API do serviço subjacente oferecer suporte a um parâmetro `MaxItems`, ele ainda estará disponível e funcionará conforme a API especifica. No entanto, ele não terá mais o comportamento adicionado de limitar o número de itens retornados na saída do cmdlet.

Para limitar o número de itens retornados na saída final, canalize a saída para o cmdlet `Select-Object` e especifique o parâmetro `-First` *n*, onde *n* é o número máximo de itens a serem incluídos na saída final.

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

Nem todos os serviços da AWS ofereciam suporte ao `-MaxItems` da mesma forma, portanto, isso remove essa inconsistência e os resultados inesperados que às vezes ocorriam. Além disso, o `-MaxItems` combinado com o novo parâmetro `-Select` poderia, às vezes, produzir resultados confusos.

Parâmetros de fluxo mais fáceis de usar

Agora os parâmetros do tipo `Stream` ou `byte[]` podem aceitar os valores `string`, `string[]` ou `FileInfo`.

Por exemplo, você pode usar qualquer um dos exemplos a seguir.

```
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"', '}')
```

O AWS Tools for PowerShell converte todas as strings para `byte[]` usando a codificação UTF-8.

Estendendo o pipe por nome da propriedade

Para tornar a experiência do usuário mais consistente, agora é possível passar a entrada do pipeline especificando o nome da propriedade para qualquer parâmetro.

No exemplo a seguir, criamos um objeto personalizado com propriedades que têm nomes que correspondem aos nomes de parâmetro do cmdlet de destino. Quando o cmdlet é executado, ele consome automaticamente essas propriedades como seus parâmetros.

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

> **Note**
>
> Algumas propriedades ofereciam suporte a isso em versões anteriores do AWS Tools for PowerShell. A versão 4 torna isso mais consistente, ativando-o para todos os parâmetros.

Parâmetros comuns estáticos

Para melhorar a consistência na versão 4.0 do AWS Tools for PowerShell, todos os parâmetros são estáticos.

Em versões anteriores do AWS Tools for PowerShell, alguns parâmetros comuns como `AccessKey`, `SecretKey`, `ProfileName` ou `Region` eram dinâmicos, enquanto todos os outros parâmetros eram estáticos. Isso pode criar problemas porque PowerShell vincula os parâmetros estáticos antes dos dinâmicos. Por exemplo, digamos que você tenha executado o comando a seguir.

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

Versões anteriores do PowerShell vinculavam o valor `us-west-2` ao parâmetro `-RegionName` estático em vez do parâmetro `-Region` dinâmico. Provavelmente, isso poderia confundir os usuários.

O **`AWS.Tools`** declara e aplica os parâmetros obrigatórios

Agora os módulos do `AWS.Tools.*` declaram e aplicam os parâmetros de cmdlet obrigatórios. Quando um AWS serviço declara que um parâmetro de uma API é necessário, PowerShell solicita o parâmetro de cmdlet correspondente, caso você não o tenha especificado. Isso se aplica somente a o `AWS.Tools`. Para garantir a compatibilidade com versões anteriores, isso não se aplica a. AWSPowerShell NetCore ou AWSPowerShell.

Todos os parâmetros são anuláveis

Agora é possível atribuir `$null` aos parâmetros de tipo de valor (números e datas), Essa alteração não deve afetar os scripts existentes. Isso permite que você ignore o prompt para um parâmetro obrigatório. Os parâmetros obrigatórios são aplicados somente no `AWS.Tools`.

Se você executar o exemplo a seguir usando a versão 4, ele efetivamente ignorará a validação no lado do cliente porque você fornece um "valor" para cada parâmetro obrigatório. No entanto, a chamada de serviço de API do Amazon EC2 falhará, pois o serviço da AWS ainda exige essas informações.

```
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
```

Remover recursos defasados anteriormente

Os seguintes recursos foram defasados nas versões anteriores do AWS Tools for PowerShell e foram removidos da versão 4:

- Removido o parâmetro `-Terminate` do cmdlet `Stop-EC2Instance`. Use `Remove-EC2Instance` em vez disso.
- O `-ProfileName` parâmetro foi removido do AWSCredential cmdlet Clear-. Use `Remove-AWSCredentialProfile` em vez disso.
- Removidos os cmdlets `Import-EC2Instance` e `Import-EC2Volume`.

Conceitos básicos da AWS Tools for Windows PowerShell

Alguns tópicos desta seção descrevem os fundamentos do uso das ferramentas para Windows PowerShell depois de instalar as ferramentas. Por exemplo, eles explicam como especificar quais credenciais e regiões da AWS as ferramentas para Windows PowerShell devem usar ao interagir com a AWS.

Outros tópicos desta seção fornecem informações sobre formas avançadas de configurar as ferramentas, o ambiente e os projetos.

Tópicos

Configure a autenticação da ferramenta com AWS

Você deve estabelecer como seu código é autenticado AWS ao desenvolver com Serviços da AWS. Há diferentes maneiras de configurar o acesso programático aos AWS recursos, dependendo do ambiente e do AWS acesso disponível para você.

Para ver vários métodos de autenticação do Tools for PowerShell, consulte Autenticação e acesso no Guia de referência de AWS SDKs e ferramentas.

Este tópico pressupõe que um novo usuário esteja se desenvolvendo localmente, não tenha recebido um método de autenticação do empregador e o usará AWS IAM Identity Center para obter credenciais temporárias. Se seu ambiente não se enquadra nessas suposições, algumas das informações neste tópico podem não se aplicar a você ou algumas das informações podem já ter sido fornecidas.

A configuração desse ambiente requer várias etapas, que são resumidas da seguinte forma:

1. Habilitar e configurar o Centro de Identidade do IAM
2. Configure as ferramentas PowerShell para usar o IAM Identity Center.
3. Iniciar uma sessão do portal de AWS acesso

Habilitar e configurar o Centro de Identidade do IAM

Para ser usado AWS IAM Identity Center, ele deve primeiro ser ativado e configurado. Para ver detalhes sobre como fazer isso PowerShell, consulte a Etapa 1 no tópico sobre autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas. Especificamente, siga todas as instruções necessárias em Não estabeleci acesso por meio do Centro de Identidade do IAM.

Configure as ferramentas PowerShell para usar o IAM Identity Center.

> **Note**
>
> A partir da versão 4.1.538 do Tools for PowerShell, o método recomendado para configurar as credenciais de SSO e iniciar uma sessão do portal de AWS acesso é usar os `Invoke-AWSSSOLogin`cmdlets `Initialize-AWSSSOConfiguration`e, conforme descrito neste tópico. Se você não tiver acesso a essa versão do Tools for PowerShell (ou posterior) ou não puder usar esses cmdlets, ainda poderá executar essas tarefas usando o. AWS CLI Para saber como, consulteUse o AWS CLI para login no portal.

O procedimento a seguir atualiza o AWS `config` arquivo compartilhado com as informações de SSO que o Tools for PowerShell usa para obter credenciais temporárias. Como consequência desse procedimento, uma sessão do portal de AWS acesso também é iniciada. Se o `config` arquivo compartilhado já tiver informações de SSO e você quiser apenas saber como iniciar uma sessão do portal de acesso usando as Ferramentas para PowerShell, consulte a próxima seção neste tópico,Iniciar uma sessão do portal de AWS acesso.

1. Se você ainda não tiver feito isso, abra PowerShell e instale o AWS Tools for PowerShell conforme apropriado para seu sistema operacional e ambiente, incluindo os cmdlets comuns. Para obter informações sobre como fazer isso, consulte Instalar o AWS Tools for PowerShell.

   Por exemplo, ao instalar a versão modularizada do Tools for PowerShell no Windows, você provavelmente executaria comandos semelhantes aos seguintes:

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

2. Execute o seguinte comando . Substitua os valores de propriedade de exemplo por valores da configuração do IAM Identity Center. Para obter informações sobre essas propriedades e como encontrá-las, consulte as configurações do provedor de credenciais do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.

```
$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, você pode simplesmente usar o cmdlet sozinho e o Tools for PowerShell solicitará os valores da propriedade. `Initialize-AWSSSOConfiguration`

   Considerações sobre determinados valores de propriedade:

   - Se você simplesmente seguiu as instruções para ativar e configurar o IAM Identity Center, o valor para `-RoleName` pode ser`PowerUserAccess`. Mas se você criou um conjunto de permissões do IAM Identity Center especificamente para o PowerShell trabalho, use-o em vez disso.
   - Certifique-se de usar o Região da AWS local em que você configurou o IAM Identity Center.

3. Nesse ponto, o AWS `config` arquivo compartilhado contém um perfil chamado `my-sso-profile` com um conjunto de valores de configuração que podem ser referenciados nas Ferramentas para PowerShell. Para encontrar a localização desse arquivo, consulte Localização dos arquivos compartilhados no Guia de referência de ferramentas e SDKs da AWS .

   O Tools for PowerShell usa o provedor de token SSO do perfil para adquirir credenciais antes de enviar solicitações para. AWS O `sso_role_name` valor, que é uma função do IAM conectada a um conjunto de permissões do IAM Identity Center, deve permitir o acesso ao Serviços da AWS usado em seu aplicativo.

O exemplo a seguir mostra o perfil que foi criado usando o comando mostrado acima. Alguns dos valores da propriedade e sua ordem podem ser diferentes em seu perfil real. A `sso-session` propriedade do perfil se refere à seção chamada`my-sso-session`, que contém configurações para iniciar uma sessão do portal de AWS acesso.

```
[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. Se você já tiver uma sessão ativa do portal de AWS acesso, as Ferramentas para PowerShell informam que você já está logado.

   Se não for esse o caso, use as Ferramentas para PowerShell tentar abrir automaticamente a página de autorização de SSO em seu navegador padrão. Siga as instruções no seu navegador, que podem incluir um código de autorização de SSO, nome de usuário e senha, além de permissão para acessar AWS IAM Identity Center contas e conjuntos de permissões.

   O Tools for PowerShell informa que o login com SSO foi bem-sucedido.

Iniciar uma sessão do portal de AWS acesso

Antes de executar os comandos que acessam Serviços da AWS, você precisa de uma sessão ativa do portal de AWS acesso para que o Tools for PowerShell possa usar a autenticação do IAM Identity Center para resolver as credenciais. Para entrar no portal de AWS acesso, execute o seguinte comando em PowerShell, onde `-ProfileName my-sso-profile` está o nome do perfil que foi criado no `config` arquivo compartilhado quando você seguiu o procedimento na seção anterior deste tópico.

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

Se você já tiver uma sessão ativa do portal de AWS acesso, as Ferramentas para PowerShell informam que você já está logado.

Se não for esse o caso, use as Ferramentas para PowerShell tentar abrir automaticamente a página de autorização de SSO em seu navegador padrão. Siga as instruções no seu navegador, que podem incluir um código de autorização de SSO, nome de usuário e senha, além de permissão para acessar AWS IAM Identity Center contas e conjuntos de permissões.

O Tools for PowerShell informa que o login com SSO foi bem-sucedido.

Para testar se você já tem uma sessão ativa, execute o comando a seguir após instalar ou importar o `AWS.Tools.SecurityToken` módulo conforme necessário.

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

A resposta ao `Get-STSCallerIdentity` cmdlet relata a conta do IAM Identity Center e o conjunto de permissões configurados no arquivo compartilhado`config`.

Exemplo

Veja a seguir um exemplo de como usar o IAM Identity Center com as Ferramentas para PowerShell. Ele presume o seguinte:

- Você habilitou Centro de Identidade do IAM e o configurou conforme descrito anteriormente neste tópico. As propriedades de SSO estão no `my-sso-profile` perfil, que foi configurado anteriormente neste tópico.
- Quando você faz login por meio dos `Invoke-AWSSSOLogin` cmdlets `Initialize-AWSSSOConfiguration` ou, o usuário tem pelo menos permissões de somente leitura para o Amazon S3.
- Alguns buckets do S3 estão disponíveis para esse usuário visualizar.

Instale ou importe o `AWS.Tools.S3` módulo conforme necessário e, em seguida, use o PowerShell comando a seguir para exibir uma lista dos buckets do S3.

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

Mais informações

- Para obter mais opções de autenticação para o Tools for PowerShell, como o uso de perfis e variáveis de ambiente, consulte o capítulo de configuração no Guia de referência de AWS SDKs e ferramentas.

- Alguns comandos exigem que uma AWS região seja especificada. Há várias maneiras de fazer isso, incluindo a opção de `-Region` cmdlet, o `[default]` perfil e a variável de `AWS_REGION` ambiente. Para obter mais informações, consulte Especificar AWS regiões este guia e a AWS região no Guia de referência de AWS SDKs e ferramentas.
- Para saber mais sobre as práticas recomendadas, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.
- Para criar AWS credenciais de curto prazo, consulte Credenciais de segurança temporárias no Guia do usuário do IAM.
- Para saber mais sobre outros provedores de credenciais, consulte Provedores de credenciais padronizados no Guia de referência de AWS SDKs e ferramentas.

Tópicos

Use o AWS CLI para login no portal

A partir da versão 4.1.538 do Tools for PowerShell, o método recomendado para configurar as credenciais de SSO e iniciar uma sessão do portal de AWS acesso é usar os `Invoke-AWSSSOLogin`cmdlets `Initialize-AWSSSOConfiguration`e, conforme descrito em. Configure a autenticação da ferramenta com AWS Se você não tiver acesso a essa versão do Tools for PowerShell (ou posterior) ou não puder usar esses cmdlets, ainda poderá executar essas tarefas usando o. AWS CLI

Configure as ferramentas PowerShell para usar o IAM Identity Center por meio do AWS CLI.

Se você ainda não tiver feito isso, certifique-se de habilitar e configurar o IAM Identity Center antes de continuar.

As informações sobre como configurar as ferramentas PowerShell para usar o IAM Identity Center por meio do estão AWS CLI na Etapa 2 do tópico sobre autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas. Depois de concluir essa configuração, o sistema deverá conter os seguintes elementos:

- O AWS CLI, que você usa para iniciar uma sessão do portal de AWS acesso antes de executar seu aplicativo.

- O AWS `config` arquivo compartilhado que contém um `[default]`perfil com um conjunto de valores de configuração que podem ser referenciados nas Ferramentas para PowerShell. Para encontrar a localização desse arquivo, consulte Localização dos arquivos compartilhados no Guia de referência de ferramentas e SDKs da AWS . O Tools for PowerShell usa o provedor de token SSO do perfil para adquirir credenciais antes de enviar solicitações para. AWS O `sso_role_name` valor, que é uma função do IAM conectada a um conjunto de permissões do IAM Identity Center, deve permitir o acesso ao Serviços da AWS usado em seu aplicativo.

  O `config` arquivo de exemplo a seguir mostra um `[default]` perfil configurado com um provedor de token SSO. A configuração `sso_session` do perfil se refere à seção chamada `sso-session`. A `sso-session` seção contém configurações para iniciar uma sessão do portal de AWS acesso.

```
[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**
>
> Sua PowerShell sessão deve ter os seguintes módulos instalados e importados para que a resolução de SSO funcione:
>
> - `AWS.Tools.SSO`
> - `AWS.Tools.SSOOIDC`
>
> Se você estiver usando uma versão mais antiga do Tools for PowerShell e não tiver esses módulos, receberá um erro semelhante ao seguinte: “Assembly AWSSDK .SSOOIDC could not be found...”.

Iniciar uma sessão do portal de AWS acesso

Antes de executar os comandos que acessam Serviços da AWS, você precisa de uma sessão ativa do portal de AWS acesso para que as Ferramentas para Windows PowerShell possam usar a autenticação do IAM Identity Center para resolver as credenciais. Dependendo da duração da sessão configurada, seu acesso acabará expirando e o Tools for Windows PowerShell encontrará um erro de autenticação. Para entrar no portal de AWS acesso, execute o seguinte comando no AWS CLI.

```
aws sso login
```

Como você está usando o `[default]` perfil, não precisa chamar o comando com a `--profile` opção. Se a configuração do seu provedor de token SSO estiver usando um perfil nomeado, o comando estará `aws sso login --profile named-profile` em vez disso. Para obter mais informações sobre perfis nomeados, consulte a seção Perfis no Guia de referência de AWS SDKs e ferramentas.

Para testar se você já tem uma sessão ativa, execute o seguinte AWS CLI comando (com a mesma consideração para o perfil nomeado):

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

A resposta a esse comando deve relatar a conta do Centro de Identidade do IAM e o conjunto de permissões configurados no arquivo compartilhado `config`.

> **Note**
>
> Se você já tiver uma sessão ativa do portal de AWS acesso e executá-la`aws sso login`, não será necessário fornecer credenciais.
> O processo de login pode solicitar que você permita o AWS CLI acesso aos seus dados. Como o AWS CLI é construído sobre o SDK para Python, as mensagens de permissão podem conter variações do `botocore` nome.

Exemplo

Veja a seguir um exemplo de como usar o IAM Identity Center com as Ferramentas para PowerShell. Ele presume o seguinte:

- Você habilitou Centro de Identidade do IAM e o configurou conforme descrito anteriormente neste tópico. As propriedades de SSO estão no perfil `[default]`.
- Quando você faz login por meio do AWS CLI usando`aws sso login`, esse usuário tem pelo menos permissões de somente leitura para o Amazon S3.
- Alguns buckets do S3 estão disponíveis para esse usuário visualizar.

Use os PowerShell comandos a seguir para exibir uma lista dos buckets do 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
```

Conforme mencionado acima, como você está usando o `[default]` perfil, não precisa chamar o `Get-S3Bucket` cmdlet com a `-ProfileName` opção. Se a configuração do provedor de token de SSO estiver usando um perfil nomeado, o comando será `Get-S3Bucket -ProfileName` *`named-profile`*. Para obter mais informações sobre perfis nomeados, consulte a seção Perfis no Guia de referência de AWS SDKs e ferramentas.

Mais informações

- Para obter mais opções de autenticação para o Tools for PowerShell, como o uso de perfis e variáveis de ambiente, consulte o capítulo de configuração no Guia de referência de AWS SDKs e ferramentas.
- Alguns comandos exigem que uma AWS região seja especificada. Há várias maneiras de fazer isso, incluindo a opção de `-Region` cmdlet, o `[default]` perfil e a variável de `AWS_REGION`

ambiente. Para obter mais informações, consulte Especificar AWS regiões este guia e a AWS região no Guia de referência de AWS SDKs e ferramentas.

- Para saber mais sobre as práticas recomendadas, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.
- Para criar AWS credenciais de curto prazo, consulte Credenciais de segurança temporárias no Guia do usuário do IAM.
- Para saber mais sobre outros provedores de credenciais, consulte Provedores de credenciais padronizados no Guia de referência de AWS SDKs e ferramentas.

Especificar AWS regiões

Há duas maneiras de especificar a AWS região a ser usada ao executar AWS Tools for PowerShell comandos:

- Use o parâmetro `-Region` comum em comandos individuais.
- Use o comando `Set-DefaultAWSRegion` para definir uma região padrão para todos os comandos.

Muitos AWS cmdlets falham se o Tools for Windows não PowerShell conseguir descobrir qual região usar. As exceções incluem cmdlets para Amazon S3, Amazon SES AWS Identity and Access Management e, que automaticamente assumem como padrão um endpoint global.

Para especificar a região para um único AWS comando

Adicione o parâmetro `-Region` ao seu comando, como o seguinte.

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

Para definir uma região padrão para todos os comandos da AWS CLI na sessão atual

No prompt de PowerShell comando, digite o comando a seguir.

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

> **Note**
>
> Essa configuração é persistida apenas durante a sessão atual. Para aplicar a configuração a todas as suas PowerShell sessões, adicione esse comando ao seu PowerShell perfil da mesma forma que você fez com o `Import-Module` comando.

Para visualizar a região padrão atual para todos os comandos da AWS CLI

No prompt de PowerShell comando, digite o comando a seguir.

```
PS > Get-DefaultAWSRegion

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

Para limpar a região padrão atual para todos os comandos da AWS CLI

No prompt de PowerShell comando, digite o comando a seguir.

```
PS > Clear-DefaultAWSRegion
```

Para ver uma lista de todas as AWS regiões disponíveis

No prompt de PowerShell comando, digite o comando a seguir. A terceira coluna na saída de exemplo identifica qual região é a padrão para a sessão atual.

```
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**
>
> Algumas regiões podem ser compatíveis, mas não estão incluídas na saída do cmdlet `Get-AWSRegion`. Por exemplo, isso às vezes acontece em regiões que ainda não são globais. Se você não puder especificar uma região adicionando o parâmetro `-Region` a um comando, tente especificar a região em um endpoint personalizado, conforme mostrado na próxima seção.

Especificar um endpoint não padrão ou personalizado

Especifique um endpoint personalizado como URL adicionando o parâmetro `-EndpointUrl` comum ao PowerShell comando do Tools for Windows, no formato de exemplo a seguir.

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

Veja a seguir um comando de exemplo que usa o cmdlet `Get-EC2Instance`. Neste exemplo, o endpoint personalizado está na região `us-west-2` ou Oeste dos EUA (Oregon), mas você poderá usar qualquer outra região da AWS compatível, incluindo regiões que não são enumeradas pelo `Get-AWSRegion`.

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

Mais informações

Para obter informações adicionais sobre AWS regiões, consulte AWS Região no Guia de referência de AWS SDKs e ferramentas.

Configurar a identidade federada com o AWS Tools for PowerShell

Para permitir que os usuários na sua organização acessem os recursos da AWS, você deverá configurar um método de autenticação padrão e reproduzível para fins de segurança, auditabilidade, conformidade e a capacidade de oferecer suporte à separação de contas e funções. Embora seja comum fornecer aos usuários a capacidade de acessar APIs da AWS, sem acesso a uma API federada, também é necessário criar usuários do AWS Identity and Access Management (IAM),

o que anula a finalidade de usar a federação. Este tópico descreve o suporte ao SAML (Security Assertion Markup Language) no AWS Tools for PowerShell que facilita sua solução de acesso federado.

O suporte ao SAML no AWS Tools for PowerShell permite que você forneça aos usuários acesso federado aos produtos da AWS. O SAML é um formato padrão aberto, baseado em XML, para a transmissão de dados de autorização e autenticação de usuários entre serviços; especificamente, entre um provedor de identidade (como o Active Directory Federation Services) e um provedor de serviços (como a AWS). Para obter mais informações sobre SAML e como ele funciona, consulte SAML na Wikipédia ou Especificações técnicas do SAML no site da OASIS (Organization for the Advancement of Structured Information Standards). O suporte ao SAML no AWS Tools for PowerShell é compatível com o SAML 2.0.

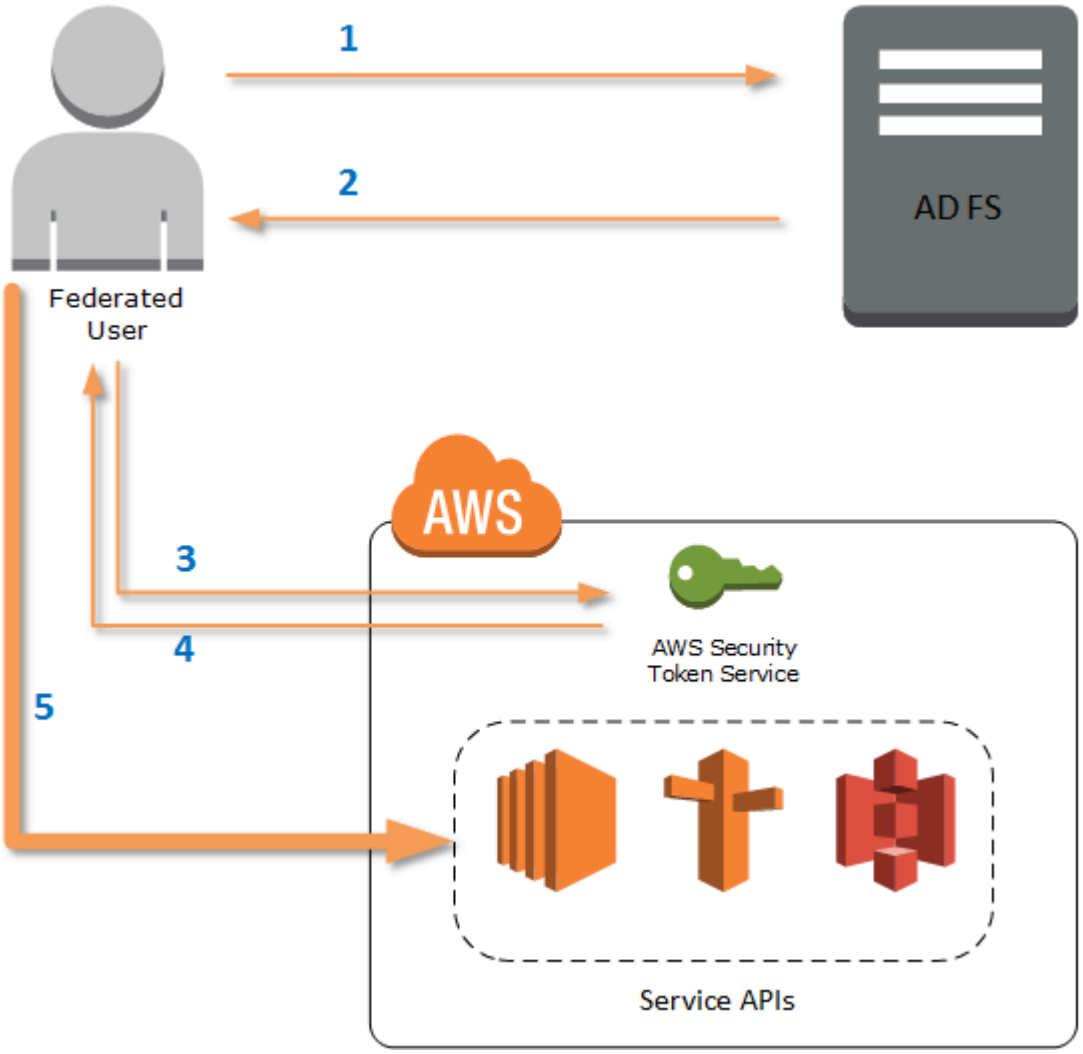
Pré-requisitos

Você deve ter o seguinte antes de tentar usar o suporte a SAML pela primeira vez.

- Uma solução de identidade federada corretamente integrada com sua conta da AWS para acesso ao console usando apenas as credenciais da sua organização. Para obter mais informações sobre como fazer isso especificamente para o Serviços de Federação do Active Directory (AD FS), consulte Sobre a federação SAML 2.0 no Manual do usuário do IAM e a postagem do blog Como habilitar a federação para a AWS usando Windows Active Directory, AD FS e SAML 2.0). Embora a publicação do blog aborde o AD FS 2.0, as etapas serão semelhantes se você estiver executando o AD FS 3.0.
- Versão 3.1.31.0 ou mais recente do AWS Tools for PowerShell instalada em sua estação de trabalho local.

Como um usuário com federação de identidade obtém acesso federado a APIs de serviço da AWS

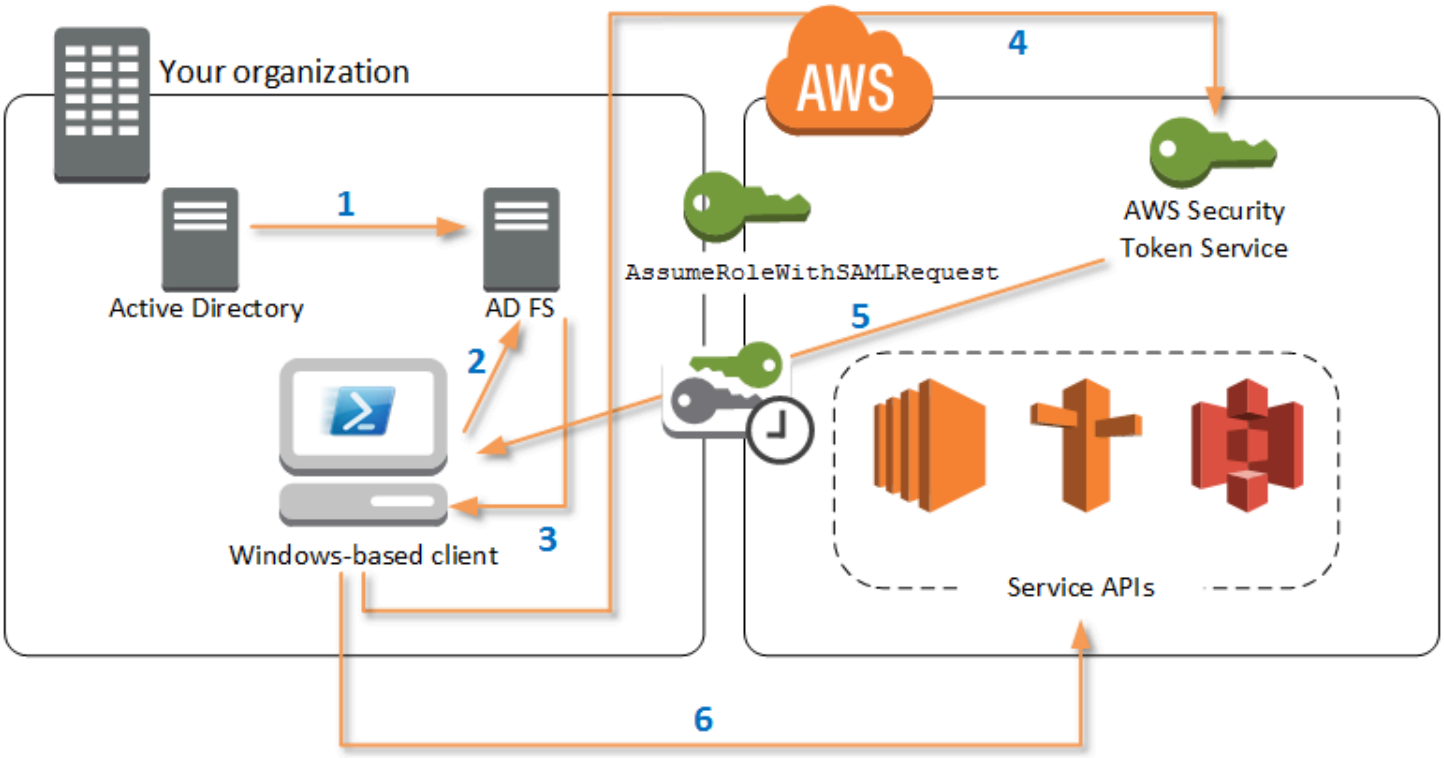
O processo a seguir descreve, em alto nível, como um usuário do Active Directory (AD) é federado pelo AD FS para obter acesso aos recursos da AWS.



1. O cliente no computador do usuário federado autentica em relação ao AD FS.
2. Se a autenticação for bem-sucedida, o AD FS enviará ao usuário uma declaração do SAML.
3. O cliente do usuário envia a declaração do SAML para o AWS Security Token Service (STS) como parte de uma solicitação de federação do SAML.
4. O STS retorna uma resposta SAML que contém credenciais temporárias da AWS para uma função que o usuário pode assumir.
5. O usuário acessa as APIs do produtos da AWS ao incluir essas credenciais temporárias na solicitação feita pelo AWS Tools for PowerShell.

Como funciona o suporte ao SAML no AWS Tools for PowerShell

Esta seção descreve como os cmdlets do AWS Tools for PowerShell permitem a configuração de federação de identidades com base em SAML para os usuários.



1. O AWS Tools for PowerShell faz a autenticação no AD FS usando as credenciais atuais do usuário do Windows, ou de forma interativa, quando o usuário tenta executar um cmdlet que requer credenciais para chamar a AWS.
2. O AD FS autentica o usuário.
3. O AD FS gera uma resposta de autenticação do SAML 2.0 que inclui uma declaração. A finalidade da declaração é identificar e fornecer informações sobre o usuário. O AWS Tools for PowerShell extrai a lista de funções autorizadas do usuário da declaração do SAML.
4. O AWS Tools for PowerShell encaminha a solicitação do SAML, incluindo os nomes de recurso da Amazon (ARN) da função solicitada, para o STS, fazendo a chamada à API `AssumeRoleWithSAMLRequest`.
5. Se a solicitação do SAML for válida, o STS retornará uma resposta que contém AWS, `AccessKeyId` e `SecretAccessKey` da `SessionToken`. Essas credenciais duram 3.600 segundos (1 hora).
6. O usuário agora tem credenciais válidas para trabalhar com quaisquer APIs de produto da AWS às quais a função do usuário está autorizada a acessar. O AWS Tools for PowerShell aplica

automaticamente essas credenciais para quaisquer chamadas subsequentes de API da AWS e as renova automaticamente quando expiram.

> **Note**
>
> Quando as credenciais expiram e novas credenciais são necessárias, o AWS Tools for PowerShell reautentica automaticamente com o AD FS e obtém novas credenciais para a próxima hora. Para usuários de contas associadas a um domínio, esse processo ocorre silenciosamente. Para contas que não estão associadas a um domínio, o AWS Tools for PowerShell solicita que os usuários insiram suas credenciais para que possam reautenticar.

Como usar cmdlets de configuração de SAML do PowerShell

O AWS Tools for PowerShell inclui dois novos cmdlets que oferecem suporte ao SAML.

- O `Set-AWSSamlEndpoint` configura o endpoint do AD FS, atribui um nome amigável ao endpoint e, opcionalmente, descreve o tipo de autenticação do endpoint.
- O `Set-AWSSamlRoleProfile` cria ou edita um perfil de conta de usuário que você deseja associar a um endpoint do AD FS, identificado ao especificar o nome amigável fornecido ao cmdlet `Set-AWSSamlEndpoint`. Cada perfil de função é mapeado para uma única função que um usuário está autorizado a executar.

  Assim como com perfis de credenciais da AWS, você atribui um nome amigável ao perfil de função. Você pode usar o mesmo nome amigável com o cmdlet `Set-AWSCredential` ou como o valor do parâmetro `-ProfileName` para qualquer cmdlet que chame APIs de serviço da AWS.

Abra uma nova sessão do AWS Tools for PowerShell. Se estiver executando o PowerShell 3.0 ou mais recente, o módulo do AWS Tools for PowerShell será automaticamente importado quando você executar qualquer um de seus cmdlets. Se estiver executando o PowerShell 2.0, você deverá importar o módulo manualmente executando o cmdlet ``Import-Module``, conforme mostrado no exemplo a seguir.

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

Como executar os cmdlets **`Set-AWSSamlEndpoint`** e **`Set-AWSSamlRoleProfile`**

1. Primeiro, defina as configurações de endpoint para o sistema do AD FS. A maneira mais simples de fazer isso é armazenar o endpoint em uma variável, como mostrado nesta etapa. Certifique-se de substituir os IDs de contas de espaço reservado e o nome de host do AD FS com os seus próprios IDs de contas e nome de host do AD FS. Especifique o nome de host do AD FS no parâmetro `Endpoint`.

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

2. Para criar as configurações de endpoint, execute o cmdlet `Set-AWSSamlEndpoint`, especificando o valor correto para o parâmetro `AuthenticationType`. Os valores válidos incluem `Basic`, `Digest`, `Kerberos`, `Negotiate` e `NTLM`. Se você não especificar esse parâmetro, o valor padrão será `Kerberos`.

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

   O cmdlet retorna o nome amigável atribuídos usando o parâmetro `-StoreAs`, para que você possa usá-lo ao executar `Set-AWSSamlRoleProfile` na próxima linha.

3. Agora, execute o cmdlet `Set-AWSSamlRoleProfile` para fazer a autenticação com o provedor de identidade do AD FS e obter o conjunto de funções (na declaração do SAML) que o usuário está autorizado a executar.

   O cmdlet `Set-AWSSamlRoleProfile` usa o conjunto retornado de funções para solicitar que o usuário selecione uma função a ser associada ao perfil especificado ou confirme que os parâmetros de dados fornecidos nos parâmetros estão presentes (se não estiverem, será solicitado que o usuário escolha). Se o usuário estiver autorizado para apenas uma função, o cmdlet associará a função ao perfil automaticamente, sem fazer a solicitação ao usuário. Não há necessidade de fornecer uma credencial para configurar um perfil para uso associado a um domínio.

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

   Como alternativa, para contas não associadas a um domínio, você pode fornecer credenciais do Active Directory e, em seguida, selecionar uma função da AWS à qual o usuário tem acesso,

como mostrado na linha a seguir. Isso será útil se você tiver contas de usuário diferentes do Active Directory para diferenciar funções em sua organização (por exemplo, funções administrativas).

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

4. Em qualquer um dos casos, o cmdlet `Set-AWSSamlRoleProfile` solicita que você escolha a função que deve ser armazenada no perfil. O exemplo a seguir mostra duas funções disponíveis: `ADFS-Dev` e `ADFS-Production`. As funções do IAM são associadas às credenciais de login do AD pelo administrador do AD FS.

```
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"):
```

   Como alternativa, é possível especificar uma função sem o prompt, inserindo o `RoleARN`, o `PrincipalARN` e os parâmetros opcionais `NetworkCredential`. Se a função especificada não estiver listada na declaração retornada pela autenticação, o usuário será solicitado a escolher entre as funções disponíveis.

```
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. Você pode criar perfis para todas as funções em um único comando ao adicionar o parâmetro `StoreAllRoles`, conforme mostrado no código a seguir. Observe que o nome da função é usado como o nome do perfil.

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

Como usar perfis de função para executar cmdlets que exigem credenciais da AWS

Para executar cmdlets que exigem credenciais da AWS, use perfis de função definidos no arquivo de credenciais compartilhadas da AWS. Forneça o nome de um perfil de função para `Set-AWSCredential` (ou como o valor para qualquer parâmetro `ProfileName` no AWS Tools for PowerShell) para obter credenciais temporárias da AWS automaticamente para a função que está descrita no perfil.

Embora você use apenas um perfil de função por vez, pode alternar entre os perfis em uma sessão de shell. O cmdlet `Set-AWSCredential` não faz a autenticação e não obtém credenciais quando você o executa sozinho; o cmdlet registra que você deseja usar um perfil de função especificado. Até que você execute um cmdlet que exija credenciais da AWS, não ocorrerá nenhuma autenticação ou solicitação de credenciais.

Agora é possível usar as credenciais temporárias da AWS obtidas com o perfil `SAMLDemoProfile` para funcionar com APIs de produtos da AWS. As seções a seguir mostram exemplos de como usar os perfis de função.

Exemplo 1: Definir uma função padrão com **`Set-AWSCredential`**

Este exemplo define uma função padrão para uma sessão do AWS Tools for PowerShell usando `Set-AWSCredential`. Em seguida, você pode executar cmdlets que exijam credenciais e sejam autorizados pela função especificada. Este exemplo lista todas as instâncias do Amazon Elastic Compute Cloud na região Oeste dos EUA (Oregon) que estão associadas ao perfil especificado com o 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}
```

Exemplo 2: Alterar perfis de função durante uma sessão do PowerShell

Este exemplo lista todos os buckets do Amazon S3 disponíveis na conta da AWS da função associada ao perfil `SAMLDemoProfile`. O exemplo mostra que, embora você possa ter usado outro perfil anteriormente em sua sessão do AWS Tools for PowerShell, é possível alterar os perfis especificando um valor diferente para o parâmetro `-ProfileName` com cmdlets que ofereçam suporte a ele. Esta é uma tarefa comum para administradores que gerenciam o Amazon S3 a partir da linha de comando do 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
```

Observe que o cmdlet `Get-S3Bucket` especifica o nome do perfil criado executando o cmdlet `Set-AWSSamlRoleProfile`. Este comando poderá ser útil se você tiver configurado um perfil de função anteriormente em sua sessão (por exemplo, ao executar o cmdlet `Set-AWSCredential`) e quiser usar um perfil de função diferente para o cmdlet `Get-S3Bucket`. O gerenciador de perfil disponibiliza credenciais temporárias para o cmdlet `Get-S3Bucket`.

Embora as credenciais expirem após uma hora (um limite imposto pelo STS), o AWS Tools for PowerShell atualizará automaticamente as credenciais solicitando uma nova declaração do SAML quando a ferramenta detectar que as credenciais atuais expiraram.

Para usuários associados a um domínio, esse processo ocorre sem interrupção, porque a identidade do Windows do usuário atual é usada durante a autenticação. Para contas de usuário não associadas a um domínio, o AWS Tools for PowerShell mostra um prompt de credencial do PowerShell solicitando a senha do usuário. O usuário fornece credenciais que são usadas para autenticar novamente o usuário e obter uma nova asserção.

Exemplo 3: Obter instâncias em uma região

O exemplo a seguir lista todas as instâncias do Amazon EC2 na região Ásia-Pacífico (Sydney) associadas à conta usada pelo perfil `ADFS-Production`. Este é um comando útil para retornar todas as instâncias do Amazon EC2 em uma região.

```
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
```

Leitura adicional

Para obter informações gerais sobre como implementar o acesso federado à API, consulte How to Implement a General Solution for Federated API/CLI Access Using SAML 2.0 (Como implementar uma solução geral para acesso federado à CLI/API usando SAML 2.0).

Em caso de dúvidas ou comentários, acesse os fóruns de desenvolvedores da AWS para ler sobre Scripts do PowerShell ou Desenvolvimento em .NET.

Aliases e descoberta de cmdlet

Esta seção mostra como listar serviços que são compatíveis com o AWS Tools for PowerShell, como mostrar o conjunto de cmdlets fornecido pelo AWS Tools for PowerShell para apoiar esses serviços e como encontrar nomes de cmdlet alternativos (também chamados de aliases) para acessar esses serviços.

Descoberta de cmdlets

Todas as operações de serviços da AWS (ou APIs) são documentadas no Guia de referência da API para cada serviço. Por exemplo, consulte Referência da API do IAM. Há, na maioria dos casos, uma correspondência um-para-um entre a API de um serviço da AWS e um cmdlet do PowerShell da AWS. Para obter o nome do cmdlet correspondente a um nome de API de serviço da AWS, execute o cmdlet `Get-AWSCmdletName` da AWS com o parâmetro `-ApiOperation` e o nome da API de serviço da AWS. Por exemplo, para obter todos os nomes de cmdlet baseados em qualquer API de serviço da AWS `DescribeInstances` disponível, execute o seguinte 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
```

O parâmetro `-ApiOperation` é o parâmetro padrão, portanto, você pode omitir o nome do parâmetro. O exemplo a seguir é equivalente ao anterior:

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

Se você sabe os nomes da API e do produto, inclua o parâmetro `-Service` junto com o prefixo do substantivo do cmdlet ou parte do nome do serviço da AWS. Por exemplo, o prefixo de substantivo do cmdlet para Amazon EC2 é `EC2`. Para obter o nome do cmdlet que corresponde à API `DescribeInstances` no serviço Amazon EC2, execute um dos comandos a seguir. Todos eles resultam na mesma saída:

```
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
```

Valores de parâmetro nesses comandos fazem distinção de maiúsculas e minúsculas.

Se não souber o nome da API do serviço da AWS ou do serviço da AWS desejado, você poderá usar o parâmetro `-ApiOperation` junto com o padrão para fazer a correspondência e o parâmetro `-MatchWithRegex`. Por exemplo, para obter todos os nomes de cmdlets disponíveis que contêm `SecurityGroup`, execute o comando a seguir.

```
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
```

Se você souber o nome do serviço da AWS, mas não a API do serviço da AWS, adicione o parâmetro `-MatchWithRegex` e o parâmetro `-Service` para limitar a pesquisa a um único serviço. Por exemplo, para obter todos os nomes de cmdlets que contêm `SecurityGroup` apenas no serviço do Amazon EC2, execute o comando a seguir.

```
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
```

Se você souber o nome do comando AWS Command Line Interface (AWS CLI), você pode usar o parâmetro `-AwsCliCommand` e o nome do comando AWS CLI desejado para obter o nome do cmdlet baseado na mesma API. Por exemplo, para obter o nome do cmdlet correspondente à chamada de comando `authorize-security-group-ingress` da AWS CLI no serviço Amazon EC2, execute o seguinte comando:

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

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

O cmdlet `Get-AWSCmdletName` precisa do nome do comando da AWS CLI apenas o suficiente para identificar o serviço e a API da AWS.

Para obter uma lista de todos os cmdlets no Tools for PowerShell Core, execute o cmdlet `Get-Command` do PowerShell, conforme mostrado no exemplo a seguir.

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

Você pode executar o mesmo comando com `-Module AWSPowerShell` para ver os cmdlets no AWS Tools for Windows PowerShell.

O cmdlet `Get-Command` gera a lista de cmdlets em ordem alfabética. Observe que, por padrão, a lista é classificada por verbo do PowerShell, em vez de substantivo do PowerShell.

Para classificar os resultados por serviço, execute o comando a seguir:

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

Para filtrar os cmdlets retornados pelo cmdlet `Get-Command`, canalize a saída para o cmdlet `Select-String` do PowerShell. Por exemplo, para visualizar o conjunto de cmdlets com regiões da AWS, execute o seguinte 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
```

Você também pode encontrar cmdlets para um serviço específico filtrando o prefixo de serviço de substantivos de cmdlet. Para ver a lista de prefixos de serviço disponíveis, execute `Get-AWSPowerShellVersion -ListServiceVersionInfo`. O exemplo a seguir retorna cmdlets que oferecem suporte ao serviço 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 de cmdlets e aliases

Os cmdlets no AWS Tools for PowerShell de cada serviço são baseados em métodos fornecidos pelo AWS SDK para o serviço. No entanto, devido a convenções de nomenclatura obrigatórias do PowerShell, o nome de um cmdlet pode ser diferente do nome da chamada ou método de API no qual é baseado. Por exemplo, o cmdlet `Get-EC2Instance` é baseado no método `DescribeInstances` do Amazon EC2.

Em alguns casos, o nome do cmdlet pode ser semelhante a um nome de método, mas talvez execute uma função diferente. Por exemplo, o método `GetObject` do Amazon S3 recupera um objeto do Amazon S3. No entanto, o cmdlet `Get-S3Object` retorna informações sobre um objeto do Amazon S3 em vez do próprio 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 obter um objeto do S3 com o AWS Tools for PowerShell, execute o 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**
>
> A ajuda de um cmdlet da AWS fornece o nome da API do AWS SDK no qual o cmdlet é baseado.
> Para obter mais informações sobre os verbos padrão do PowerShell e seus significados, consulte Verbos aprovados para comandos do PowerShell.

Todos os cmdlets da AWS que usam o verbo `Remove`, e o cmdlet `Stop-EC2Instance` quando você adiciona o parâmetro `-Terminate`, solicitam confirmação antes de continuar. Para ignorar a confirmação, adicione o parâmetro `-Force` ao seu comando.

> **Important**
>
> Os cmdlets da AWS não oferecem suporte à opção `-WhatIf`.

Aliases

Configuração das instalações de AWS Tools for PowerShell e arquivos de alias que contêm alias para muitos cmdlets da AWS. Você pode considerar esses aliases mais intuitivos do que os nomes dos cmdlets. Por exemplo, nomes de serviço e nomes de método do AWS SDK substituem verbos e substantivos do PowerShell em alguns alias. Um exemplo é o alias `EC2-DescribeInstances`.

Outros aliases usam verbos que, embora não sigam as convenções padrão do PowerShell, podem descrever melhor a operação real. Por exemplo, o arquivo de alias mapeia o alias `Get-S3Content` para o cmdlet `Read-S3Object`.

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

O arquivo de aliases está localizado no diretório de instalação do AWS Tools for PowerShell. Para carregar os aliases em seu ambiente, faça dot-source do arquivo. Veja a seguir um exemplo de Windows.

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

Para um shell do Linux ou macOS, ele pode ter a seguinte aparência:

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

Para mostrar todos os alias de AWS Tools for PowerShell, execute o comando a seguir. Esse comando usa o alias `?` do cmdlet `Where-Object` do PowerShell e a propriedade `Source` para filtrar somente aliases provenientes do 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 adicionar seus próprios aliases a esse arquivo, talvez seja necessário aumentar o valor da variável de preferência `$MaximumAliasCount` do PowerShell para um valor superior a 5500. O valor padrão é 4096. Você pode aumentá-lo para um máximo de 32768. Para fazer isso, execute o seguinte.

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

Para verificar se a alteração foi bem-sucedida, insira o nome da variável para mostrar seu valor atual.

```
PS > $MaximumAliasCount
32768
```

Pipeline $AWSHistory

Para as chamadas de produtos da AWS que retornam coleções, os objetos dentro da coleção são enumerados para o pipeline. Os objetos de resultado que contêm campos adicionais além da coleção e que não são campos de controle de paginação têm esses campos adicionados como propriedades de observação para chamadas. Essas propriedades são registradas na nova variável de sessão `$AWSHistory`, caso você precise acessar esses dados. A variável `$AWSHistory` é descrita na próxima seção.

> **Note**
>
> Em versões do Tools for Windows PowerShell anteriores à v1.1, o próprio objeto de coleção era emitido, o que exigia o uso de foreach {$_.getenumerator()} para continuar o pipeline.

Exemplos

O exemplo a seguir retorna uma lista de regiões da AWS e suas imagens de máquina do Amazon EC2 (AMIs) em cada região.

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

O exemplo a seguir interrompe todas as instâncias do Amazon EC2 na região padrão atual.

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

Como as coleções são enumeradas para o pipeline, a saída de um determinado cmdlet pode ser `$null`, um único objeto ou uma coleção. Se for uma coleção, você poderá usar a propriedade `.Count` para determinar o tamanho da coleção. No entanto, a propriedade `.Count` não está presente quando apenas um único objeto é emitido. Se o script precisar determinar, de maneira consistente, quantos objetos foram emitidos, verifique a propriedade `EmittedObjectsCount` do último valor do comando em `$AWSHistory`.

$AWSHistory

Para oferecer um suporte melhor ao pipeline, a saída dos cmdlets da AWS não é mais remodelada para incluir a resposta do serviço e as instâncias resultantes como propriedades de observação no objeto de coleção emitido. Em vez disso, para essas chamadas que emitem uma única coleção como saída, a coleção agora é enumerada para o pipeline do PowerShell. Isso significa que a resposta do AWS SDK e os dados do resultado não podem existir no pipe, pois não há objeto de coleção ao qual eles possam ser anexados.

Embora a maioria dos usuários provavelmente não precise desses dados, podem ser úteis para fins de diagnóstico, pois você pode ver exatamente o que foi enviado e recebido das chamadas de serviço subjacentes da AWS pelo cmdlet.

A partir da versão 1.1, esses dados e muito mais agora estão disponíveis em uma nova variável de shell chamada `$AWSHistory`. Essa variável mantém um registro de invocações de cmdlet da AWS e as respostas de serviço que foram recebidas para cada invocação. Esse histórico também pode ser configurado para registrar as solicitações de serviço que cada cmdlet fez. Dados úteis adicionais, como o tempo de execução geral do cmdlet, também podem ser obtidos de cada entrada. Por motivos de segurança, solicitações e respostas que contêm dados confidenciais não são registradas por padrão. No entanto, o histórico poderá ser configurado para substituir esse comportamento, se necessário. Para ter mais informações, consulte o cmdlet `Set-AWSHistoryConfiguration` mostrado abaixo.

Cada entrada na lista `$AWSHistory.Commands` lista é do tipo `AWSCmdletHistory`. Esse tipo tem os seguintes membros úteis:

CmdletName

Nome do cmdlet.

CmdletStart

Data e hora em que o cmdlet foi executado.

CmdletEnd

Data e hora em que o cmdlet concluiu todo o processamento.

Solicitações

Se a gravação da solicitação estiver ativada, lista das últimas solicitações de serviço.

Respostas

Lista das últimas respostas de serviço recebidas.

LastServiceResponse

Assistente para retornar a resposta de serviço mais recente.

LastServiceRequest

Assistente para retornar a solicitação de serviço mais recente, se disponível.

Observe que a variável `$AWSHistory` não será criada até que um cmdlet da AWS que está fazendo uma chamada de serviço seja usado. Ela é avaliada como $null até esse ponto.

> **Note**
>
> As versões anteriores do Tools for Windows PowerShell emitiam dados relacionados a respostas de serviço como propriedades `Note` no objeto retornado. Agora são encontrados nas entradas de resposta que são registradas para cada invocação na lista.

Set-AWSHistoryConfiguration

Uma invocação de cmdlet pode ter zero ou mais entradas de solicitação e resposta de serviço. Para limitar o impacto da memória, a lista `$AWSHistory` mantém um registro de apenas as cinco últimas execuções do cmdlet por padrão; e para cada uma, as últimas cinco respostas (e, se ativado, as últimas cinco solicitações de serviço). Esses limites padrão podem ser alterados ao executar o cmdlet `Set-AWSHistoryConfiguration`. Ele permite que você controle o tamanho da lista e se as solicitações de serviço também serão registradas:

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

Todos os parâmetros são opcionais.

O parâmetro `MaxCmdletHistory` define o número máximo de cmdlets que podem ser rastreados a qualquer momento. O valor 0 desativa o registro da atividade do cmdlet da AWS. O parâmetro `MaxServiceCallHistory` define o número máximo de respostas de serviço (e/ou solicitações) que são controladas para cada cmdlet. O parâmetro `RecordServiceRequests`, se especificado, ativa o rastreamento de solicitações de serviço para cada cmdlet. O parâmetro `IncludeSensitiveData`, se especificado, ativará o rastreamento de respostas e solicitações de serviço (se rastreadas) que contêm dados confidenciais de cada cmdlet.

Se executar sem parâmetros, o `Set-AWSHistoryConfiguration` on simplesmente desativará qualquer gravação de solicitação anterior, deixando os tamanhos atuais da lista inalterados.

Para limpar todas as entradas da lista de histórico atual, execute o cmdlet `Clear-AWSHistory`.

Exemplos do **`$AWSHistory`**

Enumere os detalhes dos cmdlets da AWS que estão sendo mantidos na lista para o pipeline.

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

Acesse os detalhes do último cmdlet da AWS que foi executado:

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

Acesse os detalhes da última resposta de serviço do último cmdlet da AWS que foi executado. Se a saída de um cmdlet da AWS for paginação, ele poderá fazer várias chamadas de serviço para obter todos os dados ou o volume máximo de dados (determinado por parâmetros no cmdlet).

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

Acesse os detalhes da última solicitação feita (novamente, um cmdlet pode fazer mais de uma solicitação se estiver paginando em nome do usuário). Produz $null, a menos que o rastreamento da solicitação de serviço esteja habilitado.

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

Página-para-conclusão automática para operações que retornam várias páginas

Para APIs de serviço que impõem um número máximo padrão de retorno de objetos para uma determinada chamada ou que oferecem suporte a conjuntos de resultados, todos os cmdlets

apresentam "página-para-conclusão" por padrão. Cada cmdlet faz quantas chamadas forem necessárias em seu nome para retornar o conjunto de dados completo para o pipeline.

No exemplo a seguir, que usa `Get-S3Object`, a variável `$c` contém instâncias `S3Object` para cada chave no bucket `test`, potencialmente um conjunto de dados muito grande.

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

Se você deseja manter o controle sobre a quantidade de dados retornados, poderá continuar a usar parâmetros nos cmdlets individuais (por exemplo, `MaxKey` em `Get-S3Object`) ou lidar explicitamente com a paginação usando uma combinação de parâmetros de paginação nos cmdlets e dados colocados na variável `$AWSHistory` para obter os próximos dados de token do serviço. O exemplo a seguir usa o parâmetro MaxKeys para limitar o número de instâncias `S3Object` retornadas às primeiras 500 encontradas no bucket.

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

Para saber se mais dados estavam disponíveis, mas não foram retornados, use a entrada da variável de sessão `$AWSHistory` que registrou as chamadas de serviço feitas pelo cmdlet.

Se a expressão a seguir for avaliada como $true, você poderá encontrar o marcador `next` para o próximo conjunto de resultados usando `$AWSHistory.LastServiceResponse.NextMarker`.

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

Para controlar manualmente a paginação com `Get-S3Object`, use uma combinação dos parâmetros `MaxKey` e `Marker` para o cmdlet e as observações `IsTruncated`/`NextMarker` na última resposta registrada. No exemplo a seguir, a variável `$c` contém até um máximo de 500 instâncias `S3Object` para os próximos 500 objetos que podem ser encontrados no bucket após o início do marcador de prefixo de chaves especificado.

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

Resolução de perfil e credenciais

Ordem de pesquisa de credenciais

Quando você executa um comando, o AWS Tools for PowerShell procura credenciais na ordem a seguir. Ele para ao encontrar credenciais utilizáveis.

1. As credenciais literais incorporadas como parâmetros na linha de comando.

   É altamente recomendável usar perfis em vez de colocar as credenciais literais em suas linhas de comando.

2. Um local de perfil ou um nome de perfil especificado.

   - Se você especificar apenas um nome de perfil, o comando procura um perfil especificado do armazenamento do AWS SDK e, se esse perfil não existir, o perfil especificado no arquivo de credenciais compartilhadas da AWS no local padrão.
   - Se você especificar apenas um local de perfil, o comando procurará o perfil `default` desse arquivo de credenciais.
   - Se você especificar um nome e um local, o comando procurará o perfil especificado nesse arquivo de credenciais.

   Se o perfil ou o local especificado não for encontrado, o comando lançará uma exceção. A pesquisa passará para as seguintes etapas somente se você não tiver especificado um perfil ou local.

3. Credenciais especificadas pelo parâmetro `-Credential`.
4. O perfil da sessão, se existir.
5. Use um perfil padrão, na seguinte ordem:

   a. O perfil `default` no armazenamento do AWS SDK.

   b. O perfil `default` no arquivo de credenciais compartilhadas da AWS.

   c. O perfil `AWS PS Default` no armazenamento do AWS SDK.

6. Se o comando estiver sendo executado em uma instância do Amazon EC2 configurada para usar uma função do IAM, as credenciais temporárias da instância do EC2 acessadas no perfil da instância.

   Para obter mais informações sobre o uso de funções do IAM, para instâncias do Amazon EC2. consulte AWS SDK for .NET.

Se essa pesquisa não conseguir localizar as credenciais especificadas, o comando lançará uma exceção.

Informações adicionais sobre usuários e perfis

Para executar os comandos das ferramentas para PowerShell na AWS, você precisa ter uma combinação de usuários, conjuntos de permissões e perfis de serviço apropriados para as tarefas.

Os usuários específicos, os conjuntos de permissões e os perfis de serviço que você cria e a maneira como você os utiliza dependerão de seus requisitos. Veja a seguir algumas informações adicionais sobre por que eles podem ser usados e como criá-los.

Usuários e conjuntos de permissões

Embora seja possível usar uma conta de usuário do IAM com credenciais de longo prazo para acessar os serviços da AWS, essa não é mais uma prática recomendada e deve ser evitada. Mesmo durante o desenvolvimento, é prática recomendada criar usuários e conjuntos de permissões no AWS IAM Identity Center e usar credenciais temporárias fornecidas por uma fonte de identidade.

Para desenvolvimento, você pode usar o usuário que criou ou recebeu em Configurar autenticação nas ferramentas. Se você tiver permissões apropriadas do AWS Management Console, também poderá criar conjuntos de permissões diferentes com privilégio mínimo para esse usuário ou criar usuários especificamente para projetos de desenvolvimento, fornecendo conjuntos de permissões com privilégio mínimo. A ação que você escolher, se for o caso, dependerá das circunstâncias.

Para obter mais informações sobre esses usuários e conjuntos de permissões e como criá-los, consulte Autenticação e acesso no Guia de referência de AWS SDKs e ferramentas e Conceitos básicos no Guia do usuário do AWS IAM Identity Center.

Perfis de serviço

Você pode configurar um perfil de serviço da AWS para acessar serviços da AWS em nome dos usuários. Esse tipo de acesso é apropriado quando várias pessoas estão executando a aplicação remotamente; por exemplo, em uma instância do Amazon EC2 que você criou para essa finalidade.

O processo de criação de um perfil de serviço varia de acordo com a situação, mas é basicamente o seguinte.

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.

2. Selecione Funções e, em seguida, Criar função.
3. Escolha Serviço da AWS, encontre e selecione EC2 (por exemplo) e, depois, selecione o caso de uso do EC2 (por exemplo).
4. Escolha Próximo e selecione as políticas apropriadas para os serviços da AWS que a aplicação usará.

   > **Warning**
   >
   > NÃO escolha a política AdministratorAccess porque ela concede permissões de leitura e gravação a quase tudo na sua conta.

5. Escolha Next (Próximo). Insira o Nome da função, a Descrição e as tags que desejar.

   Você encontrará informações sobre tags em Controlar o acesso usando tags de recursos da AWS no Guia do usuário do IAM.
6. Selecione Create role (Criar função).

Você pode encontrar informações gerais sobre os perfis do IAM em Identidades do IAM (usuários, grupos de usuários e perfis) no Guia do usuário do IAM. Encontre informações detalhadas sobre perfis no tópico Perfis do IAM.

Uso de credenciais herdadas

Os tópicos desta seção fornecem informações sobre como usar credenciais de longo ou curto prazo sem usar o AWS IAM Identity Center.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como AWS IAM Identity Center.

> **Note**
>
> As informações nesses tópicos são para circunstâncias em que você precisa obter e gerenciar manualmente credenciais de curto ou longo prazo. Para obter informações

adicionais sobre credenciais de curto e longo prazo, consulte Outras formas de autenticação no Guia de referência de AWS SDKs e ferramentas.
Para as práticas recomendadas de segurança, use o AWS IAM Identity Center, conforme descrito em Configurar autenticação nas ferramentas.

Avisos e orientações importantes para credenciais

Avisos para credenciais

- NÃO use as credenciais de raiz da sua conta para acessar os recursos da AWS. Estas credenciais fornecem acesso ilimitado à conta e são difíceis de revogar.
- NÃO coloque chaves de acesso literais ou informações de credenciais em comandos ou scripts. Se fizer isso, você corre o risco de expor suas credenciais acidentalmente.
- Esteja ciente de que qualquer credencial armazenada no arquivo compartilhado `credentials` da AWS, é armazenada em texto simples.

Orientação adicional para gerenciar credenciais com segurança

Para ler uma discussão geral sobre como gerenciar credenciais da AWS com segurança, consulte Credenciais de segurança da AWS na Referência geral da AWS e Melhores práticas e casos de uso de segurança no Guia do usuário do IAM. Além dessas discussões, considere o seguinte:

- Crie usuários adicionais, como usuários no Centro de Identidade do IAM, e use as credenciais deles em vez de usar suas credenciais de usuário raiz da AWS. As credenciais de outros usuários poderão ser revogadas, se necessário, ou são temporárias por natureza. Além disso, você pode aplicar uma política a cada usuário para acessar somente determinados recursos e ações e, assim, adotar uma postura de permissões com privilégio mínimo.

- Use perfis do IAM para tarefas do Amazon Elastic Container Service (Amazon ECS).

- Use perfis do IAM para aplicações em execução nas instâncias do Amazon EC2.

Tópicos

- Usar credenciais da AWS
- Credenciais compartilhadas no AWS Tools for PowerShell

Usar credenciais da AWS

Cada comando do AWS Tools for PowerShell deve incluir um conjunto de credenciais da AWS, que são usadas de forma criptográfica para assinar a solicitação de web service correspondente. Você pode especificar credenciais por comando, por sessão ou para todas as sessões.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como AWS IAM Identity Center.

> **Note**
>
> As informações neste tópico são para circunstâncias em que você precisa obter e gerenciar manualmente as credenciais de curto ou longo prazo. Para obter informações adicionais sobre credenciais de curto e longo prazo, consulte Outras formas de autenticação no Guia de referência de AWS SDKs e ferramentas.
> Para as práticas recomendadas de segurança, use o AWS IAM Identity Center, conforme descrito em Configurar autenticação nas ferramentas.

Como melhor prática, para evitar expor suas credenciais, não coloque credenciais literais em um comando. Em vez disso, crie um perfil para cada conjunto de credenciais que você deseja usar e armazene o perfil em um dos dois armazenamentos de credenciais. Especifique o nome correto do perfil no comando e o AWS Tools for PowerShell recuperará as credenciais associadas. Para ver uma discussão geral sobre como gerenciar com segurança as credenciais da AWS, consulte Práticas recomendadas para gerenciar as chaves de acesso da AWS na Referência geral da Amazon Web Services.

> **Note**
>
> Você precisa de uma conta da AWS para obter credenciais e usar o AWS Tools for PowerShell. Para criar uma conta da AWS, consulte Conceitos básicos: você é um usuário iniciante da AWS? no Guia de referência do AWS Account Management.

Tópicos

Locais de armazenamento de credenciais

O AWS Tools for PowerShell pode usar qualquer um dos dois armazenamentos de credenciais.

- O armazenamento de SDKs da AWS, que criptografa suas credenciais e as armazena em sua pasta inicial. No Windows, esse armazenamento está localizado em: `C:\Users\`*`username`*`\AppData\Local\AWSToolkit\RegisteredAccounts.json`.

  O AWS SDK for .NET e o Toolkit for Visual Studio também podem usar o armazenamento do AWS SDK.
- O arquivo de credenciais compartilhadas, que também está localizado na pasta inicial, mas armazena credenciais como texto sem formatação.

  Por padrão, o arquivo de credenciais é armazenado aqui:

  - No Windows: `C:\Users\`*`username`*`\.aws\credentials`
  - No Mac/Linux: `~/.aws/credentials`

  Os SDKs da AWS e a AWS Command Line Interface também podem usar o arquivo de credenciais. Se você estiver executando um script fora de seu contexto de usuário da AWS, certifique-se de que o arquivo que contém suas credenciais seja copiado em um local onde todas as contas de usuário (sistema local e usuário) possam acessar suas credenciais.

Como gerenciar perfis

Os perfis permitem que você faça referência a diferentes conjuntos de credenciais com AWS Tools for PowerShell. Você pode usar os cmdlets do AWS Tools for PowerShell para gerenciar seus perfis no armazenamento do AWS SDK. Você também pode gerenciar perfis no armazenamento do AWS SDK usando o Toolkit for Visual Studio ou programaticamente usando o AWS SDK for .NET.

Para obter instruções sobre como gerenciar perfis no arquivo de credenciais, consulte Práticas recomendadas para gerenciar chaves de acesso da AWS.

Adicionar um novo perfil

Para adicionar um novo perfil para o armazenamento do AWS SDK, execute o comando `Set-AWSCredential`. Ele armazena sua chave de acesso e a chave secreta no arquivo de credenciais padrão sob o nome de perfil especificado.

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

- `-AccessKey`: o ID da chave de acesso.
- `-SecretKey`– a chave secreta.
- `-StoreAs`– o nome do perfil, que deve ser exclusivo. Para especificar o perfil padrão, use o nome `default`.

Atualizar um perfil

O armazenamento da AWS SDK deve ser mantido manualmente. Se você alterar posteriormente as credenciais no serviço, por exemplo, usando o Console do IAM, ocorrerá falha na execução de um comando com as credenciais armazenadas exibindo a seguinte mensagem de erro:

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

Você pode atualizar um perfil repetindo o comando `Set-AWSCredential` para o perfil e passando para ele a novas chaves de acesso e chaves secretas.

Listar perfis

Você pode verificar a lista atual de nomes com o comando a seguir. Nesse exemplo, uma usuária chamada Shirley tem acesso a três perfis que são armazenados no arquivo de credenciais compartilhadas (`~/.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
```

Remover um perfil

Para remover um perfil que você não precisa mais, use o comando a seguir.

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

O parâmetro `-ProfileName` especifica o perfil que você deseja excluir.

O comando obsoleto Clear-AWSCredential ainda está disponível para assegurar a compatibilidade com versões anteriores, mas `Remove-AWSCredentialProfile` é o preferencial.

Especificação de credenciais

Há várias maneiras de especificar credenciais. A melhor maneira é identificar um perfil em vez de incorporar credenciais literais na linha de comando. O AWS Tools for PowerShell localiza o perfil usando uma ordem de pesquisa descrita na Ordem de pesquisa de credenciais.

No Windows, as credenciais da AWS armazenadas no AWS SDK são criptografadas com a identidade de usuário do Windows que fez login. Elas não podem ser descriptografadas usando outra conta, ou usadas em um dispositivo diferente daquele no qual foram criadas originalmente. Para executar tarefas que exigem as credenciais de outro usuário, como uma conta de usuário na qual uma tarefa programada será executada, configure um perfil de credenciais criptografadas, conforme descrito na seção anterior, que você pode usar ao fazer login no computador como esse usuário. Faça login como o usuário que executa tarefas para concluir as etapas de configuração de credenciais e crie um perfil que funcione para esse usuário. Em seguida, faça logout e depois faça login novamente com as suas credenciais para configurar a tarefa agendada.

> **Note**
>
> Use o parâmetro `-ProfileName` comum para especificar um perfil. Esse parâmetro é equivalente ao parâmetro `-StoredCredentials` em versões anteriores do AWS Tools for PowerShell. Para compatibilidade com versões anteriores, o `-StoredCredentials` ainda é suportado.

Perfil padrão (recomendado)

Todos os AWS SDKs e as ferramentas de gerenciamento poderão encontrar suas credenciais automaticamente no computador local se as credenciais estiverem armazenadas em um perfil chamado `default`. Por exemplo, caso você tenha um perfil nomeado como `default` no computador local, não é necessário executar o cmdlet `Initialize-AWSDefaultConfiguration` nem o cmdlet `Set-AWSCredential`. As ferramentas usam automaticamente os dados de chave secreta e chave de acesso. armazenados nesse perfil. Para usar a região da AWS em vez da região padrão (os resultados do `Get-DefaultAWSRegion`), você pode executar `Set-DefaultAWSRegion` e especificar uma região.

Se o perfil não foi nomeado `default`, mas você deseja usá-lo como perfil padrão para a sessão atual, execute `Set-AWSCredential` para defini-lo como perfil padrão.

Embora a execução `Initialize-AWSDefaultConfiguration` permita que você especifique um perfil padrão para cada sessão do PowerShell, o cmdlet carrega as credenciais do seu perfil de nome personalizado, mas substitui o perfil `default` pelo perfil nomeado.

Recomendamos que não executar `Initialize-AWSDefaultConfiguration` a menos que você esteja executando uma sessão do PowerShell em uma instância do Amazon EC2 que não foi iniciada com um perfil de instância e deseje configurar o perfil de credencial manualmente. Observe que, nesse caso, o perfil de credencial não contém credenciais. O perfil de credencial resultante da execução de `Initialize-AWSDefaultConfiguration` em uma instância do EC2 não armazena credenciais diretamente, mas aponta para os metadados da instância (que fornecem credenciais temporárias que são alternadas automaticamente). No entanto, ele armazena a região da instância. Outro cenário que pode exigir a execução de `Initialize-AWSDefaultConfiguration` ocorre se você deseja executar uma chamada em uma região diferente da região em que a instância está em execução. Executar esse comando substitui permanentemente a região armazenada nos metadados da instância.

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

> **Note**
>
> As credenciais padrão são incluídas no armazenamento do AWS SDK com o nome de perfil `default`. O comando substitui qualquer perfil existente por esse nome.

Se a instância do EC2 foi iniciada com um perfil de instância, o PowerShell obtém as informações de credenciais e região da AWS no perfil de instância. Não é necessário executar `Initialize-AWSDefaultConfiguration`. Não é necessário executar o cmdlet `Initialize-AWSDefaultConfiguration` em uma instância do EC2 iniciada com um perfil de instância, pois ele usa os mesmos dados de perfil de instância que o PowerShell já usa por padrão.

Perfil de sessão

Use `Set-AWSCredential` para especificar um perfil padrão para uma sessão específica. Esse perfil substitui todos os perfis padrão durante a sessão. Isso é recomendado se você deseja usar um perfil com nome personalizado em sua sessão em vez do perfil `default` atual.

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

> **Note**
>
> Nas versões do Tools for Windows PowerShell anteriores à 1.1, o cmdlet `Set-AWSCredential` não funcionava corretamente e substituía o perfil especificado por "MyProfileName". Recomendamos utilizar uma versão mais recente do Tools for Windows PowerShell.

Perfil de comando

Em comandos individuais, você pode adicionar o parâmetro `-ProfileName` para especificar um perfil que se aplique apenas a esse comando. Esse perfil substitui todos os perfis padrão ou de sessão, conforme exibido no exemplo a seguir.

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

> **Note**
>
> Quando você especificar um perfil padrão ou de sessão, também poderá adicionar um parâmetro `-Region` para substituir uma região padrão ou de sessão. Para obter mais informações, consulte Especificar AWS regiões. O exemplo a seguir especifica uma região ou um perfil padrão.

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

Por padrão, considera-se que o arquivo de credenciais compartilhadas da AWS esteja na pasta inicial do usuário (`C:\Users\username\.aws` no Windows ou `~/.aws` no Linux). Para especificar um arquivo de credenciais em um local diferente, inclua o parâmetro `-ProfileLocation` e especifique o caminho do arquivo de credenciais. O exemplo a seguir especifica um arquivo de credenciais não padrão para um comando específico.

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

> **Note**
>
> Se você estiver executando um script do PowerShell durante um período em que você normalmente não está conectado à AWS – por exemplo, você está executando um script do PowerShell como uma tarefa programada fora do horário de trabalho normal –, adicione o parâmetro `-ProfileLocation` ao especificar o perfil que deseja usar e defina o valor como o caminho do arquivo que armazena suas credenciais. Para ter certeza de que o seu script do AWS Tools for PowerShell é executado com as credenciais corretas da conta, você deve adicionar o parâmetro `-ProfileLocation` sempre que o script for executado em um contexto ou processo que não use uma conta da AWS. Você também pode copiar o arquivo de credenciais em um local que pode ser acessado no sistema local ou outra conta que seus scripts usem para executar tarefas.

Ordem de pesquisa de credenciais

Quando você executa um comando, o AWS Tools for PowerShell procura credenciais na ordem a seguir. Ele para ao encontrar credenciais utilizáveis.

1. As credenciais literais incorporadas como parâmetros na linha de comando.

   É altamente recomendável usar perfis em vez de colocar as credenciais literais em suas linhas de comando.
2. Um local de perfil ou um nome de perfil especificado.

- Se você especificar apenas um nome de perfil, o comando procura um perfil especificado do armazenamento do AWS SDK e, se esse perfil não existir, o perfil especificado no arquivo de credenciais compartilhadas da AWS no local padrão.
- Se você especificar apenas um local de perfil, o comando procurará o perfil `default` desse arquivo de credenciais.
- Se você especificar um nome e um local, o comando procurará o perfil especificado nesse arquivo de credenciais.

Se o perfil ou o local especificado não for encontrado, o comando lançará uma exceção. A pesquisa passará para as seguintes etapas somente se você não tiver especificado um perfil ou local.

3. Credenciais especificadas pelo parâmetro `-Credential`.
4. O perfil da sessão, se existir.
5. Use um perfil padrão, na seguinte ordem:
   a. O perfil `default` no armazenamento do AWS SDK.
   b. O perfil `default` no arquivo de credenciais compartilhadas da AWS.
   c. O perfil `AWS PS Default` no armazenamento do AWS SDK.
6. Se o comando estiver sendo executado em uma instância do Amazon EC2 configurada para usar uma função do IAM, as credenciais temporárias da instância do EC2 acessadas no perfil da instância.

   Para obter mais informações sobre o uso de funções do IAM, para instâncias do Amazon EC2. consulte AWS SDK for .NET.

Se essa pesquisa não conseguir localizar as credenciais especificadas, o comando lançará uma exceção.

Tratamento de credenciais no AWS Tools for PowerShell Core

Os cmdlets no AWS Tools for PowerShell Core aceitam o acesso da AWS e chaves secretas ou os nomes de perfis de credenciais ao serem executados, de forma semelhante ao AWS Tools for Windows PowerShell. Quando forem executados no Windows, os dois módulos têm acesso ao arquivo de armazenamento de credenciais do AWS SDK for .NET (armazenado no arquivo `AppData\Local\AWSToolkit\RegisteredAccounts.json` por usuário).

Esse arquivo armazena suas chaves em formato criptografado e não pode ser usado em outro computador. Ele é o primeiro arquivo no qual o AWS Tools for PowerShell procura por um perfil de credencial, e também é o arquivo em que o AWS Tools for PowerShell armazena perfis de credenciais. Para obter mais informações sobre o arquivo de armazenamento de credenciais do AWS SDK for .NET, consulte Configuração das credenciais da AWS. O módulo Tools for Windows PowerShell no momento não oferece suporte à gravação de credenciais em outros arquivos ou locais.

Os dois módulos podem ler perfis do arquivo de credenciais compartilhadas da AWS que é usado por outros AWS SDKs e pela AWS CLI. No Windows, o local padrão para esse arquivo é `C:\Users\<userid>\.aws\credentials`. Em plataformas diferentes do Windows, esse arquivo é armazenado em `~/.aws/credentials`. O parâmetro `-ProfileLocation` pode ser usado para apontar para um nome de arquivo padrão ou local do arquivo.

O armazenamento de credenciais de SDKs mantém suas credenciais no formato criptografado usando as APIs criptografadas do Windows. Essas APIs não estão disponíveis em outras plataformas, de modo que o módulo AWS Tools for PowerShell Core usa o arquivo de credenciais compartilhadas da AWS de forma exclusiva e oferece suporte à gravação de novos perfis de credencial no arquivo de credenciais compartilhadas.

Os scripts de exemplo a seguir que usam o cmdlet `Set-AWSCredential` mostram as opções para lidar com perfis de credenciais no Windows com os módulos AWSPowerShell ou 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
```

Os exemplos a seguir mostram o comportamento do módulo AWSPowerShell.NetCore nos sistemas operacionais Linux ou 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
```

Credenciais compartilhadas no AWS Tools for PowerShell

O Tools for Windows PowerShell oferece suporte ao uso do arquivo de credenciais compartilhadas da AWS, de forma semelhante à AWS CLI e a outros AWS SDKs. O Tools for Windows PowerShell agora oferece suporte à leitura e à gravação dos perfis de credenciais `basic`, `session` e `assume role` no arquivo de credenciais do .NET e no arquivo de credenciais compartilhadas da AWS. Essa funcionalidade é ativada por um novo namespace `Amazon.Runtime.CredentialManagement`.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como AWS IAM Identity Center.

> **Note**
>
> As informações neste tópico são para circunstâncias em que você precisa obter e gerenciar manualmente as credenciais de curto ou longo prazo. Para obter informações adicionais sobre credenciais de curto e longo prazo, consulte Outras formas de autenticação no Guia de referência de AWS SDKs e ferramentas.
> Para as práticas recomendadas de segurança, use o AWS IAM Identity Center, conforme descrito em Configurar autenticação nas ferramentas.

Os novos tipos de perfil e o acesso ao arquivo de credenciais compartilhadas da AWS são compatíveis com os seguintes parâmetros que foram adicionados aos cmdlets relacionados a credenciais: Initialize-AWSDefaultConfiguration, New-AWSCredential e Set-AWSCredential. Nos cmdlets de serviço, é possível fazer referência ao seus perfis adicionando o parâmetro comum, `-ProfileName`.

Uso de uma função de IAM com o AWS Tools for PowerShell

O arquivo de credenciais compartilhadas da AWS permite tipos adicionais de acesso. Por exemplo, é possível acessar seus recursos da AWS usando uma função do IAM em vez das credenciais de longo prazo de um usuário do IAM. Para fazer isso, é necessário ter um perfil padrão que tenha permissões para assumir a função. Quando você instrui o AWS Tools for PowerShell para usar um perfil que especificou uma função, o AWS Tools for PowerShell procura o perfil identificado pelo parâmetro `SourceProfile`. Essas credenciais são usadas para solicitar credenciais temporárias para a função especificada pelo parâmetro `RoleArn`. Opcionalmente, é possível exigir o uso de um dispositivo de autenticação multifator (MFA) ou de um código de `ExternalId` quando a função é assumida por terceiros.

Nome do parâmetro	Descrição
ExternalId	O ID externo definido pelo usuário a ser usado ao assumir uma função, se for necessário para a função. Normalmente, isso só é necessário quando você delega o acesso à sua conta a terceiros. O terceiro deverá incluir o ExternalID como parâmetro ao assumir a função atribuída . Para obter mais informações, consulte Como

Nome do parâmetro	Descrição
	usar um ID externo ao conceder acesso aos recursos da AWS para terceiros no Manual do usuário do IAM.
MfaSerial	O número de série MFA a ser usado ao assumir uma função, se for necessário para a função. Para obter mais informações, consulte Uso da autenticação multifator (MFA) na AWS no Manual do ujsuário do IAM.
RoleArn	O ARN da função a ser assumida para credenciais assume role. Para obter mais informações sobre como criar e usar funções do IAM, consulte Funções do IAM no Manual do usuário do IAM.
SourceProfile	O nome do perfil de origem a ser usado pelas credenciais assume role. As credenciais encontradas neste perfil são usadas para assumir a função especificada pelo parâmetro `RoleArn`.

Configuração de perfis para assumir uma função

Veja a seguir um exemplo que mostra como configurar um perfil de origem que permite assumir diretamente uma função do IAM.

O primeiro comando cria um perfil de origem que é referenciado pelo perfil de função. O segundo comando cria o perfil de função a ser assumido pela função. O terceiro comando mostra as credenciais para o perfil de função.

```
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 usar esse perfil de função com os cmdlets de serviço do Tools for Windows PowerShell, adicione o parâmetro comum `-ProfileName` ao comando para fazer referência ao perfil de função. O exemplo a seguir usa o perfil de função definido no exemplo anterior para acessar o cmdlet Get-S3Bucket. O AWS Tools for PowerShell procura as credenciais em `my_source_profile`, usa essas credenciais para chamar `AssumeRole` em nome do usuário e usa essas credenciais de função temporárias para chamar `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 dos tipos de perfil de credencial

Para definir um tipo de perfil de credencial, entenda quais parâmetros fornecem as informações necessárias ao tipo de perfil.

Tipo de credenciais	Parâmetros que você deve usar

Basic
 	
Estas são as credenciais de longo prazo para um usuário do IAM
	`-AccessKey`
	
	`-SecretKey`

Sessão:
 	
Essas são as credenciais de curto prazo para uma função do IAM que você recupera manualmente, por exemplo, chamando diretamente o cmdlet Use-STRole.
	`-AccessKey`
	
	`-SecretKey`
	
	`-SessionToken`

Tipo de credenciais	Parâmetros que você deve usar

Função:
 	
Estas são credenciais de curto prazo para uma função do IAM que o AWS Tools for PowerShell recupera para você.
	`-SourceProfile`
	
	`-RoleArn`
	
	opcional: `-ExternalId`
	
	opcional: `-MfaSerial`

O parâmetro comum **`ProfilesLocation`**

Você pode usar `-ProfileLocation` para gravar no arquivo de credenciais compartilhadas, bem como instruir um cmdlet para ler o arquivo de credenciais. A inclusão do parâmetro `-ProfileLocation` determina se o Tools for Windows PowerShell usa o arquivo de credenciais compartilhadas ou o arquivo de credenciais do .NET. A tabela a seguir descreve como o parâmetro funciona no Tools for Windows PowerShell.

Valor do local do perfil	Comportamento da resolução do perfil
nulo (não definido) ou vazio	Primeiro, pesquise o arquivo de credenciais do .NET para um perfil com o nome especificado. Se o perfil não for encontrado, pesquise o arquivo de credenciais compartilhadas da AWS em *`(user's home directory)`* `\.aws\credentials` .
O caminho para um arquivo no formato de arquivo de credenciais compartilhadas da AWS	Pesquise apenas o arquivo especificado para um perfil com o nome fornecido.

Salvar credenciais em um arquivo de credenciais

Para gravar e salvar as credenciais em um dos dois arquivos de credenciais, execute o cmdlet `Set-AWSCredential`. O exemplo a seguir mostra como fazer isso. O primeiro comando usa `Set-AWSCredential` com `-ProfileLocation` para adicionar chaves de acesso e secretas a um perfil especificado pelo parâmetro `-ProfileName`. Na segunda linha, execute o cmdlet Get-Content para exibir o conteúdo do arquivo de credenciais.

```
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
```

Exibir seus perfis de credenciais

Execute o cmdlet Get-AWSCredential e adicione o parâmetro `-ListProfileDetail` para retornar tipos de arquivos de credenciais e locais, bem como uma lista de nomes de perfil.

```
PS > Get-AWSCredential -ListProfileDetail

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

Remoção de perfis de credencial

Para remover perfis de credencial, execute o novo cmdlet Remove-AWSCredentialProfile. Clear-AWSCredential está obsoleto, mas ainda está disponível para compatibilidade com versões anteriores.

Observações importantes

Somente Initialize-AWSDefaultConfiguration, New-AWSCredential e Set-AWSCredential oferecem suporte aos parâmetros para perfis de função. Não é possível especificar os parâmetros de função diretamente em um comando, como `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`. Isso não funciona porque os cmdlets de serviço não oferecem suporte direto aos parâmetros `SourceProfile` ou `RoleArn`. Em vez disso, armazene esses parâmetros em um perfil chame o comando com o parâmetro `-ProfileName`.

Trabalhar com serviços da AWS no AWS Tools for PowerShell

Esta seção fornece exemplos de uso do AWS Tools for PowerShell para acessar serviços da AWS. Esses exemplos ajudam a demonstrar como usar os cmdlets para executar tarefas da AWS reais. Esses exemplos se baseiam nos cmdlets fornecidos pelas Ferramentas para PowerShell. Para ver quais cmdlets estão disponíveis, consulte a Referência do cmdlet do AWS Tools for PowerShell.

Codificação de concatenação de arquivo do PowerShell

Alguns cmdlets no AWS Tools for PowerShell editam os arquivos ou registros existentes que você tem na AWS. Um exemplo é `Edit-R53ResourceRecordSet`, que chama a API ChangeResourceRecordSets para o Amazon Route 53.

Quando você edita ou concatena arquivos no PowerShell 5.1 ou versões anteriores, o PowerShell codifica a saída em UTF-16, não em UTF-8. Isso pode adicionar caracteres indesejados e criar resultados que não são válidos. Um editor hexadecimal pode revelar os caracteres indesejados.

Para evitar a conversão da saída do arquivo para UTF-16, é possível redirecionar seu comando para o cmdlet `Out-File` do PowerShell e especificar a codificação UTF-8, conforme mostrado no exemplo a seguir:

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

Se você estiver executando comandos da AWS CLI no console do PowerShell, o mesmo comportamento se aplicará. É possível redirecionar a saída de um comando da AWS CLI para `Out-File` no console do PowerShell. Outros cmdlets, como o `Export-Csv` ou o `Export-Clixml`, também têm um parâmetro `Encoding`. Para obter uma lista completa de cmdlets que tenham um parâmetro `Encoding` e que permitam que você corrija a codificação da saída de um arquivo concatenado, execute o comando a seguir:

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

> **Note**
>
> O PowerShell 6.0 e mais recentes, incluindo o PowerShell Core, retém automaticamente a codificação UTF-8 para a saída de arquivos concatenados.

Objetos retornados para as ferramentas do Powershell

Para tornar o AWS Tools for PowerShell mais útil em um ambiente nativo do PowerShell, o objeto retornado por um cmdlet do AWS Tools for PowerShell é um objeto.NET, não o objeto de texto JSON que normalmente é retornado da API correspondente no AWS SDK. Por exemplo, `Get-S3Bucket` emite uma coleção `Buckets`, não um objeto de resposta JSON do Amazon S3. A coleção `Buckets` pode ser colocada no pipeline do PowerShell e é possível interagir com ela de maneiras apropriadas. Da mesma forma, `Get-EC2Instance` emite uma coleção de objetos .NET `Reservation`, não um objeto de resultado JSON `DescribeEC2Instances`. Esse comportamento é por design e permite que a experiência do AWS Tools for PowerShell seja mais consistente com o PowerShell idiomático.

As respostas do serviço real estão disponíveis se você precisar delas. Elas são armazenadas como propriedades `note` nos objetos retornados. Para ações de API que ofereçam suporte à paginação usando campos `NextToken`, eles também são anexados como propriedades `note`.

Amazon EC2

Esta seção aborda as etapas necessárias para executar uma instância do Amazon EC2, incluindo como:

- Recuperar uma lista de Imagens de máquina da Amazon (AMIs).
- Criar um par de chaves para autenticação SSH.
- Crie e configure um grupo de segurança do Amazon EC2.
- Executar a instância e recuperar informações sobre ela.

Amazon S3

A seção aborda as etapas necessárias para criar um site estático hospedado no Amazon S3. Ela demonstra como:

- Criar e excluir buckets do Amazon S3.

- Fazer upload de arquivos para um bucket do Amazon S3 na forma de objetos.
- Excluir objetos de um bucket do Amazon S3.
- Designar um bucket do Amazon S3 como um site.

AWS Lambda e AWS Tools for PowerShell

Esta seção fornece uma breve visão geral do módulo AWS Lambda Tools for PowerShell e descreve as etapas necessárias para configurar o módulo.

Amazon SNS e Amazon SQS

Esta seção aborda as etapas necessárias para inscrever uma fila do Amazon SQS em um tópico do Amazon SNS. Ela demonstra como:

- Crie um tópico do Amazon SNS.
- Crie uma fila do Amazon SQS.
- Inscrever a fila no tópico do .
- Envie uma mensagem para o tópico.
- Receba a mensagem da fila.

CloudWatch

Esta seção fornece um exemplo de como publicar dados personalizados no CloudWatch.

- Publicar uma métrica personalizada no seu painel do CloudWatch.

Consulte também

- Conceitos básicos da AWS Tools for Windows PowerShell

Tópicos

- Amazon S3 e Tools for Windows PowerShell
- Amazon EC2 e Tools for Windows PowerShell

- AWS Lambda e AWS Tools for PowerShell
- Amazon SQS, Amazon SNS e Tools for Windows PowerShell
- CloudWatch do AWS Tools for Windows PowerShell
- Usar o parâmetro ClientConfig em cmdlets

Amazon S3 e Tools for Windows PowerShell

Nesta seção, criamos um site estático com o AWS Tools for Windows PowerShell usando o Amazon S3 e o CloudFront. No processo, demonstramos uma série de tarefas comuns com esses serviços. Essa demonstração é modelada com base no Guia de conceitos básicos para Hospedar um site estático, que descreve um processo semelhante usando o Console de gerenciamento da AWS.

Os comandos mostrados aqui supõem que você tenha definido credenciais padrão e uma região padrão para a sua sessão do PowerShell. Portanto, credenciais e regiões não estão incluídas na chamada dos cmdlets.

> **Note**
>
> No momento, não há nenhuma API do Amazon S3 para renomear um bucket ou um objeto e, portanto, nenhum cmdlet específico do Tools for Windows PowerShell para executar esta tarefa. Para renomear um objeto no S3, recomendamos copiar o objeto usando um novo nome, executando o cmdlet Copy-S3Object e excluir o objeto original, executando o cmdlet Remove-S3Object.

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- Hospedagem de um site estático no Amazon S3
- Console do Amazon S3

Tópicos

- Criar um bucket do Amazon S3, verificar sua região e, opcionalmente, removê-lo
- Configurar um bucket do Amazon S3 como um site e ativar o registro em log
- Fazer upload de objetos para um bucket do Amazon S3

- Excluir objetos e buckets do Amazon S3
- Upload de conteúdo de texto em linha para o Amazon S3

Criar um bucket do Amazon S3, verificar sua região e, opcionalmente, removê-lo

Use o cmdlet `New-S3Bucket` para criar um novo bucket do Amazon S3. Os exemplos a seguir criam um bucket chamado `website-example`. O nome do bucket deve ser globalmente exclusivo em todas as regiões. O exemplo cria o bucket na região `us-west-1`.

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

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

Você pode verificar a região em que o bucket está localizado usando o cmdlet `Get-S3BucketLocation`.

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

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

Quando terminar este tutorial, você poderá usar a linha a seguir para remover esse bucket. Sugerimos que você deixe esse bucket no lugar, pois o usaremos em exemplos subsequentes.

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

Observe que o processo de remoção do bucket leva algum tempo para ser concluído. Se você tentar recriar um bucket com o mesmo nome imediatamente, poderá haver falha no cmdlet `New-S3Bucket` até que o antigo tenha sido removido completamente.

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- Put bucket (Referência de serviços do Amazon S3)

- Regiões do AWS PowerShell para Amazon S3

Configurar um bucket do Amazon S3 como um site e ativar o registro em log

Use o cmdlet `Write-S3BucketWebsite` para configurar um bucket do Amazon S3 como um site estático. O exemplo a seguir especifica um nome `index.html` para a página da web de conteúdo padrão e um nome `error.html` para a página da web de erro padrão. Observe que esse cmdlet não cria essas páginas. Elas precisam ser carregadas como objetos do 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    :
```

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- Site Put bucket (Referência de APIs do Amazon S)
- ACL Put bucket (Referência de APIs do Amazon S)

Fazer upload de objetos para um bucket do Amazon S3

Use o cmdlet `Write-S3Object` para fazer upload de arquivos do seu sistema de arquivos local para um bucket do Amazon S3 como objetos. O exemplo a seguir cria e carrega dois arquivos HTML simples para um bucket do Amazon S3 e verifica a existência dos objetos carregados. O parâmetro `-File` para `Write-S3Object` especifica o nome do arquivo no sistema de arquivos local. O parâmetro `-Key` especifica o nome que o objeto correspondente terá no Amazon S3.

A Amazon deduz o tipo de conteúdo dos objetos a partir das extensões de arquivos, nesse 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
```

Opções pré-configuradas de ACL

Os valores para especificar ACLs pré-configuradas com o Tools for Windows PowerShell são os mesmos que os usados pelo AWS SDK for .NET. Observe, no entanto, que eles são diferentes dos valores usados pela ação `Put Object` do Amazon S3. O Tools for Windows PowerShell oferece suporte às seguintes ACLs pré-configuradas:

- NoACL

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

Para obter mais informações sobre essas configurações de ACL pré-configurada, consulte Visão geral da lista de controle de acesso.

Observação sobre multipart upload

Se você usar a API do Amazon S3 para fazer upload de um arquivo com mais de 5 GB, será necessário utilizar o carregamento fracionado. No entanto, o cmdlet `Write-S3Object` fornecido pelo Tools for Windows PowerShell pode tratar de maneira transparente uploads de arquivos com mais de 5 GB.

Testar o site

Nesse momento, você pode testar o site, navegando até ele com um navegador. Os URLs para sites estáticos hospedados no Amazon S3 seguem um formato padrão.

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

Por exemplo:

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

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- Objeto Put (Referência de APIs do Amazon S)
- ACLs pré-configuradas (Referência de APIs do Amazon S)

Excluir objetos e buckets do Amazon S3

Esta seção descreve como excluir o site que você criou nas seções anteriores. Você pode simplesmente excluir os objetos para os arquivos HTML e, em seguida, excluir o bucket do Amazon S3 para o site.

Primeiramente, execute o cmdlet `Remove-S3Object` para excluir os objetos dos arquivos HTML do bucket do Amazon S3.

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

A resposta `False` é um artefato esperado da forma como o Amazon S3 processa a solicitação. Neste contexto, ela não indica um problema.

Agora, é possível executar o cmdlet `Remove-S3Bucket` para excluir o bucket do Amazon S3 vazio do site.

```
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    :
```

Na versão 1.1 e em versões mais recentes do AWS Tools for PowerShell, é possível adicionar o parâmetro `-DeleteBucketContent` ao `Remove-S3Bucket`, que primeiro exclui todos os objetos e as versões do objeto no bucket especificado antes de tentar remover o bucket. Dependendo do número de objetos ou versões de objetos no bucket, essa operação pode demorar um intervalo substancial de tempo. Nas versões do Tools for Windows PowerShell anteriores à 1.1, era necessário que o bucket estivesse vazio para que o `Remove-S3Bucket` pudesse excluí-lo.

> **Note**
>
> A menos que o parâmetro `-Force` seja adicionado, o AWS Tools for PowerShell solicitará confirmação antes que o cmdlet seja executado.

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- Excluir objeto (Referência de APIs do Amazon S)
- Excluir bucket (Referência de APIs do Amazon S)

Upload de conteúdo de texto em linha para o Amazon S3

O cmdlet `Write-S3Object` oferece suporte à capacidade de fazer upload de conteúdo de texto em linha para o Amazon S3. Usando o parâmetro `-Content` (alias `-Text`), você pode especificar o conteúdo baseado em texto que deve ser carregados para o Amazon S3 sem a necessidade de colocá-lo em um arquivo primeiro. O parâmetro aceita sequências de uma linha simples, bem como strings que contêm várias linhas.

```
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 e Tools for Windows PowerShell

É possível executar tarefas comuns relacionadas ao Amazon EC2 usando o AWS Tools for PowerShell.

Os comandos de exemplo mostrados aqui supõem que você tenha definido credenciais padrão e uma região padrão para a sua sessão do PowerShell. Portanto, não incluímos credenciais nem região quando chamamos o cmdlets. Para obter mais informações, consulte . Conceitos básicos da AWS Tools for Windows PowerShell.

Tópicos

- Criação de um par de chaves
- Criar um grupo de segurança usando o Windows PowerShell
- Encontrar uma Imagem de máquina da Amazon usando o Windows PowerShell
- Iniciar uma instância do Amazon EC2 usando o Windows PowerShell

Criação de um par de chaves

O exemplo `New-EC2KeyPair` a seguir cria um par de chaves e as armazena na variável `$myPSKeyPair` do PowerShell

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

Insira o objeto do par de chaves no cmdlet `Get-Member` para visualizar a estrutura do 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;}
```

Insira o objeto do par de chaves no cmdlet `Format-List` para visualizar os valores dos membros `KeyName`, `KeyFingerprint` e `KeyMaterial`. (A saída foi truncada para facilitar a leitura.)

```
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-----
```

O membro `KeyMaterial` armazena a chave privada do par de chaves. A chave pública é armazenada na AWS. Você não pode recuperar a chave pública da AWS, mas pode verificar a chave pública ao comparar o `KeyFingerprint` para a chave privada ao retornado pela AWS para a chave pública.

Exibição da impressão digital do seu par de chaves

Você pode usar o cmdlet `Get-EC2KeyPair` para visualizar a impressão digital para o seu par de chaves.

```
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
```

Armazenamento de sua chave privada

Para armazenar a chave privada em um arquivo, insira o membro `KeyFingerMaterial` no cmdlet `Out-File`.

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

Você deve especificar `-Encoding ascii` ao gravar a chave privada em um arquivo. Caso contrário, talvez as ferramentas, como `openssl`, não consigam ler o arquivo corretamente. Você pode verificar se o formato do arquivo resultante está correto usando um comando como o seguinte:

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

(A ferramenta `openssl` não está incluída no AWS Tools for PowerShell nem no AWS SDK for .NET.)

Exclusão do par de chaves

Você precisará de seu par de chaves para executar e conectar-se a uma instância. Depois que terminar de usar um par de chaves, você poderá removê-lo. Para remover a chave pública da AWS, use o cmdlet `Remove-EC2KeyPair`. Quando solicitado, pressione `Enter` para remover o par de chaves.

```
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"):
```

A variável, `$myPSKeyPair`, ainda existe na sessão atual do PowerShell atual e ainda contém informações do par de chaves. O arquivo `myPSKeyPair.pem` também existe. No entanto, a chave

privada não é mais válida porque a chave pública para o par de chaves não é mais armazenada na AWS.

Criar um grupo de segurança usando o Windows PowerShell

Use o AWS Tools for PowerShell para criar e configurar um grupo de segurança. Ao criar um grupo de segurança, você especifica se é para o EC2-Classic ou o EC2-VPC. A resposta é o ID do grupo de segurança.

Se você precisar se conectar à sua instância, deverá configurar o grupo de segurança para permitir o tráfego SSH (Linux) ou RDP (Windows).

Tópicos

Pré-requisitos

Você precisará do endereço IP público do computador, na notação CIDR. Também é possível acessar um endereço IP público de seu computador local por meio de um serviço. Por exemplo, a Amazon fornece o seguinte serviço: http://checkip.amazonaws.com/ ou https://checkip.amazonaws.com/. Para localizar outro serviço que forneça o endereço IP, use a frase de busca "qual é o meu endereço IP". Se estiver conectado por meio de um ISP ou protegido por um firewall sem um endereço IP estático, localize o intervalo de endereços IP que pode ser usado pelos computadores cliente.

> **Warning**
>
> Se especificar `0.0.0.0/0`, você habilitará o tráfego de qualquer endereço IP no mundo. Para os protocolos SSH e RDP, essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, certifique-se de autorizar o acesso somente a partir do endereço IP individual apropriado ou do intervalo de endereços.

Criação de um grupo de segurança para o EC2-Classic

> **Warning**
>
> Estamos aposentando o EC2-Classic em 15 de agosto de 2022. Recomendamos que você migre do EC2-Classic para uma VPC. Para obter mais informações, consulte Migrar do EC2-Classic para uma VPC no Guia do usuário das instâncias do Linux do Amazon EC2 ou no Guia do usuário das instâncias do Windows do Amazon EC2. Consulte também a publicação do blog EC2-Classic Networking is Retiring - Here's How to Prepare (O EC2-Classic está sendo retirado: veja como se preparar).

O exemplo a seguir usa o cmdlet `New-EC2SecurityGroup` para criar um grupo de segurança para o `EC2-Classic`.

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

sg-0a346530123456789
```

Para exibir a configuração inicial do grupo de segurança, use o 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 o grupo de segurança a fim de permitir o tráfego de entrada nas portas TCP 22 (SSH) e TCP 3389, use o cmdlet `Grant-EC2SecurityGroupIngress`. Por exemplo, o script de exemplo a seguir mostra como habilitar o tráfego SSH de um único endereço 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 se o grupo de segurança foi atualizado, execute novamente o cmdlet `Get-EC2SecurityGroup`. Observe que não é possível especificar uma regra de saída para o `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 visualizar a regra do grupo de segurança, use a propriedade `IpPermissions`.

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

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

Criação de um grupo de segurança para EC2-VPC

O exemplo `New-EC2SecurityGroup` a seguir adiciona o parâmetro `-VpcId` para criar um grupo de segurança para a VPC especificada.

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

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

Para exibir a configuração inicial do grupo de segurança, use o cmdlet `Get-EC2SecurityGroup`. Por padrão, o grupo de segurança de uma VPC contém uma regra de saída que permite todo o tráfego de saída. Não é possível fazer referência a um grupo de segurança do EC2-VPC por nome.

```
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 as permissões para tráfego de entrada na porta TCP 22 (SSH) e na porta TCP 3389, use o cmdlet `New-Object`. O script de exemplo a seguir define permissões para as portas TCP 22 e 3389 de um único endereço 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 se o grupo de segurança foi atualizado, use novamente o 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 exibir as regras de entrada, recupere a propriedade `IpPermissions` do objeto de coleção retornado pelo 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}
```

Encontrar uma Imagem de máquina da Amazon usando o Windows PowerShell

Ao iniciar uma instância do Amazon EC2, você deve especificar uma Imagem de máquina da Amazon (AMI) para servir como modelo para a instância. No entanto, os IDs de AMIs do Windows da AWS são alterados frequentemente porque a AWS fornece novas AMIs com as atualizações e aprimoramentos de segurança mais recentes. É possível usar os cmdlets Get-EC2Image e Get-EC2ImageByName para encontrar as AMIs atuais do Windows e obter seus IDs.

Tópicos

- Get-EC2Image
- Get-EC2ImageByName

Get-EC2Image

O cmdlet `Get-EC2Image` recupera uma lista de AMIs que você pode usar.

Use o parâmetro `-Owner` com o valor de matriz `amazon, self` para que `Get-EC2Image` recupere apenas as AMIs que pertencem à Amazon ou a você. Nesse contexto, você se refere ao usuário cujas credenciais foram usadas para invocar o cmdlet.

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

Você pode limitar os resultados usando o parâmetro `-Filter`. Para especificar o filtro, crie um objeto do tipo `Amazon.EC2.Model.Filter`. Por exemplo, use o filtro a seguir para exibir somente AMIs do 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
```

Veja a seguir um exemplo de uma das AMIs retornadas pelo cmdlet; a saída real do comando anterior fornece informações para muitas AMIs.

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

```
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

Get-EC2ImageByName

O cmdlet `Get-EC2ImageByName` permite que você filtre a lista de AMIs do Windows da AWS com base no tipo de configuração de servidor na qual você está interessado.

Quando executado sem parâmetros, conforme a seguir, o cmdlet emite o conjunto completo de nomes de filtros atuais:

```
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 limitar o conjunto de imagens retornadas, especifique um ou mais nomes de filtro usando o parâmetro `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
```

Iniciar uma instância do Amazon EC2 usando o Windows PowerShell

Para executar uma instância do Amazon EC2, será necessário usar o par de chaves e o grupo de segurança criados nas seções anteriores. Você também precisa do ID de uma Imagem de máquina da Amazon (AMI). Para obter mais informações, consulte a documentação a seguir:

- Criação de um par de chaves
- Criar um grupo de segurança usando o Windows PowerShell
- Encontrar uma Imagem de máquina da Amazon usando o Windows PowerShell

> **Important**
>
> Se você executar uma instância que não esteja no nível gratuito, será faturado depois do início da instância e cobrado pelo tempo que executou a instância, mesmo se ela permanecer ociosa.

Tópicos

Execução de uma instância no EC2-Classic

> **Warning**
>
> Estamos aposentando o EC2-Classic em 15 de agosto de 2022. Recomendamos que você migre do EC2-Classic para uma VPC. Para obter mais informações, consulte Migrar do EC2-Classic para uma VPC no Guia do usuário das instâncias do Linux do Amazon EC2 ou no Guia do usuário das instâncias do Windows do Amazon EC2. Consulte também a publicação do blog EC2-Classic Networking is Retiring - Here's How to Prepare (O EC2-Classic está sendo retirado: veja como se preparar).

O comando a seguir cria e executa uma única instância `t1.micro`.

```
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       : {}
```

A instância estará no estado `pending` inicialmente, mas passará para o estado `running` depois de alguns minutos. Para visualizar informações sobre sua instância, use o cmdlet `Get-EC2Instance`. Se você tiver mais de uma instância, poderá filtrar os resultados no ID de reserva usando o parâmetro `Filter`. Primeiro, crie um objeto do tipo `Amazon.EC2.Model.Filter`. Depois, chame `Get-EC2Instance`, que usa o filtro e exibe a propriedade `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                 :
```

Execução de uma instância em uma VPC

O comando a seguir cria uma única instância `m1.small` na sub-rede privada especificada. O grupo de segurança deve ser válido para a sub-rede 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       : {}
```

A instância estará no estado `pending` inicialmente, mas passará para o estado `running` depois de alguns minutos. Para visualizar informações sobre sua instância, use o cmdlet `Get-EC2Instance`. Se você tiver mais de uma instância, poderá filtrar os resultados no ID de reserva usando o parâmetro `Filter`. Primeiro, crie um objeto do tipo `Amazon.EC2.Model.Filter`. Depois, chame `Get-EC2Instance`, que usa o filtro e exibe a propriedade `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
```

Execução de uma instância spot em uma VPC

O script de exemplo a seguir solicita uma Instância spot na sub-rede especificada. O grupo de segurança deve ser um que você criou para o VPC que contém a sub-rede 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 e AWS Tools for PowerShell

Com o módulo AWSLambdaPSCore, é possível desenvolver funções AWS Lambda no PowerShell Core 6.0 usando o tempo de execução do .NET Core 2.1. Os desenvolvedores do PowerShell podem gerenciar os recursos da AWS e gravar scripts de automação no ambiente do PowerShell usando o Lambda. O suporte ao PowerShell no Lambda permite executar scripts do PowerShell ou funções do Lambda em resposta a um evento, como um evento do Amazon S3 ou um evento programado do Amazon CloudWatch. O módulo AWSLambdaPSCore é um módulo da AWS separado para PowerShell; ele não é parte do AWS Tools for PowerShell nem a instalação do módulo AWSLambdaPSCore instala o AWS Tools for PowerShell.

Após instalar o módulo AWSLambdaPSCore, você pode usar quaisquer cmdlets do PowerShell disponíveis ou desenvolver seu próprios cmdlets para criar funções sem servidor. O módulo AWS

Lambda Tools for PowerShell inclui modelos de projeto para aplicações sem servidor baseadas em PowerShell e as ferramentas para publicar projetos na AWS.

Suporte ao módulo AWSLambdaPSCore está disponível em todas as regiões compatíveis com o Lambda. Para obter mais informações sobre as regiões com suporte, consulte Tabela de regiões da AWS.

Pré-requisitos

As etapas a seguir são necessárias antes que você possa instalar e usar o módulo do AWSLambdaPSCore. Para obter mais detalhes sobre essas etapas, consulte Configuração de um ambiente de desenvolvimento do PowerShell no Guia do desenvolvedor do AWS Lambda.

- Instale a versão correta do PowerShell - o suporte do Lambda para PowerShell é baseado na plataforma cruzada do PowerShell Core versão 6.0. Você pode desenvolver funções do Lambda em PowerShell no Windows, Linux ou Mac. Se você não tiver pelo menos essa versão do PowerShell instalada, instruções estarão disponíveis no site de documentação do Microsoft PowerShell.
- Instalar o .NET Core 2.1 SDK: como PowerShell Core é baseado no .NET Core, o suporte do Lambda ao PowerShell usa o mesmo tempo de execução do Lambda do .NET Core 2.1 para as funções do Lambda tanto no .NET Core quanto no PowerShell. Os cmdlets de publicação do PowerShell para Lambda usam o .NET Core 2.1 SDK para criar o pacote de implantação do Lambda. O .NET Core 2.1 SDK está disponível na Central de download da Microsoft. Certifique-se de instalar o SDK, e não o Runtime.

Instale o módulo do AWSLambdaPSCore

Após concluir os pré-requisitos, você está pronto para instalar o módulo AWSLambdaPSCore. Execute o comando a seguir na seção do PowerShell Core.

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

Você está pronto para começar a desenvolver funções do Lambda no PowerShell. Para obter mais informações sobre como começar, consulte Programar modelo para criar funções Lambda no PowerShell no Guia de desenvolvedor do AWS Lambda.

Consulte também

- Anúncio do suporte ao Lambda para PowerShell Core no blog do desenvolvedor da AWS
- Módulo de AWSLambdaPSCore no site de galeria do PowerShell
- Definindo um ambiente de desenvolvimento do PowerShell
- Ferramentas do AWS Lambda para PowerShell no GitHub
- Console do AWS Lambda

Amazon SQS, Amazon SNS e Tools for Windows PowerShell

Esta seção fornece exemplos que mostram como:

- Crie uma fila do Amazon SQS e obtenha o nome do recurso da Amazon (ARN) da fila.
- Crie um tópico do Amazon SNS.
- Fornecer permissões ao tópico do SNS para que ele possa enviar mensagens à fila.
- Assinar a fila para o tópico do SNS
- Forneça aos usuários do IAM ou às contas da AWS permissões para publicar no tópico do SNS e ler mensagens da fila do SQS.
- Verificar resultados publicando uma mensagem no tópico e lendo a mensagem da fila.

Crie uma fila do Amazon SQS e obtenha o nome do recurso da Amazon (ARN)

O comando a seguir cria uma fila do SQS em sua região padrão. A saída mostra o URL da nova fila.

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

O comando a seguir recupera o ARN da fila.

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

Criar um tópico do Amazon SNS

O comando a seguir cria um tópico SNS em sua região padrão e retorna o ARN do novo tópico.

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

Fornecer permissões ao tópico do SNS

O script de exemplo a seguir cria uma fila do SQS e um tópico SNS e concede permissões ao tópico SNS para que ele possa enviar mensagens para a fila do 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 }
```

Assinar a fila para o tópico do SNS

O comando a seguir inscreve a fila `myQueue` no tópico SNS `myTopic` e retorna o ID da inscrição:

```
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
```

Fornecer permissões

O comando a seguir fornece a permissão para executar a ação `sns:Publish` no tópico `myTopic`

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

O comando a seguir fornece a permissão para executar as ações `sqs:ReceiveMessage` e `sqs:DeleteMessage` na fila `myQueue`

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

Verificar os resultados

O comando a seguir testa sua nova fila e tópico publicando uma mensagem no tópico SNS `myTopic` e retorna o `MessageId`.

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

O comando a seguir recupera a mensagem da fila `myQueue` do SQS e a exibe.

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

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

CloudWatch do AWS Tools for Windows PowerShell

Esta seção mostra um exemplo de como usar o Tools for Windows PowerShell para publicar dados de métrica personalizados para o CloudWatch.

Neste exemplo, que você tenha definido credenciais padrão e uma região padrão para a sua sessão do PowerShell.

Publicar uma métrica personalizada no seu painel do CloudWatch

O código do PowerShell a seguir inicializa um objeto `MetricDatum` do CloudWatch e o publica no serviço. Você pode ver o resultado dessa operação acessando o console do 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
```

Observe o seguinte:

- As informações de data/hora usadas para inicializar `$dat.Timestamp` devem estar no Horário Universal (UTC).
- O valor usado para inicializar `$dat.Value` pode ser de um valor de string entre aspas ou um valor numérico (sem aspas). O exemplo mostra um valor de string.

Consulte também

- Trabalhar com serviços da AWS no AWS Tools for PowerShell
- https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/CloudWatch/MCloudWatchPutMetricDataPutMetricDataRequest.htmlAmazonCloudWatchClient.PutMetricData (Referência de SDK do .NET)
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_MetricDatum.htmlMetricDatum (Referência de APIs de serviços)
- Console do Amazon CloudWatch

Usar o parâmetro ClientConfig em cmdlets

O parâmetro `ClientConfig` pode ser usado para especificar determinadas configurações quando você se conecta a um serviço. A maioria das propriedades possíveis desse parâmetro é definida na classe `Amazon.Runtime.ClientConfig`, que é herdada nas APIs para serviços da AWS. Para obter um exemplo de herança simples, veja a classe `Amazon.Keyspaces.AmazonKeyspacesConfig`. Além disso, alguns serviços definem

propriedades adicionais que são apropriadas somente para esse serviço. Para ver um exemplo de propriedades adicionais que foram definidas, consulte a classe `Amazon.S3.AmazonS3Config`, especificamente a propriedade `ForcePathStyle`.

Usar o parâmetro **`ClientConfig`**

Para usar o parâmetro `ClientConfig`, você pode especificá-lo na linha de comando como um objeto `ClientConfig` ou usar o nivelamento do PowerShell para transmitir uma coleção de valores de parâmetros para um comando como uma unidade. Esses métodos são mostrados nos exemplos a seguir. Os exemplos pressupõem que o módulo `AWS.Tools.S3` tenha sido instalado e importado e que você tenha um perfil de credenciais `[default]` com as permissões apropriadas.

Definir um objeto **`ClientConfig`**

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

Adicionar propriedades **`ClientConfig`** usando o nivelamento do PowerShell

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

Get-S3Object @params
```

Usar uma propriedade indefinida

Ao usar o nivelamento do PowerShell, se você especificar uma propriedade `ClientConfig` que não existe, o AWS Tools for PowerShell só detectará o erro no tempo de execução, quando retornará uma exceção. Modificação do exemplo acima:

```
$params=@{
    ClientConfig=@{
        ForcePathStyle=$true
```

```
        UndefinedProperty="Value"
        Timeout=[TimeSpan]::FromMilliseconds(150000)
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

Esse exemplo gerará uma exceção semelhante à seguinte:

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

Especificar a Região da AWS

É possível usar o parâmetro `ClientConfig` a fim de definir a Região da AWS para o comando. A região é definida por meio da propriedade `RegionEndpoint`. O AWS Tools for PowerShell calcula a região a ser usada de acordo com a seguinte precedência:

1. O parâmetro `-Region`
2. A região transmitida no parâmetro `ClientConfig`
3. O estado da sessão do PowerShell
4. O arquivo `config` da AWS compartilhado
5. As variáveis de ambiente
6. Os metadados da instância do Amazon EC2, se habilitados.

Segurança para este produto ou serviço da AWS

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS, você se contará com um datacenter e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança. A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa todos os serviços oferecidos na Nuvem AWS e por fornecer serviços que você pode usar com segurança. Nossa responsabilidade de segurança é a maior prioridade na AWS, e a eficácia da nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos Programas de Compatibilidade da AWS.

Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você estiver usando e por outros fatores, incluindo a confidencialidade dos dados, os requisitos da organização e as leis e regulamentos aplicáveis.

Esse produto ou serviço da AWS segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a página de documentação de segurança do serviço da AWS e Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade.

Tópicos

- Proteção de dados neste produto ou serviço da AWS
- Identity and Access Management
- Validação de conformidade para este produto ou serviço da AWS
- Aplicar uma versão mínima do TLS nas ferramentas para PowerShell

Proteção de dados neste produto ou serviço da AWS

O Modelo de Responsabilidade Compartilhada da AWS se aplica à proteção de dados nesse produto ou serviço da AWS. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas

de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para mais informações sobre a proteção de dados na Europa, consulte o artigo AWS Shared Responsibility Model and GDPR no Blog de segurança da AWS.

Para fins de proteção de dados, recomendamos que você proteja asConta da AWS credenciais da e configure as contas de usuário individuais com o AWS IAM Identity Center ou o AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA [multi-factor authentication]) com cada conta.
- Use SSL/TLS para se comunicar com os atributos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure o registro em log das atividades da API e do usuário com o .AWS CloudTrail
- Use AWS as soluções de criptografia da , juntamente com todos os controles de segurança padrão dos Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comandos ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com esse produto ou serviço da AWS ou outros Serviços da AWS usando o console, a API, a AWS CLI ou os AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia de dados

Um atributo fundamental de qualquer serviço seguro é que as informações sejam criptografadas quando não estão sendo usadas ativamente.

Criptografia em repouso

A própria AWS Tools for PowerShell não armazena nenhum dado do cliente além das credenciais de que precisa para interagir com os serviços da AWS em nome do usuário.

Se você usar a AWS Tools for PowerShell para invocar um serviço da AWS que transmita dados do cliente ao computador local para armazenamento, consulte o capítulo Segurança e conformidade no Guia do usuário desse serviço para obter informações sobre como esses dados são armazenados, protegidos e criptografados.

Criptografia em trânsito

Por padrão, todos os dados transmitidos do computador cliente que executa o AWS Tools for PowerShell e dos endpoints de serviço da AWS são criptografados enviando tudo por meio de uma conexão HTTPS/TLS.

Você não precisa fazer nada para ativar o uso do HTTPS/TLS. Ele está sempre ativado.

Identity and Access Management

O AWS Identity and Access Management (IAM) é um serviço da AWS service (Serviço da AWS) que ajuda a controlar o acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) a usar os recursos do AWS. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

Tópicos

Público

O uso do AWS Identity and Access Management (IAM) varia dependendo do trabalho que for realizado no AWS.

Usuário do serviço: se você usar Serviços da AWS para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões necessárias. À medida que usar mais recursos do AWS para fazer seu trabalho, você poderá precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um atributo na AWS, consulte Solução de problemas de identidade e acesso do AWS ou o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador do serviço – Se você for o responsável pelos recursos do AWS na empresa, provavelmente terá acesso total ao AWS. Cabe a você determinar quais funcionalidades e recursos do AWS os usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com a AWS, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador do IAM – Se você for um administrador do IAM, talvez queira saber detalhes sobre como pode gravar políticas para gerenciar o acesso ao AWS. Para visualizar exemplos de políticas baseadas em identidade da AWS que podem ser usadas no IAM, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Autenticando com identidades

A autenticação é a forma como você faz login na AWS usando suas credenciais de identidade. É necessário ser autenticado (fazer login na AWS) como o usuário raiz da Usuário raiz da conta da AWS, como usuário do IAM ou assumindo um perfil do IAM.

Você pode fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. Os usuários do AWS IAM Identity Center (IAM Identity Center), a autenticação única da empresa e as suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como uma identidade federada, o administrador já configurou anteriormente a federação de identidades utilizando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

É possível fazer login no ou no portal de acesso da AWS Management Console dependendo do tipo de usuário que você é.AWS Para obter mais informações sobre como fazer login na AWS, consulte How to sign in to your Conta da AWS (Como fazer login na conta da) no Início de Sessão da AWS User Guide (Guia do usuário do).

Se você acessar a AWS programaticamente, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface da linha de comando (CLI) para você assinar criptograficamente as solicitações usando as suas credenciais. Se você não utilizar as ferramentas da AWS, deverá assinar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulteAssinar AWSsolicitações de API da noGuia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça mais informações de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do AWS IAM Identity Center e Usar a autenticação multifator (MFA) naAWS no Guia do usuário do IAM.

Usuário raiz da Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos os atributos e Serviços da AWS na conta. Essa identidade, denominada usuário raiz da Conta da AWS, e é acessada por login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não utilizar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários, inclusive os que precisam de acesso de administrador, usem a federação com um provedor de identidades para acessar Serviços da AWS usando credenciais temporárias.

Identidade federada é um usuário de seu diretório de usuários corporativos, um provedor de identidades da web AWS Directory Service, o , o diretório do Centro de Identidade ou qualquer usuário que acesse os Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, elas assumem perfis que fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o .AWS IAM Identity Center Você pode criar usuários e grupos no Centro de Identidade do IAM ou se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas

Contas da AWS e aplicações. Para obter mais informações sobre o Centro de Identidade do IAM, consulte O que é o Centro de Identidade do IAM? no Guia do usuário do AWS IAM Identity Center.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas para uma única pessoa ou aplicação. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de utilização específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível utilizar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e atribuir a esse grupo permissões para administrar atributos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de uma função) no Guia do usuário do IAM.

Perfis do IAM

Um perfil do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. É possível assumir temporariamente um perfil do IAM no AWS Management Console alternando perfis. É possível assumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Uso de funções do IAM no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter

mais informações sobre perfis para federação, consulte Criar uma função para um provedor de identidade de terceiros no Guia do usuário do IAM. Se você usar o IAM Identity Center, deverá configurar um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário do AWS IAM Identity Center.

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um atributo (em vez de usar um perfil como proxy). Para saber a diferença entre perfis e políticas baseadas em atributo para acesso entre contas, consulte Como os perfis do IAM diferem das políticas baseadas em atributo no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam atributos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Encaminhamento de sessões de acesso (FAS): qualquer pessoa que utilizar uma função ou usuário do IAM para realizar ações na AWS é considerada uma entidade principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O recurso FAS utiliza as permissões da entidade principal que chama um AWS service (Serviço da AWS), combinadas às permissões do AWS service (Serviço da AWS) solicitante, para realizar solicitações para serviços downstream. As solicitações de FAS só são feitas quando um serviço recebe uma solicitação que exige interações com outros Serviços da AWS ou com recursos para serem concluídas. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.
  - Função vinculada ao serviço: uma função vinculada a serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir o perfil para executar

uma ação em seu nome. Os perfis vinculados ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode exibir, mas não pode editar as permissões para perfis vinculados ao serviço.

- Aplicações em execução no Amazon EC2: é possível usar um perfil do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir um perfil da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém a perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar os perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Como gerenciar acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou atributos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou atributo, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário raiz ou sessão de perfil) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas são armazenadas na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar AWS as políticas JSON da para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM a perfis, e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de perfil do AWS Management Console, da AWS CLI ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas embutidas ou políticas gerenciadas. As políticas embutidas são anexadas diretamente a um único usuário, grupo ou função. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e perfis na Conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recurso

Políticas baseadas em atributos são documentos de políticas JSON que você anexa a um atributo. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recurso, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse recurso e em que condições. Você precisa especificar uma entidade principal em uma política baseada em recurso. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Políticas baseadas em atributos são políticas em linha que estão localizadas nesse serviço. Não é possível usar as políticas gerenciadas da AWS do IAM em uma política baseada em atributos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou funções da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recurso, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços compatíveis com ACLs. Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS aceita tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs): SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS pertencentes à sua empresa. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas-membro, incluindo cada .Usuário raiz da conta da AWS Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) noAWS Organizations Guia do usuário do .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliaçãode políticas no Guia do usuário do IAM.

Como Serviços da AWS funcionam com o IAM

Para obter uma visão geral de como Serviços da AWS funcionam com a maioria dos atributos do IAM, consulte Serviços da AWS compatíveis com o IAM no Guia do usuário do IAM.

Para saber como usar um AWS service (Serviço da AWS) específico com o IAM, consulte a seção de segurança do Guia do usuário do serviço relevante.

Solução de problemas de identidade e acesso do AWS

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o e o IAM.AWS

Tópicos

- Não tenho autorização para executar uma ação no AWS
- Não estou autorizado a executar iam:PassRole
- Quero permitir que pessoas fora da minha Conta da AWS acessem meus recursos AWS

Não tenho autorização para executar uma ação no AWS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `awes:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao atributo *`my-example-widget`* usando a ação `awes:`*`GetWidget`*.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a executar iam:PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS.

Alguns Serviços da AWS permitem que você passe uma função existente para o serviço, em vez de criar uma nova função de serviço ou função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no AWS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus recursos AWS

Você pode criar uma função que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recurso ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Saiba mais consultando o seguinte:

- Para saber se o AWS suporta esses recursos, consulte Como Serviços da AWS funcionam com o IAM.
- Saiba como conceder acesso a seus recursos em todos os Contas da AWS pertencentes a você, consulte Fornecendo Acesso a um Usuário do IAM em Outro Conta da AWS Pertencente a Você no Guia de Usuário do IAM.

- Para saber como conceder acesso a seus recursos para Contas da AWS terceirizadas, consulte Fornecimento de acesso a Contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em atributos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em atributos no Guia do usuário do IAM.

Validação de conformidade para este produto ou serviço da AWS

Para saber se um AWS service (Serviço da AWS) está no escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo por programa de conformidade em que você está interessado. Para obter informações gerais, consulteAWS Programas de conformidade.

É possível fazer download de relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Downloading Reports inAWS Artifact.

Sua responsabilidade de conformidade ao usar o Serviços da AWS é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. A AWS fornece os seguintes atributos para ajudar com a conformidade:

- Guias de referência rápida de conformidade e segurança - estes guias de implantação discutem considerações sobre arquitetura e fornecem as etapas para a implantação de ambientes de linha de base focados em segurança e conformidade na AWS.
- Architecting for HIPAA Security and Compliance on Amazon Web Services (Arquitetura para segurança e conformidade com HIPAA no Amazon Web Services): esse estudo técnico descreve como as empresas podem usar a AWS para criar aplicações adequadas aos padrões HIPAA.

> **Note**
>
> Nem todos os Serviços da AWS estão qualificados pela HIPAA. Para obter mais informações, consulte a Referência dos serviços qualificados pela HIPAA.

- atributos de conformidade da AWS: essa coleção de manuais e guias pode ser aplicada a seu setor e local.

- Guias de conformidade do cliente da AWS: entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as práticas recomendadas para proteção de Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- Avaliar atributos com regras no AWS Config Guia do desenvolvedor: o serviço AWS Config avalia como as configurações de atributos estão em conformidade com práticas internas, diretrizes do setor e regulamentos.
- AWS Security Hub: este AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança na AWS. O Security Hub usa controles de segurança para avaliar os atributos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- AWS Audit Manager: esse AWS service (Serviço da AWS) ajuda a auditar continuamente seu uso da para simplificar a forma como você gerencia os riscos e a conformidade com regulamentos e padrões do setor.

Esse produto ou serviço da AWS segue o modelo de responsabilidade compartilhada por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a página de documentação de segurança do serviço da AWS e Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade.

Aplicar uma versão mínima do TLS nas ferramentas para PowerShell

Para aumentar a segurança ao se comunicar com serviços da AWS, você deve configurar as ferramentas para PowerShell para usar a versão apropriada do TLS. Para obter informações sobre como fazer isso, consulte Aplicar uma versão mínima do TLS no Guia do desenvolvedor do AWS SDK for .NET.

Referência do cmdlet da Ferramentas para PowerShell

As Ferramentas para PowerShell fornece cmdlets que você pode usar para acessar serviços da AWS. Para ver quais cmdlets estão disponíveis, consulte a Referência do cmdlet do AWS Tools for PowerShell.

Histórico do documento

Este tópico descreve alterações significativas na documentação do AWS Tools for PowerShell.

Também atualizamos a documentação periodicamente em resposta a comentários dos clientes. Para enviar comentários sobre um tópico, use os botões de feedback ao lado de "Esta página foi útil?" localizado na parte inferior de cada página.

Para obter informações adicionais sobre alterações e atualizações no AWS Tools for PowerShell, consulte as notas de lançamento.

Alteração	Descrição	Data
Configure a autenticação da ferramenta com AWS	Foram adicionadas informaçõ es sobre o suporte para SSO no AWS Tools for PowerShell.	15 de março de 2024
Referência de cmdlet para as Ferramentas para PowerShell	Seção adicionada com um link para a referência do PowerShell cmdlet Tools for.	17 de novembro de 2023
Inclusão de mais atualizações de práticas recomendadas do IAM	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte Práticas recomenda das de segurança no IAM.	12 de outubro de 2023
Instalar no Windows	Foram removidas as informações sobre a instalaçã o do Tools for Windows PowerShell usando o MSI, que se tornou obsoleto.	25 de setembro de 2023
Atualizações de práticas recomendadas do IAM	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações,	8 de setembro de 2023

	consulte Práticas recomendadas de segurança no IAM.	
Pipelining e $ AWSHistory	O parâmetro `IncludeSensitiveData` foi adicionado ao cmdlet `Set-AWSHistoryConfiguration` .	9 de março de 2023
Usando o ClientConfig parâmetro em cmdlets	Foram adicionadas informações sobre o suporte para o ClientConfig parâmetro.	28 de outubro de 2022
Execute uma instância do Amazon EC2 usando o Windows PowerShell	Adicionadas notas sobre a retirada do EC2-Classic.	26 de julho de 2022
AWS Tools for PowerShell Versão 4	Adicionadas informações sobre a versão 4, incluindo instruções de instalação para Windows e Linux/macOS e um tópico sobre migração que descreve as diferenças da versão 3 e apresenta novos recursos.	21 de novembro de 2019
AWS Tools for PowerShell 3.3.563	Adicionadas informações sobre como instalar e usar a versão de visualização do módulo `AWS.Tools.Common` . Esse novo módulo divide o pacote monolítico antigo em um módulo compartilhado e um módulo por AWS serviço.	18 de outubro de 2019

AWS Tools for PowerShell 3.3.343.0	Foram adicionadas informações à seção Usando a AWS Tools for PowerShell seção que apresenta as AWS Lambda ferramentas PowerShell para desenvolvedores PowerShell principais criarem AWS Lambda funções.	11 de setembro de 2018
AWS Tools for Windows PowerShell 3.1.31.0	Adição de informações à seção Conceitos básicos sobre novos cmdlets que usam Security Assertion Markup Language (SAML) para oferecer suporte à configuração de identidade federada para os usuários.	1 de dezembro de 2015
AWS Tools for Windows PowerShell 2.3.19	Foram adicionadas informações à seção Cmdlets Discovery and Aliases sobre o novo `Get-AWSCmdletName` cmdlet que podem ajudar os usuários a encontrar mais facilmente os cmdlets desejados. AWS	5 de fevereiro de 2015

AWS Tools for Windows PowerShell 1.1.1.0	A saída da coleção dos cmdlets é sempre enumerada no pipeline. PowerShell Suporte automático para chamadas de serviço paginável. A nova variável $ AWSHistory shell coleta respostas de serviço e, opcionalmente, solicitaç ões de serviço. AWSRegion as instâncias usam o campo Região em vez de SystemName para ajudar no pipeline. O Remove-S3Bucket suporta a opção - switch. DeleteObjects Problema de usabilidade corrigido com Set-. AWSCredentials Inicializar - AWSDefaults relata de onde obteve credenciais e dados da região. Stop-EC2Instance aceita instâncias Amazon.EC 2.Model.Reservation como entrada. Tipos de parâmetro Generic List<T> substituí dos por tipos de matriz (T[]). Cmdlets que excluem ou encerram o prompt de recursos para confirmaç ão antes da exclusão. O cmdlet Write-S3Object oferece suporte ao conteúdo de texto em linha para upload para o Amazon S3.	15 de maio de 2013

AWS Tools for Windows PowerShell 1.0.1.0	O local de instalação do PowerShell módulo Tools for Windows foi alterado para que os ambientes que usam o Windows PowerShell versão 3 possam aproveitar o carregamento automático. O módulo e os arquivos de suporte agora são instalados em uma subpasta `AWSPowerShell` abaixo de `AWS ToolsPowerShell`. Os arquivos de versões anteriores existentes na pasta `AWS ToolsPowerShell` são removidos automaticamente pelo instalador. O `PSModulePath` para Windows PowerShell (todas as versões) é atualizado nesta versão para conter a pasta principal do módulo (`AWS ToolsPowerShell`). Para sistemas com Windows PowerShell versão 2, o atalho do menu Iniciar é atualizado para importar o módulo do novo local e, em seguida, executá-lo `Initialize-AWSDefaults`. Para sistemas com Windows PowerShell versão 3, o atalho do menu Iniciar é atualizado para remover o `Import-Module` comando, deixando apenas `Initializ`	21 de dezembro de 2012

	`e-AWSDefaults` . Se você editou seu PowerShell perfil para executar um `Import-Module` dos `AWSPowerShell.psd1` arquivos, precisará atualizá-lo para apontar para o novo local do arquivo (ou, se estiver usando a PowerShell versão 3, remover a `Import-Module` instrução, pois ela não é mais necessária). Como resultado dessas alterações, o PowerShell módulo Tools for Windows agora está listado como um módulo disponível durante a execução. `Get-Module -ListAvailable` Além disso, para usuários do Windows PowerShell versão 3, a execução de qualquer cmdlet exportado pelo módulo carregará automaticamente o módulo no PowerShell shell atual sem precisar usá-lo primeiro. `Import-Module` Isso permite o uso interativo de cmdlets em um sistema com uma política de execução que não permita a execução do script.	
AWS Tools for Windows PowerShell 1.0.0.0	Lançamento inicial	6 de dezembro de 2012

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.