

Guia do usuário

# AWS Tools for PowerShell



# AWS Tools for PowerShell: Guia do usuário

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

---

# Table of Contents

O que é o AWS Tools for PowerShell? .....	1
Manutenção e suporte para as versões principais do SDK .....	2
AWS.Tools .....	2
AWSPowerShell.NetCore .....	3
AWSPowerShell .....	3
Como usar este guia .....	4
Instalação .....	5
Instalar no Windows .....	5
Pré-requisitos .....	6
Instalar o AWS.Tools .....	6
Instalar AWSPowerShell. NetCore .....	9
Instalar AWSPowerShell .....	10
Ativar a execução do script .....	11
Versionamento .....	12
Atualizando AWS Tools for PowerShell .....	14
Instalar no Linux ou no macOS .....	16
Visão geral da configuração .....	16
Pré-requisitos .....	6
Instalar o AWS.Tools .....	17
Instalar AWSPowerShell. NetCore .....	20
Execução do script .....	11
Configurando o console PowerShell .....	22
Inicialize sua sessão PowerShell .....	22
Versionamento .....	12
Atualizando o AWS Tools for PowerShell no Linux ou no macOS .....	24
Informações relacionadas .....	25
Migrar do AWS Tools for PowerShell versão 3.3 para a versão 4 .....	25
Nova versão totalmente modularizada do AWS.Tools .....	25
Novo cmdlet Get-AWSService .....	26
Novo parâmetro -Select para controlar o objeto retornado por um cmdlet .....	26
Limitação mais consistente do número de itens na saída .....	28
Parâmetros de fluxo mais fáceis de usar .....	29
Estendendo o pipe por nome da propriedade .....	29
Parâmetros comuns estáticos .....	30

O AWS.Tools declara e aplica os parâmetros obrigatórios .....	30
Todos os parâmetros são anuláveis .....	30
Remover recursos defasados anteriormente .....	31
Conceitos básicos .....	32
Configurar autenticação nas ferramentas .....	32
Habilitar e configurar o Centro de Identidade do IAM .....	33
Configure as ferramentas PowerShell para usar o IAM Identity Center. ....	33
Iniciar uma sessão do portal de AWS acesso .....	35
Exemplo .....	36
Mais informações .....	36
Use o AWS CLI .....	37
Especificar AWS regiões .....	41
Especificar um endpoint não padrão ou personalizado .....	43
Mais informações .....	43
Configurar a identidade federada .....	43
Pré-requisitos .....	44
Como um usuário com federação de identidade obtém acesso federado a APIs de serviço da AWS .....	44
Como funciona o suporte ao SAML no AWS Tools for PowerShell .....	46
Como usar cmdlets de configuração de SAML do PowerShell .....	47
Leitura adicional .....	52
Aliases e descoberta de cmdlet .....	52
Descoberta de cmdlets .....	52
Nomenclatura de cmdlets e aliases .....	59
Pipeline \$AWSHistory .....	63
\$AWSHistory .....	64
Resolução de perfil e credenciais .....	68
Ordem de pesquisa de credenciais .....	68
Usuários e perfis .....	69
Usuários e conjuntos de permissões .....	69
Perfis de serviço .....	69
Uso de credenciais herdadas .....	70
Avisos e diretrizes importantes .....	71
Credenciais da AWS .....	72
Credenciais compartilhadas .....	81
Trabalhar com serviços da AWS .....	87

Codificação de concatenação de arquivo do PowerShell .....	87
Objetos retornados para as ferramentas do Powershell .....	88
Amazon EC2 .....	88
Amazon S3 .....	88
AWS Lambda e AWS Tools for PowerShell .....	89
Amazon SNS e Amazon SQS .....	89
CloudWatch .....	89
Consulte também .....	89
Tópicos .....	89
Amazon S3 e Tools for Windows PowerShell .....	90
Criar um bucket do Amazon S3, verificar sua região e, opcionalmente, removê-lo .....	91
Configurar um bucket do Amazon S3 como um site e ativar o registro em log .....	92
Fazer upload de objetos para um bucket do Amazon S3 .....	92
Excluir objetos e buckets do Amazon S3 .....	95
Upload de conteúdo de texto em linha para o Amazon S3 .....	96
Amazon EC2 e Tools for Windows PowerShell .....	97
Criar um par de chaves .....	97
Criar um grupo de segurança .....	100
Encontrar uma AMI .....	104
Iniciar uma instância do .....	108
AWS Lambda e AWS Tools for PowerShell .....	112
Pré-requisitos .....	6
Instale o módulo do AWSLambdaPSCore .....	113
Consulte também .....	89
Amazon SQS, Amazon SNS e Tools for Windows PowerShell .....	114
Crie uma fila do Amazon SQS e obtenha o nome do recurso da Amazon (ARN) .....	114
Criar um tópico do Amazon SNS .....	114
Fornecer permissões ao tópico do SNS .....	115
Assinar a fila para o tópico do SNS .....	115
Fornecer permissões .....	116
Verificar os resultados .....	116
CloudWatch do AWS Tools for Windows PowerShell .....	117
Publicar uma métrica personalizada no seu painel do CloudWatch .....	117
Consulte também .....	89
Usar ClientConfig .....	118
Usar o parâmetro ClientConfig .....	119

---

Usar uma propriedade indefinida .....	119
Especificar a Região da AWS .....	120
Exemplos de código .....	121
Ações e cenários .....	121
ACM .....	123
AppStream 2.0 .....	128
Aurora .....	155
Auto Scaling .....	156
AWS Budgets .....	193
AWS Cloud9 .....	195
AWS CloudFormation .....	202
CloudFront .....	215
CloudTrail .....	223
CloudWatch .....	228
CodeCommit .....	233
CodeDeploy .....	238
CodePipeline .....	257
Identidade do Amazon Cognito .....	276
AWS Config .....	280
Device Farm .....	300
AWS Directory Service .....	301
AWS DMS .....	327
DynamoDB .....	328
Amazon EC2 .....	343
Amazon ECR .....	475
Amazon ECS .....	477
Amazon EFS .....	483
Amazon EKS .....	490
Elastic Load Balancing - Versão 1 .....	503
Elastic Load Balancing - Versão 2 .....	523
Amazon FSx .....	548
AWS Glue .....	556
AWS Health .....	558
IAM .....	559
Kinesis .....	634
Lambda .....	638

Amazon ML .....	651
Macie .....	657
AWS OpsWorks .....	658
AWS Price List .....	659
Resource Groups (Grupos de recursos) .....	662
Resource Groups Tagging API .....	670
route 53 .....	675
Amazon S3 .....	691
S3 Glacier .....	727
Amazon SES .....	731
Amazon SNS .....	732
Amazon SQS .....	734
AWS STS .....	746
AWS Support .....	751
Systems Manager .....	757
Amazon Translate .....	830
AWS WAFV2 .....	831
WorkSpaces .....	832
Segurança .....	849
Proteção de dados .....	849
Criptografia de dados .....	851
Identity and Access Management .....	851
Público .....	852
Autenticando com identidades .....	852
Como gerenciar acesso usando políticas .....	856
Como Serviços da AWS funcionam com o IAM .....	859
Solução de problemas de identidade e acesso do AWS .....	859
Compliance Validation .....	861
Aplicar uma versão mínima do TLS .....	862
Considerações adicionais sobre segurança .....	863
Registro de informações confidenciais .....	863
Referência do cmdlet .....	864
Histórico do documento .....	865
.....	dcclxxi

# O que é o AWS Tools for PowerShell?

AWS Tools for PowerShell São um conjunto de PowerShell módulos baseados na funcionalidade exposta pelo AWS SDK for .NET. Eles AWS Tools for PowerShell permitem que você crie scripts de operações em seus AWS recursos a partir da linha de PowerShell comando.

Os cmdlets fornecem uma PowerShell experiência idiomática para especificar parâmetros e lidar com resultados, mesmo que sejam implementados usando as várias AWS APIs de consulta HTTP de serviço. Por exemplo, os cmdlets do PowerShell pipeline de AWS Tools for PowerShell suporte, ou seja, você pode canalizar PowerShell objetos para dentro e para fora dos cmdlets.

Eles AWS Tools for PowerShell são flexíveis na forma como permitem que você gerencie as credenciais, incluindo suporte para a infraestrutura 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.

Eles AWS Tools for PowerShell oferecem suporte ao mesmo conjunto de serviços e AWS regiões compatíveis com o SDK. Você pode instalar o AWS Tools for PowerShell em computadores que executam sistemas operacionais Windows, Linux ou macOS.

## Note

AWS Tools for PowerShell a versão 4 é a versão principal mais recente e é uma atualização compatível com versões anteriores da versão 3.3. AWS Tools for PowerShell 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](#).

Eles AWS Tools for PowerShell estão disponíveis como os três pacotes distintos a seguir:

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



# Manutenção e suporte para as versões principais do 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](#):

- [AWS Política de manutenção de SDKs e ferramentas](#)
- [AWS Matriz de suporte de versões de SDKs e ferramentas](#)

## AWS.Tools- Uma versão modularizada do AWS Tools for PowerShell

PowerShell Gallery **AWS.Tools**

ZIP Archive **AWS.Tools**

Essa versão do AWS Tools for PowerShell é a versão recomendada para qualquer computador executado 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.

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

O `AWS.Tools.Installer` módulo fornece cmdlets que permitem instalar, atualizar e remover os módulos de cada um dos AWS serviços. 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 AWS serviço, basta executar o comando. PowerShell importa automaticamente o `AWS.Tools.Common` módulo e o módulo do AWS serviço cujo cmdlet você deseja executar. Esse módulo é instalado automaticamente se você usar o módulo `AWS.Tools.Installer` para instalar os módulos do serviço.

Você pode instalar essa versão do AWS Tools for PowerShell em computadores que estão 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

PowerShell Gallery **AWSPowerShell.NetCore**

ZIP Archive **AWSPowerShell.NetCore**

Essa versão consiste em um único módulo grande que contém suporte para todos os AWS serviços. Antes de usar este módulo, você deverá importá-lo manualmente.

Você pode instalar essa versão do AWS Tools for PowerShell em computadores que estão 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, quando precisamos especificar apenas esta versão, nos referimos a ela pelo nome do módulo: *AWSPowerShell. NetCore*.

## AWSPowerShell - Uma versão de módulo único para Windows PowerShell

PowerShell Gallery **AWSPowerShell**

ZIP Archive **AWSPowerShell**

Essa versão do AWS Tools for PowerShell é compatível e pode ser instalada somente em computadores Windows que estejam executando as PowerShell versões 2.0 a 5.1 do Windows.

Ele não é compatível com o PowerShell Core 6.0 ou posterior, nem com qualquer outro sistema operacional (Linux ou macOS). Essa versão consiste em um único módulo grande que contém suporte para todos os AWS serviços.

Ao longo deste guia, quando precisamos especificar apenas esta versão, nos referimos 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 AWS Tools for PowerShell o. Inclui como AWS se inscrever, caso ainda não tenha uma conta, e como criar um usuário do IAM que você possa usar para executar os cmdlets.

### [Conceitos básicos da AWS Tools for Windows PowerShell](#)

Esta seção descreve os fundamentos do uso do AWS Tools for PowerShell, como especificar credenciais e AWS regiões, localizar cmdlets para um serviço específico e usar 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 realizar algumas das AWS tarefas mais comuns.

# 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](#)
- [Instalando AWS Tools for PowerShell no Linux ou macOS](#)
- [Migrar do AWS Tools for PowerShell versão 3.3 para a versão 4](#)

## 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 extraíndo-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 extraíndo-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 [PowerShellGet](#) instalado 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 compartilhados `AWS.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 `powershell` 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 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 em [Atualizando 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 começouudo `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 [ListObjectsResponse](#) objeto. 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 específica. 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](#)
- [Especificar AWS regiões](#)
- [Configurar a identidade federada com o AWS Tools for PowerShell](#)
- [Aliases e descoberta de cmdlet](#)
- [Pipeline \\$AWSHistory](#)
- [Resolução de perfil e credenciais](#)
- [Informações adicionais sobre usuários e perfis](#)
- [Uso de credenciais herdadas](#)

## 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](#), 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, consulte [Use 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 `compartilhadoconfig`.

## 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](#)

## 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](#), 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.SSOIDC`

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 .SSOIDC 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, SS00IDC

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

# 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.

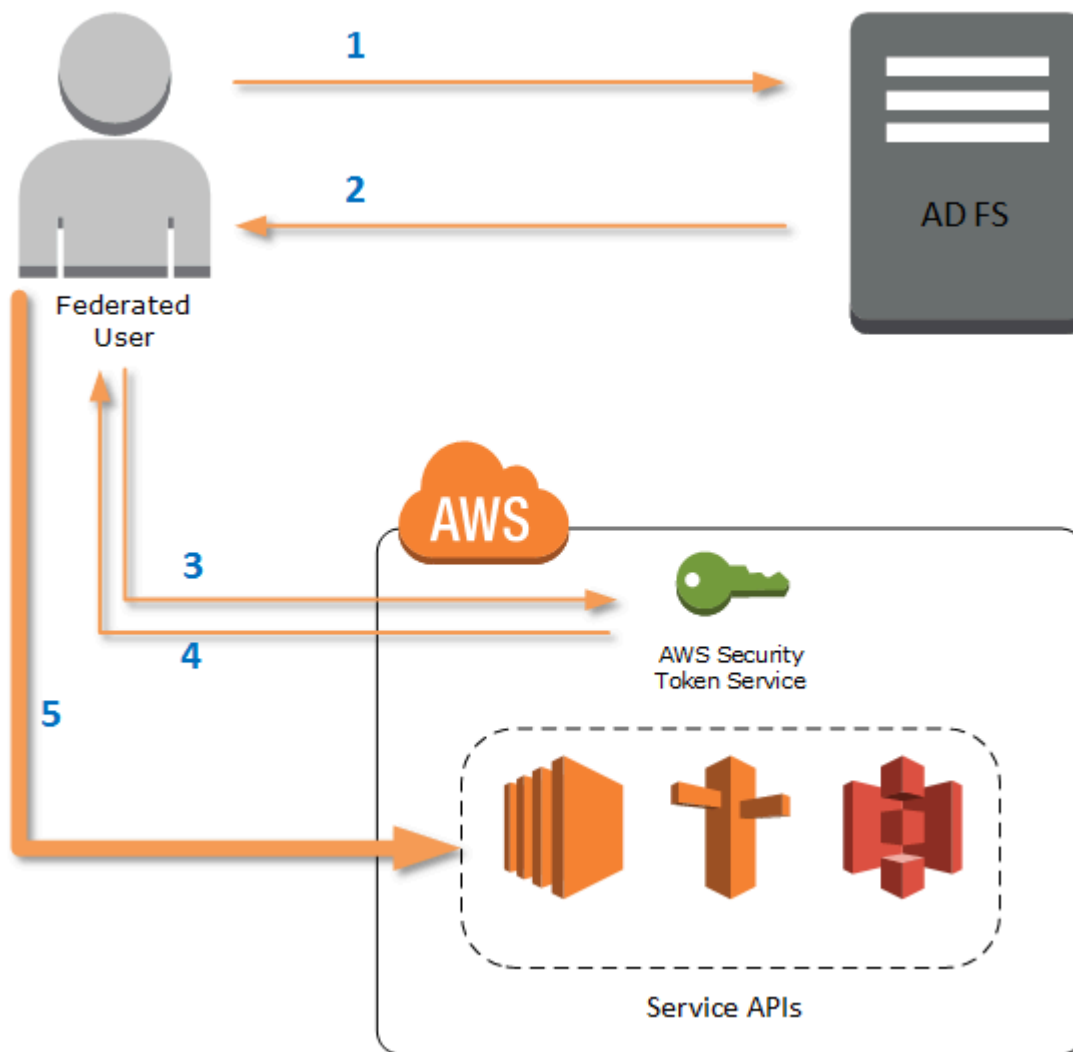
## Pré-requisitos

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

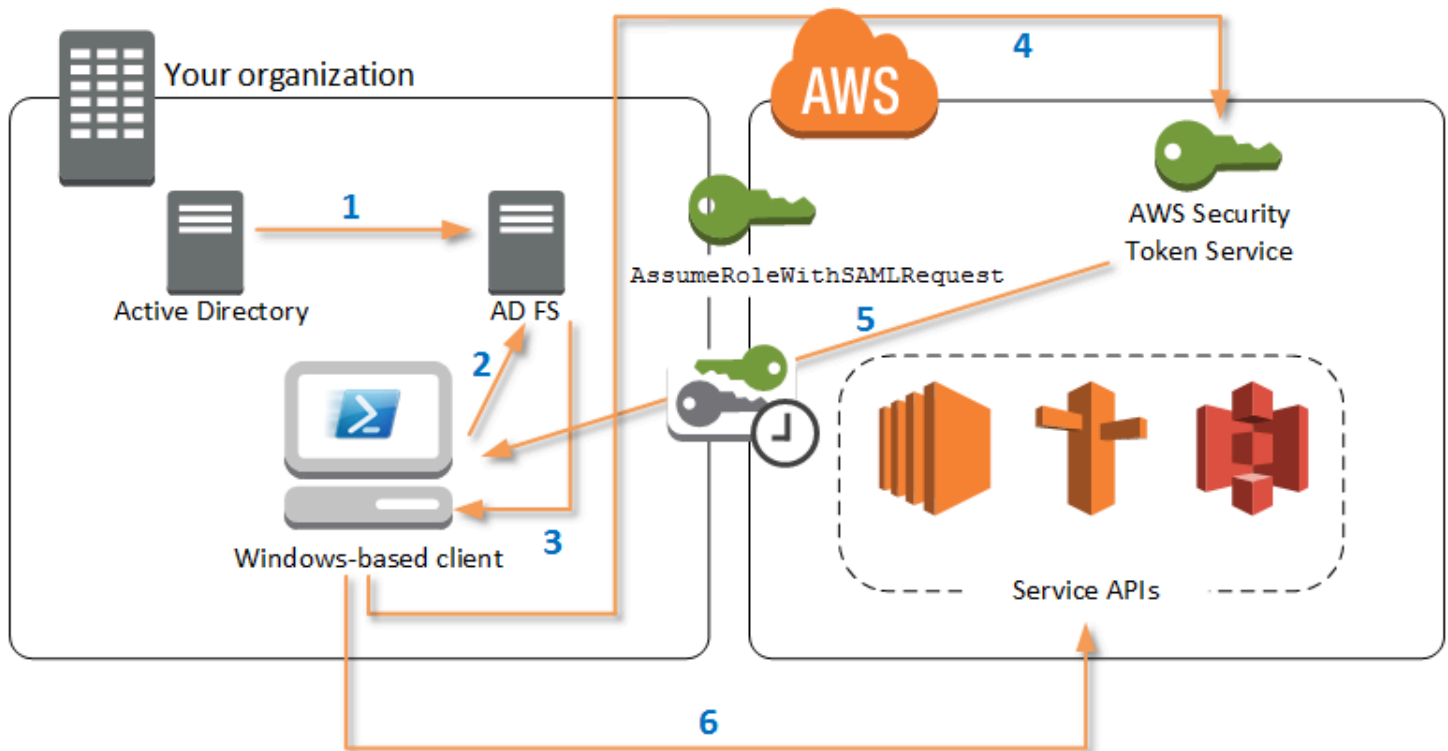
O processo a seguir descreve, em alto nível, como um usuário do Active Directory (AD) é federado pelo AD FS para obter acesso aos recursos da AWS.



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 dos 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 `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
```

- 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
```

- 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 AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWEEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Disable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWEEventSource AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Enable-CWERule AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEEventBus AWSPowerShell.NetCore	3.3.563.1	
Cmdlet	Get-CWEEventBusList AWSPowerShell.NetCore	3.3.563.1	



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.txt
```

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

**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 AWSPowerShell	Get-ASAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-ASACommunications	3.3.343.0
Alias AWSPowerShell	Get-ASAServices	3.3.343.0
Alias AWSPowerShell	Get-ASASeverityLevels	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckRefreshStatuses	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorChecks	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckSummaries	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHooks	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHookTypes	3.3.343.0
Alias AWSPowerShell	Get-AWSCredentials	3.3.343.0
Alias AWSPowerShell	Get-CDApplications	3.3.343.0
Alias AWSPowerShell	Get-CDDeployments	3.3.343.0
Alias AWSPowerShell	Get-CFCloudFrontOriginAccessIdentities	3.3.343.0
Alias AWSPowerShell	Get-CFDistributions	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigRules	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigurationRecorders	3.3.343.0
Alias AWSPowerShell	Get-CFGDeliveryChannels	3.3.343.0
Alias AWSPowerShell	Get-CFInvalidations	3.3.343.0
Alias AWSPowerShell	Get-CFNAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-CFNStackEvents	3.3.343.0
...		

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` 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](#)
- [Como gerenciar perfis](#)
- [Especificação de credenciais](#)
- [Ordem de pesquisa de credenciais](#)
- [Tratamento de credenciais no AWS Tools for PowerShell Core](#)

## 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 execute `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 substituíria 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
<code>ExternalId</code>	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 <code>ExternalID</code> como parâmetro ao assumir a função atribuída. Para obter mais informações, consulte <a href="#">Como</a>

Nome do parâmetro	Descrição
	<a href="#">usar um ID externo ao conceder acesso aos recursos da AWS para terceiros</a> 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 <a href="#">Uso da autenticação multifator (MFA) na AWS</a> no Manual do usuário 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 <a href="#">Funções do IAM</a> 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	-AccessKey
Estas são as credenciais de longo prazo para um usuário do IAM	-SecretKey
Sessão:	-AccessKey
Essas são as credenciais de curto prazo para uma função do IAM que você recupera manualmente, por exemplo, chamando diretamente o cmdlet <a href="#">Use-STRole</a> .	-SecretKey -SessionToken

Tipo de credenciais	Parâmetros que você deve usar
<p>Função:</p> <p>Estas são credenciais de curto prazo para uma função do IAM que o AWS Tools for PowerShell recupera para você.</p>	<p>-SourceProfile</p> <p>-RoleArn</p> <p>opcional: -ExternalId</p> <p>opcional: -MfaSerial</p>

## 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 <i>(user's home directory) \.aws \credentials</i> .
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\user\.aws\credentials -ProfileName
  basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\user\.aws\credentials

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

## 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\user\.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      : T7h1D0eLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xA1I
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](#)
- [Crie um grupo de segurança usando o Windows PowerShell](#)
- [Encontrar uma Imagem de máquina da Amazon usando o Windows PowerShell](#)
- [Execute 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-----
                   MIIIEogIBAAKCAQEAKK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                   Mz6bt0xPcE7EMeH1wySUP8nouAS9xb1917+Vkd74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5i1/MpkrRogHq0B0rigeTeV5Yc31v00RFFPu0Kz4kcm...
                   w3Jg8dKsWn0p10pX7V3sRC02KgJIbejQUvBFGi50QK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvgWkcFQkLmRHRoDpPb+OdFsZtjHZDpMVFmA9tT8EdbkEF...
                   3SrNeqZPsxJJIX0odb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                   GG1LfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZwa7KFNRZuCuX7jssC...
                   x098abxMr3o3TNU6p1ZYRJEQ0oJr0W+kc+/8SWb8NIwfltwmJEy...
                   1BX9X8WFX/A8VLHrT1elrKmlkNECgYEAwltkV1p0JAFhz9p7ZFEv...
                   vvVsPaF0Ev9bk9pqhx269PB50x2KokwCagDMMaYvasWobuLmNu/1...
                   lmwRx7KTeQ7W1J30LgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                   63g6N6rk2FkHZX1E62BgbewUd3eZ0S05Ip4VUdvtGcuc8/qa+e5C...
                   KXgyt9n164pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                   gYALEI7m1jJKpHWAes0hiemw5VmKyIZpzGstSJsFStER1AjiETDH...
                   YAtnI4J8dRyP9I7B0VOn3wNfIjk85gi1/00c+j8S65giLafndWGR...
                   9R9wIkm5BMUcSRRcDy0yuwKBgEbkOnGGSD0ah4HkvrUkepIbUDTD...
                   AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hk0SmWhlZGwikJ5VizBf...
                   drkBr/vTKVRMTi31VFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                   TTld5N6opg0XAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                   x302duuy7/smTwwskEWRK5IrUxoMv/VVYaqdzc0ajwieNrb1r7c...
                   -----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.

## Crie um grupo de segurança usando o Windows PowerShell

Você pode usar 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](#)
- [Criação de um grupo de segurança para o EC2-Classic](#)
- [Criação de um grupo de segurança para EC2-VPC](#)

### 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. É recomendável migrar do EC2-Classic para uma VPC. [Para obter mais informações, consulte Migrar do EC2-Classic para uma VPC no Guia do usuário do Amazon EC2 ou no Guia do usuário 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, xvdc, xvdc...}
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
```

## Execute 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 seguinte documentação do :

- [Criação de um par de chaves](#)
- [Crie um grupo de segurança usando o Windows PowerShell](#)
- [Encontre 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](#)
- [Execução de uma instância em uma VPC](#)
- [Execução de uma instância spot em uma VPC](#)

### Execução de uma instância no EC2-Classic

### Warning

Estamos aposentando o EC2-Classic em 15 de agosto de 2022. É recomendável migrar do EC2-Classic para uma VPC. [Para obter mais informações, consulte Migrar do EC2-Classic para uma VPC no Guia do usuário do Amazon EC2 ou no Guia do usuário 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" :
"11E17A2+X0uJZnw3T1gcXz4C4KPLXZxbxoEMIirelh13u/oxkWmz5+9tJKFMns1Z0qQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1b03hUENL3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
vyuGkJReV710JWPHnCN
+qTE21Id2RPkF0eGtLGawTsSPTWEvJdDbL1f7E0zZ0q1niXTUtpsZ8Swx01X3Q06u9i9qBFt0ekJFZNJp6Avu05hIk1b4yo
y0a8Y191Wp7a7EoWaBn0zhCESe7o
kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyv3WbaSvg==",
        "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 :
AQEB2vvk1e5c0KFjeIWJticabkc664yuDEjhucnI0qdVUmie7bX7GiJb17F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHn18S1bQa0R/kD+Saaq40Ivfj8x3M40h1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
+HmXdkax2Wd+9AxrH1QZV5ur1MoByKWWbDbsqoYJTJquCc10gWIak/sBx/
daBRMTiVQ4GHsrQWVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
sHN12776axknhg3j9K/Xwj54DixdsegrnKolx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2Msey0eM/Jjucg6Sr9VUnTWVhV8ErXmtoiEg==

```

## 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.html> `AmazonCloudWatchClient.PutMetricData` (Referência de SDK do .NET)
- [https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API\\_MetricDatum.html](https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_MetricDatum.html) `MetricDatum` (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.

# Ferramentas para exemplos PowerShell de código

Os exemplos de código neste tópico mostram como usar o AWS Tools for PowerShell with AWS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Exemplos entre serviços são amostras de aplicações que funcionam em vários Serviços da AWS.

## Exemplos

- [Ações e cenários usando Ferramentas para PowerShell](#)

# Ações e cenários usando Ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Serviços da AWS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

## Serviços

- [Exemplos de ACM usando ferramentas para PowerShell](#)
- [AppStream Exemplos 2.0 usando ferramentas para PowerShell](#)
- [Exemplos do Aurora usando ferramentas para PowerShell](#)
- [Exemplos de Auto Scaling usando ferramentas para PowerShell](#)
- [AWS Budgets exemplos usando ferramentas para PowerShell](#)
- [AWS Cloud9 exemplos usando ferramentas para PowerShell](#)



- [AWS CloudFormation exemplos usando ferramentas para PowerShell](#)
- [CloudFront exemplos usando ferramentas para PowerShell](#)
- [CloudTrail exemplos usando ferramentas para PowerShell](#)
- [CloudWatch exemplos usando ferramentas para PowerShell](#)
- [CodeCommit exemplos usando ferramentas para PowerShell](#)
- [CodeDeploy exemplos usando ferramentas para PowerShell](#)
- [CodePipeline exemplos usando ferramentas para PowerShell](#)
- [Exemplos de identidade do Amazon Cognito usando ferramentas para PowerShell](#)
- [AWS Config exemplos usando ferramentas para PowerShell](#)
- [Exemplos de Device Farm usando o Tools for PowerShell](#)
- [AWS Directory Service exemplos usando ferramentas para PowerShell](#)
- [AWS DMS exemplos usando ferramentas para PowerShell](#)
- [Exemplos do DynamoDB usando ferramentas para PowerShell](#)
- [Exemplos do Amazon EC2 usando ferramentas para PowerShell](#)
- [Exemplos do Amazon ECR usando ferramentas para PowerShell](#)
- [Exemplos do Amazon ECS usando ferramentas para PowerShell](#)
- [Exemplos do Amazon EFS usando ferramentas para PowerShell](#)
- [Exemplos do Amazon EKS usando ferramentas para PowerShell](#)
- [Elastic Load Balancing - Exemplos da versão 1 usando ferramentas para PowerShell](#)
- [Elastic Load Balancing - Exemplos da versão 2 usando ferramentas para PowerShell](#)
- [Exemplos do Amazon FSx usando ferramentas para PowerShell](#)
- [AWS Glue exemplos usando ferramentas para PowerShell](#)
- [AWS Health exemplos usando ferramentas para PowerShell](#)
- [Exemplos de IAM usando ferramentas para PowerShell](#)
- [Exemplos do Kinesis usando o Tools for PowerShell](#)
- [Exemplos de Lambda usando ferramentas para PowerShell](#)
- [Exemplos de Amazon ML usando ferramentas para PowerShell](#)
- [Exemplos de Macie usando ferramentas para PowerShell](#)
- [AWS OpsWorks exemplos usando ferramentas para PowerShell](#)

- [AWS Price List exemplos usando ferramentas para PowerShell](#)
- [Exemplos de Resource Groups usando Tools for PowerShell](#)
- [Exemplos da API de marcação de Resource Groups usando o Tools for PowerShell](#)
- [Exemplos do Route 53 usando Ferramentas para PowerShell](#)
- [Exemplos do Amazon S3 usando ferramentas para PowerShell](#)
- [Exemplos do S3 Glacier usando ferramentas para PowerShell](#)
- [Exemplos do Amazon SES usando ferramentas para PowerShell](#)
- [Exemplos do Amazon SNS usando ferramentas para PowerShell](#)
- [Exemplos do Amazon SQS usando ferramentas para PowerShell](#)
- [AWS STS exemplos usando ferramentas para PowerShell](#)
- [AWS Support exemplos usando ferramentas para PowerShell](#)
- [Exemplos do Systems Manager usando o Tools for PowerShell](#)
- [Exemplos do Amazon Translate usando ferramentas para PowerShell](#)
- [AWS WAFV2 exemplos usando ferramentas para PowerShell](#)
- [WorkSpaces exemplos usando ferramentas para PowerShell](#)

## Exemplos de ACM usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Get-ACMCertificate

O código de exemplo a seguir mostra como usar Get-ACMCertificate.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo mostra como retornar um certificado e sua cadeia usando o ARN do certificado.

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

- Para obter detalhes da API, consulte [GetCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-ACMCertificateDetail

O código de exemplo a seguir mostra como usar Get-ACMCertificateDetail.

#### Ferramentas para PowerShell

Exemplo 1: Retorna detalhes do certificado especificado.

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

#### Saída:

```
CertificateArn      : arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt          : 1/21/2016 5:55:59 PM
DomainName        : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy           : {}
IssuedAt          : 1/1/0001 12:00:00 AM
Issuer            :
KeyAlgorithm       : RSA-2048
NotAfter          : 1/1/0001 12:00:00 AM
NotBefore         : 1/1/0001 12:00:00 AM
RevocationReason  :
```

```
RevokedAt           : 1/1/0001 12:00:00 AM
Serial              :
SignatureAlgorithm  : SHA256WITHRSA
Status              : PENDING_VALIDATION
Subject             : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- Para obter detalhes da API, consulte [DescribeCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ACMCertificateList

O código de exemplo a seguir mostra como usar `Get-ACMCertificateList`.

### Ferramentas para PowerShell

Exemplo 1: recupera uma lista de todos os ARNs do seu certificado e o nome de domínio de cada um. O cmdlet paginará automaticamente para recuperar todos os ARNs. Para controlar manualmente a paginação, use o `MaxItems` parâmetro - para controlar quantos ARNs de certificado são retornados para cada chamada de serviço e o `NextToken` parâmetro - para indicar o ponto de partida de cada chamada.

```
Get-ACMCertificateList
```

Saída:

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

Exemplo 2: Recupera uma lista de todos os ARNs do seu certificado em que o status do certificado corresponde aos estados fornecidos.

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

Exemplo 3: Este exemplo retorna uma lista de todos os certificados na região us-east-1 que têm um tipo de chave `RSA_2048` e um uso estendido da chave, ou finalidade, de `CODE_SIGNING`.

Você pode encontrar os valores desses parâmetros de filtragem no tópico de referência da API de ListCertificates filtros: [https://docs.aws.amazon.com/acm/latest/APIReference/API\\_Filters.html](https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html).

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

Saída:

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

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

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

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

arn:aws:acm:us-east-1:8xxxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz

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

- Para obter detalhes da API, consulte [ListCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ACMCertificate

O código de exemplo a seguir mostra como usar New-ACMCertificate.

### Ferramentas para PowerShell

Exemplo 1: Cria um novo certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

Exemplo 2: Cria um novo certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

- Para obter detalhes da API, consulte [RequestCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ACMCertificate

O código de exemplo a seguir mostra como usar Remove-ACMCertificate.

### Ferramentas para PowerShell

Exemplo 1: Exclui o certificado identificado pelo ARN fornecido e pela chave privada associada. O cmdlet solicitará a confirmação antes de continuar; adicione a opção -Force para suprimir a confirmação.

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

- Para obter detalhes da API, consulte [DeleteCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Send-ACMValidationEmail

O código de exemplo a seguir mostra como usar Send-ACMValidationEmail.

### Ferramentas para PowerShell

Exemplo 1: Solicita que o e-mail para validar a propriedade do domínio para 'www.example.com' seja enviado. Se o \$ do seu shell ConfirmPreference estiver definido como 'Médio' ou inferior,

o cmdlet solicitará a confirmação antes de continuar. Adicione a opção `-Force` para suprimir as solicitações de confirmação.

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

- Para obter detalhes da API, consulte [ResendValidationEmail](#) em Referência de AWS Tools for PowerShell cmdlet.

## AppStream Exemplos 2.0 usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AppStream 2.0.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-APSResourceTag**

O código de exemplo a seguir mostra como usar `Add-APSResourceTag`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona uma tag de recurso ao AppStream recurso

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

Saída:

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

- Para obter detalhes da API, consulte [TagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Copy-APSIImage

O código de exemplo a seguir mostra como usar Copy-APSIImage.

## Ferramentas para PowerShell

Exemplo 1: Esta amostra copia uma imagem para outra região

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

Saída:

```
TestImageCopy
```

- Para obter detalhes da API, consulte [CopyImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-APSUser

O código de exemplo a seguir mostra como usar Disable-APSUser.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa um usuário no USERPOOL



```
Disable-APUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Para obter detalhes da API, consulte [DisableUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-APUser

O código de exemplo a seguir mostra como usar `Enable-APUser`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita um usuário desativado no USERPOOL

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

- Para obter detalhes da API, consulte [EnableUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSAssociatedFleetList

O código de exemplo a seguir mostra como usar `Get-APSAssociatedFleetList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe a frota associada a uma pilha

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

Saída:

```
PowershellFleet
```

- Para obter detalhes da API, consulte [ListAssociatedFleets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSAssociatedStackList

O código de exemplo a seguir mostra como usar `Get-APSAssociatedStackList`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe a pilha associada a uma frota

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

Saída:

```
PowershellStack
```

- Para obter detalhes da API, consulte [ListAssociatedStacks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSDirectoryConfigList

O código de exemplo a seguir mostra como usar Get-APSDirectoryConfigList.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe as configurações de diretório criadas em AppStream

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

Saída:

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

- Para obter detalhes da API, consulte [DescribeDirectoryConfigs](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSFleetList

O código de exemplo a seguir mostra como usar Get-APSFleetList.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe detalhes de uma frota

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

### Saída:

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

- Para obter detalhes da API, consulte [DescribeFleets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSIImageBuilderList

O código de exemplo a seguir mostra como usar `Get-APSIImageBuilderList`.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra exibe detalhes de um ImageBuilder

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

### Saída:

```
AccessEndpoints : {}
AppstreamAgentVersion : 06-19-2019
```

```

Arn                : arn:aws:appstream:us-east-1:1234567890:image-builder/
TestImage
CreatedTime       : 1/14/2019 4:33:05 AM
Description       :
DisplayName       : TestImage
DomainJoinInfo    :
EnableDefaultInternetAccess : False
IamRoleArn        :
ImageArn          : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors : {}
InstanceType      : stream.standard.large
Name              : TestImage
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform         : WINDOWS
State             : STOPPED
StateChangeReason :
VpcConfig         : Amazon.AppStream.Model.VpcConfig

```

- Para obter detalhes da API, consulte [DescreverImageBuilders](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSIImageList

O código de exemplo a seguir mostra como usar Get-APSIImageList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe AppStream imagens privadas

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

Saída:

```

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

```

- Para obter detalhes da API, consulte [DescreverImagem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSIImagePermission

O código de exemplo a seguir mostra como usar Get-APSIImagePermission.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe permissões de imagem em uma AppStream imagem compartilhada

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

Saída:

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

- Para obter detalhes da API, consulte [DescreverImagePermissions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSSessionList

O código de exemplo a seguir mostra como usar Get-APSSessionList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe a lista de sessões de uma frota

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

Saída:

```
AuthenticationType      : API
ConnectionState         : CONNECTED
FleetName                : PowershellFleet
Id                       : d8987c70-4394-4324-a396-2d485c26f2a2
```

```

MaxExpirationTime      : 12/27/2019 4:54:07 AM
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
StackName              : PowershellStack
StartTime              : 12/26/2019 12:54:12 PM
State                  : ACTIVE
UserId                 : Test

```

- Para obter detalhes da API, consulte [DescribeSessions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSSStackList

O código de exemplo a seguir mostra como usar Get-APSSStackList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe uma lista de AppStream pilhas

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

Saída:

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

- Para obter detalhes da API, consulte [DescribeStacks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSTagsForResourceList

O código de exemplo a seguir mostra como usar Get-APSTagsForResourceList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe tags em um AppStream recurso

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

Saída:

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

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSUsageReportSubscription

O código de exemplo a seguir mostra como usar `Get-APSUsageReportSubscription`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe detalhes `AppStreamUsageReport` de configuração

```
Get-APSUsageReportSubscription
```

Saída:

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

- Para obter detalhes da API, consulte [DescribeUsageReportSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSUser

O código de exemplo a seguir mostra como usar `Get-APSUser`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe a lista de usuários com status ativado

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

Saída:

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

- Para obter detalhes da API, consulte [DescribeUsers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-APSUserStackAssociation

O código de exemplo a seguir mostra como usar `Get-APSUserStackAssociation`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe a lista de usuários atribuídos a uma pilha

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Saída:

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

- Para obter detalhes da API, consulte [DescribeUserStackAssociations](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSDirectoryConfig

O código de exemplo a seguir mostra como usar `New-APSDirectoryConfig`.



## Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma configuração de diretório no AppStream

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

Saída:

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

- Para obter detalhes da API, consulte [CreateDirectoryConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSFleet

O código de exemplo a seguir mostra como usar New-APSFleet.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova AppStream frota

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

Saída:

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

```

DisconnectTimeoutInSeconds    : 900
DisplayName                    : TestFleet
DomainJoinInfo                :
EnableDefaultInternetAccess   : True
FleetErrors                   : {}
FleetType                     : ON_DEMAND
IamRoleArn                    :
IdleDisconnectTimeoutInSeconds : 0
ImageArn                      : arn:aws:appstream:us-west-2:123456789012:image/
SessionScriptTest
ImageName                     : SessionScriptTest
InstanceType                  : stream.standard.medium
MaxUserDurationInSeconds      : 57600
Name                          : TestFleet
State                         : STOPPED
VpcConfig                     : Amazon.AppStream.Model.VpcConfig

```

- Para obter detalhes da API, consulte [CreateFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSIImageBuilder

O código de exemplo a seguir mostra como usar New-APSIImageBuilder.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um Image Builder em AppStream

```

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

```

Saída:

```

AccessEndpoints                : {}
AppstreamAgentVersion          : 12-16-2019
Arn                            : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                    : 12/27/2019 11:39:24 AM
Description                    :
DisplayName                    : TestIB

```

```

DomainJoinInfo           :
EnableDefaultInternetAccess : True
IamRoleArn               :
ImageArn                 : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors       : {}
InstanceType             : stream.standard.medium
Name                     : TestIB
NetworkAccessConfiguration :
Platform                 : WINDOWS
State                    : PENDING
StateChangeReason        :
VpcConfig                 : Amazon.AppStream.Model.VpcConfig

```

- Para obter detalhes da API, consulte [CreateImageBuilder](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSIImageBuilderStreamingURL

O código de exemplo a seguir mostra como usar New-APSIImageBuilderStreamingURL.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cria um URL ImageBuilder de streaming com validade de 2 horas

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

Saída:

```

Expires                StreamingURL
-----                -
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRjZCI6IjM5MzQwM

```

- Para obter detalhes da API, consulte [CreateImageBuilderStreamingURL na Referência do AWS Tools for PowerShell](#) Cmdlet.

## New-APSSStack

O código de exemplo a seguir mostra como usar New-APSSStack.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova AppStream pilha

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

Saída:

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

- Para obter detalhes da API, consulte [CreateStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSSstreamingURL

O código de exemplo a seguir mostra como usar New-APSSstreamingURL.

## Ferramentas para PowerShell

Exemplo 1: este exemplo cria um URL de streaming do Stack

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

Saída:

```
Expires                StreamingURL
-----                -
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXB1IjoiRU5EX1VTRVViLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJCi6IjM5M
```

- Para obter detalhes da API, consulte [CreateStreamingURL na Referência do AWS Tools for PowerShell](#) Cmdlet.

## New-APSUsageReportSubscription

O código de exemplo a seguir mostra como usar New-APSUsageReportSubscription.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita relatórios AppStream de uso

```
New-APSUsageReportSubscription
```

Saída:

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

- Para obter detalhes da API, consulte [CreateUsageReportSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-APSUser

O código de exemplo a seguir mostra como usar New-APSUser.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um usuário no USERPOOL

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

Saída:

```
Test@lab.com
```

- Para obter detalhes da API, consulte [CreateUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-APSFleet

O código de exemplo a seguir mostra como usar Register-APSFleet.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra registra a frota com uma pilha

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

- Para obter detalhes da API, consulte [AssociateFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-APSUserStackBatch

O código de exemplo a seguir mostra como usar Register-APSUserStackBatch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atribui pilha a um usuário em USERPOOL

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

- Para obter detalhes da API, consulte [BatchAssociateUserStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSDirectoryConfig

O código de exemplo a seguir mostra como usar Remove-APSDirectoryConfig.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a configuração AppStream do diretório

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

Saída:

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

- Para obter detalhes da API, consulte [DeleteDirectoryConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSFleet

O código de exemplo a seguir mostra como usar Remove-APSFleet.

Ferramentas para PowerShell

Exemplo 1: Este exemplo remove e exclui uma frota AppStream

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

Saída:

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

- Para obter detalhes da API, consulte [DeleteFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSIImage

O código de exemplo a seguir mostra como usar Remove-APSIImage.

## Ferramentas para PowerShell

### Exemplo 1: Este exemplo exclui uma imagem

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

#### Saída:

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

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

- Para obter detalhes da API, consulte [DeletelImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSIImageBuilder

O código de exemplo a seguir mostra como usar Remove-APSIImageBuilder.

## Ferramentas para PowerShell

### Exemplo 1: Este exemplo exclui um ImageBuilder



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

### Saída:

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

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

- Para obter detalhes da API, consulte [DeleteImageBuilder](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSIImagePermission

O código de exemplo a seguir mostra como usar `Remove-APSIImagePermission`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove as permissões de uma imagem

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

### Saída:

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

- Para obter detalhes da API, consulte [DeleteImagePermissions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSResourceTag

O código de exemplo a seguir mostra como usar Remove-APSResourceTag.

### Ferramentas para PowerShell

Exemplo 1: este exemplo remove uma tag de recurso do AppStream recurso

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

### Saída:

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

- Para obter detalhes da API, consulte [UntagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSStack

O código de exemplo a seguir mostra como usar Remove-APSStack.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui uma pilha

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

Saída:

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

- Para obter detalhes da API, consulte [DeleteStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APSUsageReportSubscription

O código de exemplo a seguir mostra como usar Remove-APSUsageReportSubscription.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo desativa a assinatura do Relatório AppStream de Uso

```
Remove-APSUsageReportSubscription
```

Saída:

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

- Para obter detalhes da API, consulte [DeleteUsageReportSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-APUser

O código de exemplo a seguir mostra como usar `Remove-APUser`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui um usuário do USERPOOL

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

Saída:

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

- Para obter detalhes da API, consulte [DeleteUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Revoke-APSSession

O código de exemplo a seguir mostra como usar `Revoke-APSSession`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo revoga uma sessão para a frota AppStream

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

- Para obter detalhes da API, consulte [ExpireSession](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-APSFleet

O código de exemplo a seguir mostra como usar `Start-APSFleet`.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra inicia uma frota

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

- Para obter detalhes da API, consulte [StartFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-APSIImageBuilder

O código de exemplo a seguir mostra como usar Start-APSIImageBuilder.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra inicia um ImageBuilder

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

Saída:

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

- Para obter detalhes da API, consulte [StartImageBuilder](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-APSFleet

O código de exemplo a seguir mostra como usar Stop-APSFleet.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra interrompe uma frota

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

- Para obter detalhes da API, consulte [StopFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-APSIImageBuilder

O código de exemplo a seguir mostra como usar Stop-APSIImageBuilder.

### Ferramentas para PowerShell

Exemplo 1: Esta amostra interrompe um ImageBuilder

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

Saída:

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

```
Platform           : WINDOWS
State              : STOPPING
StateChangeReason  :
VpcConfig          : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes da API, consulte [StopImageBuilder](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-APSFleet

O código de exemplo a seguir mostra como usar Unregister-APSFleet.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela o registro de uma frota da pilha

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

- Para obter detalhes da API, consulte [DisassociateFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-APSUserStackBatch

O código de exemplo a seguir mostra como usar Unregister-APSUserStackBatch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove um usuário de uma pilha atribuída

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

- Para obter detalhes da API, consulte [BatchDisassociateUserStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-APSDirectoryConfig

O código de exemplo a seguir mostra como usar Update-APSDirectoryConfig.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza a configuração do diretório criada no AppStream

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

Saída:

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

- Para obter detalhes da API, consulte [UpdateDirectoryConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-APSFleet

O código de exemplo a seguir mostra como usar Update-APSFleet.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza as propriedades de uma frota

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

Saída:

```
Arn                  : arn:aws:appstream:us-east-1:123456789012:fleet/
PowershellFleet
ComputeCapacityStatus : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime          : 4/24/2019 8:39:41 AM
Description           : PowershellFleet
DisconnectTimeoutInSeconds : 950
```



```
DisplayName           : PowershellFleet
DomainJoinInfo        :
EnableDefaultInternetAccess : True
FleetErrors           : {}
FleetType             : ON_DEMAND
IamRoleArn            :
IdleDisconnectTimeoutInSeconds : 900
ImageArn              : arn:aws:appstream:us-east-1:123456789012:image/
Powershell
ImageName             : Powershell
InstanceType          : stream.standard.medium
MaxUserDurationInSeconds : 57600
Name                  : PowershellFleet
State                 : STOPPED
VpcConfig             : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes da API, consulte [UpdateFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-APSIImagePermission

O código de exemplo a seguir mostra como usar Update-APSIImagePermission.

### Ferramentas para PowerShell

Exemplo 1: esta amostra compartilha uma AppStream imagem com outra conta

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

- Para obter detalhes da API, consulte [UpdateImagePermission](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-APSSStack

O código de exemplo a seguir mostra como usar Update-APSSStack.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza (ativa) a persistência das configurações do aplicativo e as pastas pessoais em uma pilha

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

### Saída:

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

- Para obter detalhes da API, consulte [UpdateStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Aurora usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Aurora.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Get-RDSOrderableDBInstanceOption

O código de exemplo a seguir mostra como usar `Get-RDSOrderableDBInstanceOption`.

#### Ferramentas para PowerShell

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

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

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

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

- Para obter detalhes da API, consulte [DescribeOrderableDB InstanceOptions](#) in AWS Tools for PowerShell Cmdlet Reference.

## Exemplos de Auto Scaling usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

### **Add-ASLoadBalancer**

O código de exemplo a seguir mostra como usar `Add-ASLoadBalancer`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa o balanceador de carga especificado ao grupo de Auto Scaling especificado.

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

- Para obter detalhes da API, consulte [AttachLoadBalancers](#) em Referência de AWS Tools for PowerShell cmdlet.

### **Complete-ASLifecycleAction**

O código de exemplo a seguir mostra como usar `Complete-ASLifecycleAction`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo conclui a ação do ciclo de vida especificada.

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

- Para obter detalhes da API, consulte [CompleteLifecycleAction](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-ASMetricsCollection

O código de exemplo a seguir mostra como usar `Disable-ASMetricsCollection`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o monitoramento das métricas especificadas para o grupo de Auto Scaling especificado.

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

Exemplo 2: Este exemplo desativa o monitoramento de todas as métricas do grupo de Auto Scaling especificado.

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

- Para obter detalhes da API, consulte [DisableMetricsCollection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-ASInstance

O código de exemplo a seguir mostra como usar `Dismount-ASInstance`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo separa a instância especificada do grupo de Auto Scaling especificado e diminui a capacidade desejada para que o Auto Scaling não execute uma instância substituta.

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

Saída:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e  
AutoScalingGroupName : my-asg
```

```

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

```

Exemplo 2: Este exemplo separa a instância especificada do grupo de Auto Scaling especificado sem diminuir a capacidade desejada. O Auto Scaling inicia uma instância substituta.

```

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

```

Saída:

```

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

```

- Para obter detalhes da API, consulte [DetachInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-ASLoadBalancer

O código de exemplo a seguir mostra como usar `Dismount-ASLoadBalancer`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo separa o balanceador de carga especificado do grupo de Auto Scaling especificado.

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

- Para obter detalhes da API, consulte [DetachLoadBalancers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-ASMetricsCollection

O código de exemplo a seguir mostra como usar `Enable-ASMetricsCollection`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo permite o monitoramento das métricas especificadas para o grupo de Auto Scaling especificado.

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

Exemplo 2: Este exemplo permite o monitoramento de todas as métricas do grupo de Auto Scaling especificado.

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

- Para obter detalhes da API, consulte [EnableMetricsCollection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enter-ASStandby

O código de exemplo a seguir mostra como usar `Enter-ASStandby`.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo coloca a instância especificada no modo de espera e diminui a capacidade desejada para que o Auto Scaling não execute uma instância substituta.

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

### Saída:

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

Exemplo 2: Esse exemplo coloca a instância especificada no modo de espera sem diminuir a capacidade desejada. O Auto Scaling inicia uma instância substituta.

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

### Saída:

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



- Para obter detalhes da API, consulte [EnterStandby](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exit-ASStandby

O código de exemplo a seguir mostra como usar `Exit-ASStandby`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo move a instância especificada para fora do modo de espera.

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

Saída:

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

- Para obter detalhes da API, consulte [ExitStandby](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASAccountLimit

O código de exemplo a seguir mostra como usar `Get-ASAccountLimit`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve os limites de recursos do Auto Scaling para sua AWS conta.

```
Get-ASAccountLimit
```

**Saída:**

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

- Para obter detalhes da API, consulte [DescribeAccountLimits](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-ASAdjustmentType**

O código de exemplo a seguir mostra como usar `Get-ASAdjustmentType`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve os tipos de ajuste que são compatíveis com o Auto Scaling.

```
Get-ASAdjustmentType
```

**Saída:**

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

- Para obter detalhes da API, consulte [DescribeAdjustmentTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-ASAutoScalingGroup**

O código de exemplo a seguir mostra como usar `Get-ASAutoScalingGroup`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo lista os nomes dos seus grupos do Auto Scaling.

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

**Saída:**

```
AutoScalingGroupName
```

```
-----
```

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

Exemplo 2: Este exemplo descreve o grupo de Auto Scaling especificado.

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

Saída:

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

Exemplo 3: Este exemplo descreve os dois grupos de Auto Scaling especificados.

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

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling para o grupo de Auto Scaling especificado.

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

Exemplo 5: Este exemplo descreve todos os seus grupos de Auto Scaling.

```
Get-ASAutoScalingGroup
```

Exemplo 6: Este exemplo descreve todos os seus grupos de Auto Scaling, em lotes de 10.

```
$nextToken = $null
do {
    Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

Exemplo 7: Este LaunchTemplate exemplo descreve o grupo de Auto Scaling especificado. Este exemplo pressupõe que as “Opções de compra de instância” estejam definidas como “Aderir ao modelo de lançamento”. Caso essa opção esteja definida como “Combinar opções de compra e tipos de instância”, LaunchTemplate pode ser acessada usando "MixedInstancesPolicy.LaunchTemplate" propriedade.

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

Saída:

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

- Para obter detalhes da API, consulte [DescribeAutoScalingGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASAutoScalingInstance

O código de exemplo a seguir mostra como usar Get-ASAutoScalingInstance.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os IDs das suas instâncias do Auto Scaling.

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

Saída:

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

Exemplo 2: Este exemplo descreve a instância de Auto Scaling especificada.

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

Saída:

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

Exemplo 3: Este exemplo descreve as duas instâncias do Auto Scaling especificadas.

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

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling para o grupo de Auto Scaling especificado.

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

Exemplo 5: Este exemplo descreve todas as suas instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance
```

Exemplo 6: Este exemplo descreve todas as suas instâncias do Auto Scaling, em lotes de 10.

```
$nextToken = $null
do {
    Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeAutoScalingInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASAutoScalingNotificationType

O código de exemplo a seguir mostra como usar `Get-ASAutoScalingNotificationType`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os tipos de notificação compatíveis com o Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Saída:

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

- Para obter detalhes da API, consulte [DescribeAutoScalingNotificationTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASLaunchConfiguration

O código de exemplo a seguir mostra como usar `Get-ASLaunchConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os nomes das suas configurações de lançamento.

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

**Saída:**

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

Exemplo 2: Este exemplo descreve a configuração de lançamento especificada.

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

**Saída:**

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

Exemplo 3: Este exemplo descreve as duas configurações de inicialização especificadas.

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

Exemplo 4: Este exemplo descreve todas as suas configurações de lançamento.

```
Get-ASLaunchConfiguration
```

Exemplo 5: Este exemplo descreve todas as suas configurações de lançamento, em lotes de 10.

```
$nextToken = $null
do {
    Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeLaunchConfigurations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASLifecycleHook

O código de exemplo a seguir mostra como usar `Get-ASLifecycleHook`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o gancho do ciclo de vida especificado.

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

Saída:

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



Exemplo 2: Este exemplo descreve todos os ganchos do ciclo de vida do grupo de Auto Scaling especificado.

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

Exemplo 3: Este exemplo descreve todos os ganchos do ciclo de vida de todos os seus grupos de Auto Scaling.

```
Get-ASLifecycleHook
```

- Para obter detalhes da API, consulte [DescribeLifecycleHooks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASLifecycleHookType

O código de exemplo a seguir mostra como usar `Get-ASLifecycleHookType`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os tipos de ganchos do ciclo de vida suportados pelo Auto Scaling.

```
Get-ASLifecycleHookType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCHING  
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Para obter detalhes da API, consulte [DescribeLifecycleHookTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASLoadBalancer

O código de exemplo a seguir mostra como usar `Get-ASLoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve os balanceadores de carga para o grupo de Auto Scaling especificado.

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

Saída:

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

- Para obter detalhes da API, consulte [DescribeLoadBalancers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASMetricCollectionType

O código de exemplo a seguir mostra como usar `Get-ASMetricCollectionType`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os tipos de coleção de métricas que são compatíveis com o Auto Scaling.

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

Saída:

```
Metric
-----
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Exemplo 2: Este exemplo lista as granularidades correspondentes.

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

Saída:

```
Granularity
-----
1Minute
```

- Para obter detalhes da API, consulte [DescribeMetricCollectionTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASNotificationConfiguration

O código de exemplo a seguir mostra como usar `Get-ASNotificationConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as ações de notificação associadas ao grupo de Auto Scaling especificado.

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

Saída:

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

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

Exemplo 2: Este exemplo descreve as ações de notificação associadas a todos os seus grupos do Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Para obter detalhes da API, consulte [DescribeNotificationConfigurations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASPolicy

O código de exemplo a seguir mostra como usar `Get-ASPolicy`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve todas as políticas do grupo de Auto Scaling especificado.

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

Saída:

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

Exemplo 2: Este exemplo descreve as políticas especificadas para o grupo de Auto Scaling especificado.

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

Exemplo 3: Este exemplo descreve todas as políticas de todos os seus grupos de Auto Scaling.

```
Get-ASPolicy
```

- Para obter detalhes da API, consulte [DescribePolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASScalingActivity

O código de exemplo a seguir mostra como usar `Get-ASScalingActivity`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as atividades de escalonamento das últimas seis semanas para o grupo de Auto Scaling especificado.

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

### Saída:

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

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
                      AutoScalingGroup changing the desired capacity
                      from 0 to 1. At 2015-11-20T22:57:58Z an instance was
                      started in response to a difference betwe
                      en desired and actual capacity, increasing the capacity from
                      0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
                      ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress              : 100
StartTime             : 11/20/2015 2:57:59 PM
StatusCode            : Successful
StatusMessage        :
```

Exemplo 2: Este exemplo descreve a atividade de escalabilidade especificada.

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

Exemplo 3: Este exemplo descreve as atividades de escalonamento das últimas seis semanas para todos os seus grupos de Auto Scaling.

```
Get-ASScalingActivity
```

- Para obter detalhes da API, consulte [DescribeScalingActivities](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASScalingProcessType

O código de exemplo a seguir mostra como usar Get-ASScalingProcessType.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os tipos de processo que são suportados pelo Auto Scaling.

```
Get-ASScalingProcessType
```

Saída:

```
ProcessName
-----
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Para obter detalhes da API, consulte [DescribeScalingProcessTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASScheduledAction

O código de exemplo a seguir mostra como usar Get-ASScheduledAction.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as ações de escalonamento programadas para o grupo de Auto Scaling especificado.

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

Saída:

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

Exemplo 2: Este exemplo descreve as ações de escalonamento programadas especificadas.

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

Exemplo 3: Este exemplo descreve as ações de escalonamento programadas que começam no horário especificado.

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

Exemplo 4: Este exemplo descreve as ações de escalonamento programadas que terminam no horário especificado.

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

Exemplo 5: Este exemplo descreve as ações de escalonamento programadas para todos os seus grupos de Auto Scaling.

```
Get-ASScheduledAction
```

- Para obter detalhes da API, consulte [DescribeScheduledActions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASTag

O código de exemplo a seguir mostra como usar Get-ASTag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as tags com um valor-chave de 'MyTag' ou 'MyTag2'. Os valores possíveis para o nome do filtro são auto-scaling-group ", 'chave', 'valor' e 'propagate-at-launch'. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

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

Saída:

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

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

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.



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

Exemplo 3: Este exemplo descreve todas as tags de todos os seus grupos de Auto Scaling.

```
Get-ASTag
```

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASTerminationPolicyType

O código de exemplo a seguir mostra como usar `Get-ASTerminationPolicyType`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as políticas de rescisão suportadas pelo Auto Scaling.

```
Get-ASTerminationPolicyType
```

Saída:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Para obter detalhes da API, consulte [DescribeTerminationPolicyTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Mount-ASInstance

O código de exemplo a seguir mostra como usar `Mount-ASInstance`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa a instância especificada ao grupo de Auto Scaling especificado. O Auto Scaling aumenta automaticamente a capacidade desejada do grupo Auto Scaling.

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

- Para obter detalhes da API, consulte [AttachInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ASAutoScalingGroup

O código de exemplo a seguir mostra como usar New-ASAutoScalingGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um grupo de Auto Scaling com o nome e os atributos especificados. A capacidade padrão desejada é o tamanho mínimo. Portanto, esse grupo de Auto Scaling inicia duas instâncias, uma em cada uma das duas zonas de disponibilidade especificadas.

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

- Para obter detalhes da API, consulte [CreateAutoScalingGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ASLaunchConfiguration

O código de exemplo a seguir mostra como usar New-ASLaunchConfiguration.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma configuração de inicialização chamada 'my-lc'. As instâncias do EC2 lançadas pelos grupos do Auto Scaling que usam essa configuração de execução usam o tipo de instância, a AMI, o grupo de segurança e a função do IAM especificados.

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

- Para obter detalhes da API, consulte [CreateLaunchConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASAutoScalingGroup

O código de exemplo a seguir mostra como usar Remove-ASAutoScalingGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o grupo de Auto Scaling especificado se ele não tiver instâncias em execução. A confirmação será solicitada antes que a operação continue.

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

Saída:

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

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

Exemplo 3: Este exemplo exclui o grupo de Auto Scaling especificado e encerra todas as instâncias em execução que ele contém.

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

- Para obter detalhes da API, consulte [DeleteAutoScalingGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASLaunchConfiguration

O código de exemplo a seguir mostra como usar Remove-ASLaunchConfiguration.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a configuração de inicialização especificada se ela não estiver anexada a um grupo de Auto Scaling. A confirmação será solicitada antes que a operação continue.

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

Saída:

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

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

- Para obter detalhes da API, consulte [DeleteLaunchConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASLifecycleHook

O código de exemplo a seguir mostra como usar Remove-ASLifecycleHook.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o gancho de ciclo de vida especificado para o grupo de Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

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

Saída:

```
Confirm
```

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

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

- Para obter detalhes da API, consulte [DeleteLifecycleHook](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASNotificationConfiguration

O código de exemplo a seguir mostra como usar Remove-ASNotificationConfiguration.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a ação de notificação especificada. A confirmação será solicitada antes que a operação continue.

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

Saída:

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

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

- Para obter detalhes da API, consulte [DeleteNotificationConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASPolicy

O código de exemplo a seguir mostra como usar `Remove-ASPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a política especificada para o grupo de Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

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

Saída:

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

Exemplo 2: Se você especificar o parâmetro `Force`, não será solicitada a confirmação antes que a operação continue.

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

- Para obter detalhes da API, consulte [DeletePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASScheduledAction

O código de exemplo a seguir mostra como usar `Remove-ASScheduledAction`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a ação agendada especificada para o grupo de Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction"
```

Saída:

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction  
"myScheduledAction" -Force
```

- Para obter detalhes da API, consulte [DeleteScheduledAction](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ASTag

O código de exemplo a seguir mostra como usar Remove-ASTag.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a tag especificada do grupo de Auto Scaling especificado. A confirmação será solicitada antes que a operação continue. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag" } )
```

Saída:

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

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes que a operação continue.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
Key="myTag" } ) -Force
```

Exemplo 3: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte [DeleteTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Resume-ASProcess

O código de exemplo a seguir mostra como usar Resume-ASProcess.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retoma o processo de Auto Scaling especificado para o grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo retoma todos os processos de Auto Scaling suspensos para o grupo de Auto Scaling especificado.



```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte [ResumeProcesses](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ASDesiredCapacity

O código de exemplo a seguir mostra como usar Set-ASDesiredCapacity.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo define o tamanho do grupo de Auto Scaling especificado.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Exemplo 2: Este exemplo define o tamanho do grupo de Auto Scaling especificado e aguarda a conclusão do período de espera antes de escalar para o novo tamanho.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown $true
```

- Para obter detalhes da API, consulte [SetDesiredCapacity](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ASInstanceHealth

O código de exemplo a seguir mostra como usar Set-ASInstanceHealth.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo define o status da instância especificada como “Não íntegra”, retirando-a de serviço. O Auto Scaling encerra e substitui a instância.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Exemplo 2: Este exemplo define o status da instância especificada como 'Saudável', mantendo-a em serviço. Qualquer período de carência de verificação de saúde para o grupo Auto Scaling não é respeitado.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -  
ShouldRespectGracePeriod $false
```

- Para obter detalhes da API, consulte [SetInstanceHealth](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ASInstanceProtection

O código de exemplo a seguir mostra como usar Set-ASInstanceProtection.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo ativa a proteção de instância para a instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $true
```

Exemplo 2: Este exemplo desativa a proteção de instância para a instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -  
ProtectedFromScaleIn $false
```

- Para obter detalhes da API, consulte [SetInstanceProtection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ASTag

O código de exemplo a seguir mostra como usar Set-ASTag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona uma única tag ao grupo de Auto Scaling especificado. A chave da tag é 'myTag' e o valor da tag é 'myTagValue'. O Auto Scaling propaga essa tag para as instâncias subsequentes do EC2 lançadas pelo grupo Auto Scaling. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";  
Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar `New-Object` para criar a tag para o parâmetro `Tag`.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Para obter detalhes da API, consulte [CreateOrUpdateTag](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-ASPolicy

O código de exemplo a seguir mostra como usar `Start-ASPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo executa a política especificada para o grupo de Auto Scaling especificado.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Exemplo 2: Este exemplo executa a política especificada para o grupo de Auto Scaling especificado, após aguardar a conclusão do período de espera.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Para obter detalhes da API, consulte [ExecutePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-ASInstanceInAutoScalingGroup

O código de exemplo a seguir mostra como usar `Stop-ASInstanceInAutoScalingGroup`.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo encerra a instância especificada e diminui a capacidade desejada de seu grupo de Auto Scaling para que o Auto Scaling não execute uma instância substituta.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause               : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of  
service in response to a user  
                    request, shrinking the capacity from 2 to 1.  
Description         : Terminating EC2 instance: i-93633f9b  
Details             : {"Availability Zone":"us-west-2b","Subnet  
ID":"subnet-5264e837"}  
EndTime             :  
Progress            : 0  
StartTime           : 11/22/2015 8:09:03 AM  
StatusCode          : InProgress  
StatusMessage       :
```

Exemplo 2: Esse exemplo encerra a instância especificada sem diminuir a capacidade desejada do grupo Auto Scaling. O Auto Scaling inicia uma instância substituta.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -  
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5  
AutoScalingGroupName :  
Cause               : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of  
service in response to a user  
                    request.  
Description         : Terminating EC2 instance: i-93633f9b  
Details             : {"Availability Zone":"us-west-2b","Subnet  
ID":"subnet-5264e837"}  
EndTime             :  
Progress            : 0
```

```
StartTime      : 11/22/2015 8:09:03 AM
StatusCode     : InProgress
StatusMessage  :
```

- Para obter detalhes da API, consulte [TerminateInstanceInAutoScalingGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Suspend-ASProcess

O código de exemplo a seguir mostra como usar Suspend-ASProcess.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo suspende o processo de Auto Scaling especificado para o grupo de Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo suspende todos os processos de Auto Scaling para o grupo de Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte [SuspendProcesses](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-ASAutoScalingGroup

O código de exemplo a seguir mostra como usar Update-ASAutoScalingGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza o tamanho mínimo e máximo do grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Exemplo 2: Este exemplo atualiza o período de espera padrão do grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Exemplo 3: Este exemplo atualiza as zonas de disponibilidade do grupo de Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-west-2a", "us-west-2b")
```

Exemplo 4: Este exemplo atualiza o grupo de Auto Scaling especificado para usar as verificações de saúde do Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -HealthCheckGracePeriod 60
```

- Para obter detalhes da API, consulte [UpdateAutoScalingGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-ASLifecycleActionHeartbeat

O código de exemplo a seguir mostra como usar Write-ASLifecycleActionHeartbeat.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo registra uma pulsação para a ação de ciclo de vida especificada. Isso mantém a instância em um estado pendente até que você conclua a ação personalizada.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes da API, consulte [RecordLifecycleActionHeartbeat](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-ASLifecycleHook

O código de exemplo a seguir mostra como usar Write-ASLifecycleHook.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona o gancho de ciclo de vida especificado ao grupo de Auto Scaling especificado.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName  
"myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -  
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN  
"arn:aws:iam::123456789012:role/my-iam-role"
```

- Para obter detalhes da API, consulte [PutLifecycleHook](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-ASNotificationConfiguration

O código de exemplo a seguir mostra como usar Write-ASNotificationConfiguration.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo configura o grupo de Auto Scaling especificado para enviar uma notificação ao tópico do SNS especificado ao iniciar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
"autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-  
topic"
```

Exemplo 2: Este exemplo configura o grupo de Auto Scaling especificado para enviar uma notificação ao tópico do SNS especificado ao iniciar ou encerrar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType  
@"autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -  
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Para obter detalhes da API, consulte [PutNotificationConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-ASScalingPolicy

O código de exemplo a seguir mostra como usar Write-ASScalingPolicy.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona a política especificada ao grupo de Auto Scaling especificado. O tipo de ajuste especificado determina como interpretar o ScalingAdjustment parâmetro. Com

'ChangeInCapacity', um valor positivo aumenta a capacidade pelo número especificado de instâncias e um valor negativo diminui a capacidade pelo número especificado de instâncias.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType  
"ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Saída:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-  
e1d769fc24ef:autoScalingGroupName/my-asg  
:policyName/myScaleInPolicy
```

- Para obter detalhes da API, consulte [PutScalingPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-ASScheduledUpdateGroupAction

O código de exemplo a seguir mostra como usar Write-ASScheduledUpdateGroupAction.

Ferramentas para PowerShell

Exemplo 1: Este exemplo cria ou atualiza uma ação agendada única para alterar a capacidade desejada no horário de início especificado.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName  
"myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Para obter detalhes da API, consulte [PutScheduledUpdateGroupAction](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Budgets exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Budgets.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.



Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### New-BGTBudget

O código de exemplo a seguir mostra como usar New-BGTBudget.

### Ferramentas para PowerShell

Exemplo 1: cria um novo orçamento com as restrições orçamentárias e de tempo especificadas com notificações por e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @"user@domain.com"
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Para obter detalhes da API, consulte [CreateBudget](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Cloud9 exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Cloud9.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-C9EnvironmentData**

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentData`.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre os ambientes de desenvolvimento AWS Cloud9 especificados.

```
Get-C9EnvironmentData -EnvironmentId
685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Saída:

```
Arn          : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
```

```

Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh

```

Exemplo 2: Este exemplo obtém informações sobre o status do ciclo de vida do ambiente de desenvolvimento Cloud9 especificado AWS .

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Saída:

```

FailureResource Reason Status
-----
                                CREATED

```

- Para obter detalhes da API, consulte [DescribeEnvironments](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-C9EnvironmentList

O código de exemplo a seguir mostra como usar Get-C9EnvironmentList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista de identificadores de ambiente de AWS desenvolvimento Cloud9 disponíveis.

```
Get-C9EnvironmentList
```

**Saída:**

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Para obter detalhes da API, consulte [ListEnvironments](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-C9EnvironmentMembershipList**

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentMembershipList`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo obtém informações sobre os membros do ambiente do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX
```

**Saída:**

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId       : AIDAJ3BA602FMJWCXHEX

EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId       : AIDAJ3LOR0M0UXTBSU6EX
```

Exemplo 2: Este exemplo obtém informações sobre o proprietário do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -
Permission owner
```

**Saída:**

```
EnvironmentId : ffd88420d4824eeeeaaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOXTBSU6EX
```

Exemplo 3: Este exemplo obtém informações sobre o membro do ambiente especificado para vários ambientes de desenvolvimento do AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Saída:

```
EnvironmentId : ffd88420d4824eeeeaaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOXTBSU6EX
```

- Para obter detalhes da API, consulte [DescribeEnvironmentMemberships](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-C9EnvironmentStatus

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentStatus`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações de status para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Saída:

Message	Status
-----	-----
Environment is ready to use	ready

- Para obter detalhes da API, consulte [DescribeEnvironmentStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-C9EnvironmentEC2

O código de exemplo a seguir mostra como usar `New-C9EnvironmentEC2`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um ambiente de desenvolvimento AWS Cloud9 com as configurações especificadas, inicia uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e, em seguida, se conecta da instância ao ambiente.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
"My demonstration development environment." -InstanceType t2.micro -OwnerArn
arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Saída:

```
ffd88420d4824eeeeaaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte [CreateEnvironmentEc2](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-C9EnvironmentMembership

O código de exemplo a seguir mostra como usar `New-C9EnvironmentMembership`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona o membro do ambiente especificado ao ambiente de desenvolvimento do AWS Cloud9 especificado.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
-EnvironmentId ffd88420d4824eeeeaaa8a04bfde8cEX -Permission read-write
```

## Saída:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId       : AIDAJ3BA602FMJWCXHEX
```

- Para obter detalhes da API, consulte [CreateEnvironmentMembership](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-C9Environment

O código de exemplo a seguir mostra como usar Remove-C9Environment.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o ambiente de desenvolvimento AWS Cloud9 especificado. Se uma instância do Amazon EC2 estiver conectada ao ambiente, a instância também será encerrada.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX
```

- Para obter detalhes da API, consulte [DeleteEnvironment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-C9EnvironmentMembership

O código de exemplo a seguir mostra como usar Remove-C9EnvironmentMembership.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o membro do ambiente especificado do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX
```

- Para obter detalhes da API, consulte [DeleteEnvironmentMembership](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-C9Environment

O código de exemplo a seguir mostra como usar Update-C9Environment.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo altera as configurações especificadas do ambiente de desenvolvimento AWS Cloud9 existente especificado.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -Description "My changed demonstration development environment." -Name my-changed-demo-env
```

- Para obter detalhes da API, consulte [UpdateEnvironment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-C9EnvironmentMembership

O código de exemplo a seguir mostra como usar Update-C9EnvironmentMembership.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo altera as configurações do membro do ambiente existente especificado para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser -EnvironmentId ffd88420d4824eeeeaea8a04bfde8cEX -Permission read-only
```

### Saída:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCXHEX
```

- Para obter detalhes da API, consulte [UpdateEnvironmentMembership](#) em Referência de AWS Tools for PowerShell cmdlet.



# AWS CloudFormation exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS CloudFormation.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Get-CFNStack

O código de exemplo a seguir mostra como usar Get-CFNStack.

### Ferramentas para PowerShell

Exemplo 1: retorna um conjunto de instâncias de pilha descrevendo todas as pilhas do usuário.

```
Get-CFNStack
```

Exemplo 2: retorna uma instância de pilha descrevendo a pilha especificada

```
Get-CFNStack -StackName "myStack"
```

Exemplo 3: retorna um conjunto de instâncias de pilha descrevendo todas as pilhas do usuário usando paginação manual. O token inicial para a próxima página é recuperado após cada chamada, sendo que \$null indica que não há mais detalhes a serem recuperados.

```
$nextToken = $null
```

```
do {  
    Get-CFNStack -NextToken $nextToken  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeStacks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNStackEvent

O código de exemplo a seguir mostra como usar Get-CFNStackEvent.

### Ferramentas para PowerShell

Exemplo 1: retorna todos os eventos relacionados à pilha especificada.

```
Get-CFNStackEvent -StackName "myStack"
```

Exemplo 2: retorna todos os eventos relacionados à pilha especificada usando paginação manual a partir do token especificado. O token inicial para a próxima página é recuperado após cada chamada, sendo que \$null indica que não há mais eventos a serem recuperados.

```
$nextToken = $null  
do {  
    Get-CFNStack -StackName "myStack" -NextToken $nextToken  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeStackEvents](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNStackResource

O código de exemplo a seguir mostra como usar Get-CFNStackResource.

### Ferramentas para PowerShell

Exemplo 1: retorna a descrição de um recurso identificado no modelo associado à pilha especificada pelo ID lógico "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Para obter detalhes da API, consulte [DescribeStackResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNStackResourceList

O código de exemplo a seguir mostra como usar `Get-CFNStackResourceList`.

### Ferramentas para PowerShell

Exemplo 1: retorna as descrições dos AWS recursos de até 100 recursos associados à pilha especificada. Para obter detalhes de todos os recursos associados a uma pilha, use o `Get-CFNStackResourceSummary`, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -StackName "myStack"
```

Exemplo 2: retorna a descrição da instância do Amazon EC2 identificada no modelo associado à pilha especificada pelo ID lógico "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Exemplo 3: retorna a descrição de até 100 recursos associados à pilha contendo uma instância do Amazon EC2 identificada pelo ID de instância "i-123456". Para obter detalhes de todos os recursos associados a uma pilha, use o `Get-CFNStackResourceSummary`, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Exemplo 4: retorna a descrição da instância do Amazon EC2 identificada pelo ID lógico "Ec2Instance" no modelo de uma pilha. A pilha é identificada usando o ID de recurso físico de um recurso que ela contém; nesse caso, também uma instância do Amazon EC2 com o ID de instância "i-123456". Um outro recurso físico também poderia ser usado para identificar a pilha, dependendo do conteúdo do modelo, por exemplo, um bucket do Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId "Ec2Instance"
```

- Para obter detalhes da API, consulte [DescribeStackResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNStackResourceSummary

O código de exemplo a seguir mostra como usar Get-CFNStackResourceSummary.

### Ferramentas para PowerShell

Exemplo 1: retorna as descrições de todos os recursos associados à pilha especificada.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

Exemplo 2: retorna as descrições de todos os recursos da associados à pilha especificada usando a paginação manual dos resultados. O token inicial para a próxima página é recuperado após cada chamada, sendo que \$null indica que não há mais detalhes a serem recuperados.

```
$nextToken = $null
do {
    Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [ListStackResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNStackSummary

O código de exemplo a seguir mostra como usar Get-CFNStackSummary.

### Ferramentas para PowerShell

Exemplo 1: retorna informações resumidas de todas as pilhas.

```
Get-CFNStackSummary
```

Exemplo 2: retorna informações resumidas de todas as pilhas que estão sendo criadas no momento.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Exemplo 3: retorna informações resumidas de todas as pilhas que estão sendo criadas ou atualizadas no momento.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

Exemplo 4: retorna informações resumidas de todas as pilhas que estão sendo criadas ou atualizadas no momento usando a paginação manual dos resultados. O token inicial para a próxima página é recuperado após cada chamada, sendo que \$null indica que não há mais detalhes a serem recuperados.

```
$nextToken = $null
do {
    Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
    "UPDATE_IN_PROGRESS") -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [ListStacks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFNTemplate

O código de exemplo a seguir mostra como usar Get-CFNTemplate.

### Ferramentas para PowerShell

Exemplo 1: retorna o modelo associado à pilha especificada.

```
Get-CFNTemplate -StackName "myStack"
```

- Para obter detalhes da API, consulte [GetTemplate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Measure-CFNTemplateCost

O código de exemplo a seguir mostra como usar Measure-CFNTemplateCost.

## Ferramentas para PowerShell

Exemplo 1: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é obtido da URL do Amazon S3 especificada e do único parâmetro de personalização aplicado. O parâmetro também pode ser especificado usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }
```

Exemplo 2: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido e os parâmetros de personalização aplicados (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, " e 'KeyName')InstanceType. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
ParameterValue="myKeyPairName" }, `
                                     @{ ParameterKey="InstanceType";
ParameterValue="m1.large" })
```

Exemplo 3: usa New-Object para criar o conjunto de parâmetros do modelo e retorna uma URL de calculadora mensal AWS simples com uma string de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido, com parâmetros de personalização (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, " e KeyName ")InstanceType.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
```

```
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter @( $p1,
    $p2 )
```

- Para obter detalhes da API, consulte [EstimateTemplateCost](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CFNStack

O código de exemplo a seguir mostra como usar New-CFNStack.

### Ferramentas para PowerShell

Exemplo 1: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela não será revertida.

```
New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
    @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
    -DisableRollback $true
```

Exemplo 2: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela será revertida.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
```

```
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
    -TemplateBody "{TEMPLATE CONTENT HERE}" `
    -Parameter @( $p1, $p2 ) `
    -OnFailure "ROLLBACK"
```

Exemplo 3: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback \$false).

```
New-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback \$false). Os AENs de notificação especificados receberão os eventos relacionados à pilha publicados.

```
New-CFNStack -StackName "myStack" `
    -TemplateURL https://s3.amazonaws.com/mytemplates/templatefile.template
    -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
    -NotificationARN @( "arn1", "arn2" )
```

- Para obter detalhes da API, consulte [CreateStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CFNStack

O código de exemplo a seguir mostra como usar Remove-CFNStack.



## Ferramentas para PowerShell

Exemplo 1: exclui a pilha especificada.

```
Remove-CFNStack -StackName "myStack"
```

- Para obter detalhes da API, consulte [DeleteStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Resume-CFNUpdateRollback

O código de exemplo a seguir mostra como usar Resume-CFNUpdateRollback.

## Ferramentas para PowerShell

Exemplo 1: continua a reversão da pilha nomeada, que deve estar no estado "UPDATE\_ROLLBACK\_FAILED". Se a reversão contínua for bem-sucedida, a pilha entrará no estado "UPDATE\_ROLLBACK\_COMPLETE".

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Para obter detalhes da API, consulte [ContinueUpdateRollback](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-CFNUpdateStack

O código de exemplo a seguir mostra como usar Stop-CFNUpdateStack.

## Ferramentas para PowerShell

Exemplo 1: cancela uma atualização na pilha especificada.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Para obter detalhes da API, consulte [CancelUpdateStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Test-CFNStack

O código de exemplo a seguir mostra como usar Test-CFNStack.

## Ferramentas para PowerShell

Exemplo 1: testa se a pilha atingiu um dos estados UPDATE\_ROLLBACK\_COMPLETE, CREATE\_COMPLETE, ROLLBACK\_COMPLETE ou UPDATE\_COMPLETE.

```
Test-CFNStack -StackName MyStack
```

Saída:

```
False
```

Exemplo 2: testa se a pilha atingiu o status UPDATE\_COMPLETE ou UPDATE\_ROLLBACK\_COMPLETE.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Saída:

```
True
```

- Para obter detalhes da API, consulte [Test-CfnStack em Cmdlet Reference](#).AWS Tools for PowerShell

## Test-CFNTemplate

O código de exemplo a seguir mostra como usar Test-CFNTemplate.

## Ferramentas para PowerShell

Exemplo 1: valida o conteúdo do modelo especificado. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Exemplo 2: valida o modelo especificado acessado por meio de uma URL do Amazon S3. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/mytemplates/  
templatefile.template
```

- Para obter detalhes da API, consulte [ValidateTemplate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CFNStack

O código de exemplo a seguir mostra como usar Update-CFNStack.

### Ferramentas para PowerShell

Exemplo 1: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV1" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 2: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
                             @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 3: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV2" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: atualiza a pilha "myStack" com o modelo obtido no Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados

no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 5: atualiza a pilha "myStack" que, este exemplo pressupõe conter recursos do IAM, com o modelo obtido do Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. As pilhas contendo recursos do IAM exigem que você especifique o parâmetro -Capabilities "CAPABILITY\_IAM", caso contrário, a atualização falhará com um erro ". InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Para obter detalhes da API, consulte [UpdateStack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Wait-CFNStack

O código de exemplo a seguir mostra como usar Wait-CFNStack.

### Ferramentas para PowerShell

Exemplo 1: testa se a pilha atingiu um dos estados UPDATE\_ROLLBACK\_COMPLETE, CREATE\_COMPLETE, ROLLBACK\_COMPLETE ou UPDATE\_COMPLETE. Se a pilha não estiver em um dos estados, o comando dorme por dois segundos antes de testar o status novamente. Isso é repetido até que a pilha alcance um dos estados solicitados ou o período de tempo limite padrão de 60 segundos termine. Se o período de tempo limite for excedido, uma

exceção será lançada. Se a pilha atingir um dos estados solicitados dentro do período de tempo limite, ela será retornada ao pipeline.

```
$stack = Wait-CFNStack -StackName MyStack
```

Exemplo 2: Este exemplo espera por um total de 5 minutos (300 segundos) para que a pilha alcance qualquer um dos estados especificados. Neste exemplo, o estado é atingido antes do tempo limite e, portanto, o objeto da pilha é retornado ao pipeline.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status  
CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Saída:

```
Capabilities      : {CAPABILITY_IAM}  
ChangeSetId      :  
CreationTime     : 6/1/2017 9:29:33 AM  
Description      : AWS CloudFormation Sample Template  
ec2_instance_with_instance_profile: Create an EC2 instance with an associated  
instance profile. **WARNING** This template creates one or more Amazon EC2  
instances and an Amazon SQS queue. You will be billed for the  
AWS resources used if you create a stack from this template.  
DisableRollback  : False  
LastUpdatedTime  : 1/1/0001 12:00:00 AM  
NotificationARNs : {}  
Outputs          : {}  
Parameters       : {}  
RoleARN          :  
StackId          : arn:aws:cloudformation:us-west-2:123456789012:stack/  
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1  
StackName        : MyStack  
StackStatus      : CREATE_COMPLETE  
StackStatusReason :  
Tags             : {}  
TimeoutInMinutes : 0
```

Exemplo 3: Este exemplo mostra a saída de erro quando uma pilha não atinge um dos estados solicitados dentro do período de tempo limite (nesse caso, o período padrão de 60 segundos).

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

**Saída:**

```

Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
stack MyStack in region us-west-2 to reach one of state(s):
UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~
+ CategoryInfo          : InvalidOperation:
(Amazon.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
InvalidOperationException
+ FullyQualifiedErrorId :
InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet

```

- Para obter detalhes da API, consulte [Wait-CfnStack em Cmdlet Reference](#).AWS Tools for PowerShell

## CloudFront exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CloudFront.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-CFCloudFrontOriginAccessIdentity**

O código de exemplo a seguir mostra como usar Get-CFCloudFrontOriginAccessIdentity.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna uma identidade de acesso de CloudFront origem específica da Amazon, especificada pelo parâmetro `-Id`. Embora o parâmetro `-Id` não seja obrigatório, se você não o especificar, nenhum resultado será retornado.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Saída:

```

CloudFrontOriginAccessIdentityConfig      Id
-----
S3CanonicalUserId
-----
Amazon.CloudFront.Model.CloudFrontOr...  E3XXXXXXXXXXRT
4b6e...
```

- Para obter detalhes da API, consulte [GetCloudFrontOriginAccessIdentity](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-CFCloudFrontOriginAccessIdentityConfig

O código de exemplo a seguir mostra como usar `Get-CFCloudFrontOriginAccessIdentityConfig`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna informações de configuração sobre uma única identidade de acesso de CloudFront origem da Amazon, especificada pelo parâmetro `-Id`. Ocorrem erros se nenhum parâmetro `-Id` for especificado.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Saída:

```

CallerReference                               Comment
-----
mycallerreference: 2/1/2011 1:16:32 PM        Caller reference:
2/1/2011 1:16:32 PM
```

- Para obter detalhes da API, consulte [GetCloudFrontOriginAccessIdentityConfigem](#) Referência de AWS Tools for PowerShell cmdlet.

## Get-CFCloudFrontOriginAccessIdentityList

O código de exemplo a seguir mostra como usar Get-CFCloudFrontOriginAccessIdentityList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna uma lista de identidades de acesso de CloudFront origem da Amazon. Como o MaxItem parâmetro - especifica um valor de 2, os resultados incluem duas identidades.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Saída:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXXX9B
Quantity    : 2
```

- Para obter detalhes da API, consulte [ListCloudFrontOriginAccessIdentities](#)em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFDistribution

O código de exemplo a seguir mostra como usar Get-CFDistribution.

### Ferramentas para PowerShell

Exemplo 1: recupera as informações de uma distribuição específica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte [GetDistribution](#)em Referência de AWS Tools for PowerShell cmdlet.



## Get-CFDistributionConfig

O código de exemplo a seguir mostra como usar `Get-CFDistributionConfig`.

### Ferramentas para PowerShell

Exemplo 1: recupera a configuração de uma distribuição específica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte [GetDistributionConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFDistributionList

O código de exemplo a seguir mostra como usar `Get-CFDistributionList`.

### Ferramentas para PowerShell

Exemplo 1: retorno de distribuições.

```
Get-CFDistributionList
```

- Para obter detalhes da API, consulte [ListDistributions](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CFDistribution

O código de exemplo a seguir mostra como usar `New-CFDistribution`.

### Ferramentas para PowerShell

Exemplo 1: Cria uma CloudFront distribuição básica, configurada com registro e armazenamento em cache.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "ps-cmdlet-sample.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3originConfig = New-Object Amazon.CloudFront.Model.S3originConfig
$origin.S3originConfig.OriginAccessIdentity = ""
New-CFDistribution `
    -DistributionConfig_Enabled $true `
```

```

-DistributionConfig_Comment "Test distribution" `
-Origins_Item $origin `
-Origins_Quantity 1 `
-Logging_Enabled $true `
-Logging_IncludeCookie $true `
-Logging_Bucket ps-cmdlet-sample-logging.s3.amazonaws.com `
-Logging_Prefix "help/" `
-DistributionConfig_CallerReference Client1 `
-DistributionConfig_DefaultRootObject index.html `
-DefaultCacheBehavior_TargetOriginId $origin.Id `
-ForwardedValues_QueryString $true `
-Cookies_Forward all `
-WhitelistedNames_Quantity 0 `
-TrustedSigners_Enabled $false `
-TrustedSigners_Quantity 0 `
-DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
-DefaultCacheBehavior_MinTTL 1000 `
-DistributionConfig_PriceClass "PriceClass_All" `
-CacheBehaviors_Quantity 0 `
-Aliases_Quantity 0

```

- Para obter detalhes da API, consulte [CreateDistribution](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CFInvalidation

O código de exemplo a seguir mostra como usar New-CFInvalidation.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma nova invalidação em uma distribuição com um ID de EXAMPLNSTXAXE. CallerReference É um ID exclusivo escolhido pelo usuário; nesse caso, é usado um carimbo de data/hora representando 15 de maio de 2019 às 9h. A variável \$Paths armazena três caminhos para arquivos de imagem e mídia que o usuário não deseja como parte do cache distribuído. O valor do parâmetro -Paths\_Quantity é o número total de caminhos especificados no parâmetro -Paths\_Item.

```

$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLNSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
3

```

**Saída:**

```

Invalidation                               Location
-----
Amazon.CloudFront.Model.Invalidations https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLEN5TAXE/invalidation/EXAMPLE8N0K9H

```

- Para obter detalhes da API, consulte [CreateInvalidation](#) em Referência de AWS Tools for PowerShell cmdlet.

**New-CFSignedCookie**

O código de exemplo a seguir mostra como usar `New-CFSignedCookie`.

**Ferramentas para PowerShell**

Exemplo 1: cria um cookie assinado para o recurso especificado usando uma política predefinida. O cookie será válido por um ano.

```

$params = @{
  "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params

```

**Saída:**

```

Expires
-----
[CloudFront-Expires, 1472227284]

```

Exemplo 2: cria um cookie assinado para os recursos especificados usando uma política personalizada. O cookie será válido em 24 horas e expirará uma semana depois.

```

$start = (Get-Date).AddHours(24)
$params = @{

```

```
"ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
"KeyPairId"="AKIAIOSFODNN7EXAMPLE"
"PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
"ExpiresOn"=$start.AddDays(7)
  "ActiveFrom"=$start
}
```

```
New-CFSignedCookie @params
```

### Saída:

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

Exemplo 3: cria um cookie assinado para os recursos especificados usando uma política personalizada. O cookie será válido em 24 horas e expirará uma semana depois. O acesso aos recursos é restrito ao intervalo de IP especificado.

```
$start = (Get-Date).AddHours(24)
$params = @{
  "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
  "IpRange"="192.0.2.0/24"
}
```

```
New-CFSignedCookie @params
```

### Saída:

```
Policy
-----
[CloudFront-Policy, eyJTd...wIjo...
```

- Para obter detalhes da API, consulte [New-CF SignedCookie em AWS Tools for PowerShell Cmdlet Reference](#).

## New-CFSignedUrl

O código de exemplo a seguir mostra como usar `New-CFSignedUrl`.

### Ferramentas para PowerShell

Exemplo 1: cria uma URL assinada para o recurso especificado usando uma política padronizada. O URL será válido por uma hora. Um objeto `System.Uri` contendo o URL assinado é emitido para o pipeline.

```
$params = @{
  "ResourceUri"="https://cdn.example.com/index.html"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Exemplo 2: cria uma URL assinada para o recurso especificado usando uma política personalizada. O URL será válido a partir de 24 horas e expirará uma semana depois.

```
$start = (Get-Date).AddHours(24)
$params = @{
  "ResourceUri"="https://cdn.example.com/index.html"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddDays(7)
  "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

Exemplo 3: cria uma URL assinada para o recurso especificado usando uma política personalizada. O URL será válido a partir de 24 horas e expirará uma semana depois. O acesso ao recurso é restrito ao intervalo de IP especificado.

```
$start = (Get-Date).AddHours(24)
$params = @{
  "ResourceUri"="https://cdn.example.com/index.html"
  "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
  "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
  "ExpiresOn"=(Get-Date).AddDays(7)
```

```
"ActiveFrom"=$start
"IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- Para obter detalhes da API, consulte [New-CF SignedUrl em AWS Tools for PowerShell Cmdlet Reference](#).

## CloudTrail exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CloudTrail.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Find-CTEvent**

O código de exemplo a seguir mostra como usar Find-CTEvent.

#### Ferramentas para PowerShell

Exemplo 1: retorna todos os eventos que ocorreram nos últimos sete dias. Por padrão, o cmdlet faz automaticamente várias chamadas para entregar todos os eventos, saindo quando o serviço indica que não há mais dados disponíveis.

```
Find-CTEvent
```

Exemplo 2: retorna todos os eventos que ocorreram nos últimos sete dias, especificando uma região que não é o padrão atual do shell.

```
Find-CTEvent -Region eu-central-1
```

Exemplo 3: retorna todos os eventos associados à chamada da RunInstances API.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";  
AttributeValue="RunInstances" }
```

Exemplo 4: retorna os primeiros 5 eventos disponíveis. O token a ser usado para recuperar outros eventos é anexado como uma propriedade de nota chamada 'NextToken' ao **\$AWSHistory.LastServiceResponse** membro.

```
Find-CTEvent -MaxResult 5
```

Exemplo 5: retorna os próximos 10 eventos usando o token “próxima página” de uma chamada anterior para indicar por onde começar a retornar eventos na sequência.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

Exemplo 6: Este exemplo mostra como percorrer os eventos disponíveis usando paginação manual, obtendo no máximo 5 eventos por chamada.

```
$nextToken = $null  
do  
{  
    Find-CTEvent -MaxResult 5 -NextToken $nextToken  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [LookupEvents](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CTTrail

O código de exemplo a seguir mostra como usar `Get-CTTrail`.

## Ferramentas para PowerShell

Exemplo 1: retorna as configurações de todas as trilhas associadas à região atual da sua conta.

```
Get-CTTrail
```

Exemplo 2: retorna as configurações das trilhas especificadas.

```
Get-CTTrail -TrailNameList trail1, trail2
```

Exemplo 3: Retorna as configurações das trilhas especificadas que foram criadas em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Get-CTTrail -TrailNameList trailABC, trailDEF -Region eu-central-1
```

- Para obter detalhes da API, consulte [DescribeTrails](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CTTrailStatus

O código de exemplo a seguir mostra como usar Get-CTTrailStatus.

## Ferramentas para PowerShell

Exemplo 1: Retorna informações de status da trilha com o nome 'myExampleTrail'. Os dados retornados incluem informações sobre erros de entrega, erros do Amazon SNS e do Amazon S3, além dos horários de início e término do registro da trilha. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Get-CTTrailStatus -Name myExampleTrail
```

Exemplo 2: Retorna informações de status de uma trilha que foi criada em uma região diferente do shell padrão atual (nesse caso, a região de Frankfurt (eu-central-1)).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte [GetTrailStatus](#) em Referência de AWS Tools for PowerShell cmdlet.



## New-CTTrail

O código de exemplo a seguir mostra como usar `New-CTTrail`.

### Ferramentas para PowerShell

Exemplo 1: cria uma trilha que usará o bucket 'mycloudtrailbucket' para armazenamento de arquivos de log.

```
New-CTTrail -Name="awscloudtrail-example" -S3BucketName="mycloudtrailbucket"
```

Exemplo 2: cria uma trilha que usará o bucket 'mycloudtrailbucket' para armazenamento de arquivos de log. Os objetos S3 que representam os registros terão um prefixo de chave comum de 'mylogs'. Quando novos registros forem entregues ao bucket, uma notificação será enviada para o tópico do SNS 'mlog-deliverytopic'. Este exemplo usa splatting para fornecer os valores dos parâmetros ao cmdlet.

```
$params = @{  
    Name="awscloudtrail-example"  
    S3BucketName="mycloudtrailbucket"  
    S3KeyPrefix="mylogs"  
    SnsTopicName="mlog-deliverytopic"  
}  
New-CTTrail @params
```

- Para obter detalhes da API, consulte [CreateTrail](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CTTrail

O código de exemplo a seguir mostra como usar `Remove-CTTrail`.

### Ferramentas para PowerShell

Exemplo 1: Exclui a trilha especificada. Você será solicitado a confirmar antes que o comando seja executado. Para suprimir a confirmação, adicione o parâmetro `-Force` switch.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Para obter detalhes da API, consulte [DeleteTrail](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-CTLogging

O código de exemplo a seguir mostra como usar `Start-CTLogging`.

### Ferramentas para PowerShell

Exemplo 1: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para a trilha chamada 'myExampleTrail'. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Start-CTLogging -Name myExampleTrail
```

Exemplo 2: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte [StartLogging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-CTLogging

O código de exemplo a seguir mostra como usar `Stop-CTLogging`.

### Ferramentas para PowerShell

Exemplo 1: suspende a gravação de chamadas de AWS API e a entrega do arquivo de log para a trilha chamada 'myExampleTrail'. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Stop-CTLogging -Name myExampleTrail
```

Exemplo 2: suspende a gravação de chamadas de AWS API e a entrega de arquivos de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte [StopLogging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CTTrail

O código de exemplo a seguir mostra como usar Update-CTTrail.

### Ferramentas para PowerShell

Exemplo 1: atualiza a trilha especificada para que os eventos globais do serviço (como os do IAM) sejam registrados e altera o prefixo de chave comum dos arquivos de log futuros para “globallogs”.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -S3KeyPrefix "globallogs"
```

Exemplo 2: atualiza a trilha especificada para que as notificações sobre novas entregas de registros sejam enviadas para o tópico do SNS especificado.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Exemplo 3: atualiza a trilha especificada para que os registros sejam entregues em um bucket diferente.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Para obter detalhes da API, consulte [UpdateTrail](#) em Referência de AWS Tools for PowerShell cmdlet.

## CloudWatch exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Get-CWDashboard

O código de exemplo a seguir mostra como usar Get-CWDashboard.

#### Ferramentas para PowerShell

Exemplo 1: retorna o ARN do corpo do painel especificado.

```
Get-CWDashboard -DashboardName Dashboard1
```

Saída:

```
DashboardArn                DashboardBody
-----
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Para obter detalhes da API, consulte [GetDashboard](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-CWDashboardList

O código de exemplo a seguir mostra como usar Get-CWDashboardList.

#### Ferramentas para PowerShell

Exemplo 1: retorna a coleção de painéis para sua conta.

```
Get-CWDashboardList
```

**Saída:**

```
DashboardArn DashboardName LastModified      Size
-----
arn:...      Dashboard1      7/6/2017 8:14:15 PM 252
```

Exemplo 2: retorna a coleção de painéis para sua conta cujos nomes começam com o prefixo “dev”.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Para obter detalhes da API, consulte [ListDashboards](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-CWDashboard**

O código de exemplo a seguir mostra como usar `Remove-CWDashboard`.

**Ferramentas para PowerShell**

Exemplo 1: exclui o painel especificado, solicitando uma confirmação antes de continuar. Para ignorar a confirmação, adicione a opção `-Force` para o comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte [DeleteDashboards](#) em Referência de AWS Tools for PowerShell cmdlet.

**Write-CWDashboard**

O código de exemplo a seguir mostra como usar `Write-CWDashboard`.

**Ferramentas para PowerShell**

Exemplo 1: cria ou atualiza o painel denominado “Dashboard1” para incluir dois widgets de métricas lado a lado.

```
$dashBody = @"
{
    "widgets": [
```

```
{
  "type":"metric",
  "x":0,
  "y":0,
  "width":12,
  "height":6,
  "properties":{
    "metrics":[
      [
        "AWS/EC2",
        "CPUUtilization",
        "InstanceId",
        "i-012345"
      ]
    ],
    "period":300,
    "stat":"Average",
    "region":"us-east-1",
    "title":"EC2 Instance CPU"
  }
},
{
  "type":"metric",
  "x":12,
  "y":0,
  "width":12,
  "height":6,
  "properties":{
    "metrics":[
      [
        "AWS/S3",
        "BucketSizeBytes",
        "BucketName",
        "MyBucketName"
      ]
    ],
    "period":86400,
    "stat":"Maximum",
    "region":"us-east-1",
    "title":"MyBucketName bytes"
  }
}
]
```

```
"@  
  
Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Exemplo 2: cria ou atualiza o painel, redirecionando o conteúdo que descreve o painel para o cmdlet.

```
$dashBody = @"  
{  
...  
}  
"@  
  
$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte [PutDashboard](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-CWMetricData

O código de exemplo a seguir mostra como usar Write-CWMetricData.

### Ferramentas para PowerShell

Exemplo 1: cria um novo MetricDatum objeto e o grava no Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object  
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum  
$Metric.Timestamp = [DateTime]::UtcNow  
$Metric.MetricName = 'CPU'  
$Metric.Value = 50  
  
### Write the metric data to the CloudWatch service  
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Para obter detalhes da API, consulte [PutMetricData](#) em Referência de AWS Tools for PowerShell cmdlet.

## CodeCommit exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CodeCommit.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### Get-CCBranch

O código de exemplo a seguir mostra como usar Get-CCBranch.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre a ramificação especificada para o repositório especificado.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Saída:

BranchName	CommitId
-----	-----
MyNewBranch	7763222d...561fc9c9

- Para obter detalhes da API, consulte [GetBranchem](#) Referência de AWS Tools for PowerShell cmdlet.



## Get-CCBranchList

O código de exemplo a seguir mostra como usar `Get-CCBranchList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista de nomes de ramificações para o repositório especificado.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Saída:

```
master
MyNewBranch
```

- Para obter detalhes da API, consulte [ListBranches](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CCRepository

O código de exemplo a seguir mostra como usar `Get-CCRepository`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações para o repositório especificado.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
AccountId           : 80398EXAMPLE
Arn                 : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp       : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh        : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate        : 9/8/2015 3:21:33 PM
DefaultBranch       :
LastModifiedDate    : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
```

```
RepositoryId      : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName    : MyDemoRepo
```

- Para obter detalhes da API, consulte [GetRepository](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CCRepositoryBatch

O código de exemplo a seguir mostra como usar `Get-CCRepositoryBatch`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo confirma quais dos repositórios especificados foram encontrados e não encontrados.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Saída:

```
Repositories                RepositoriesNotFound
-----
{MyDemoRepo, MyNewRepo}     {AMissingRepo}
```

- Para obter detalhes da API, consulte [BatchGetRepositories](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CCRepositoryList

O código de exemplo a seguir mostra como usar `Get-CCRepositoryList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todos os repositórios em ordem crescente por nome do repositório.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Saída:

```
RepositoryId                RepositoryName
-----
-----
```

```
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Para obter detalhes da API, consulte [ListRepositories](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CCBranch

O código de exemplo a seguir mostra como usar New-CCBranch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova ramificação com o nome especificado para o repositório especificado e o ID de confirmação especificado.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
7763222d...561fc9c9
```

- Para obter detalhes da API, consulte [CreateBranch](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CCRepository

O código de exemplo a seguir mostra como usar New-CCRepository.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo repositório com o nome e a descrição especificados.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
repository for demonstration purposes."
```

### Saída:

```
AccountId           : 80398EXAMPLE
Arn                 : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp       : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh        : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
```

```
CreationDate      : 9/18/2015 4:13:25 PM
DefaultBranch     :
LastModifiedDate  : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId      : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName    : MyDemoRepo
```

- Para obter detalhes da API, consulte [CreateRepository](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CCRepository

O código de exemplo a seguir mostra como usar `Remove-CCRepository`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui à força o repositório especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro `-Force` para excluir o repositório sem um aviso.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Para obter detalhes da API, consulte [DeleteRepository](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CCDefaultBranch

O código de exemplo a seguir mostra como usar `Update-CCDefaultBranch`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo altera a ramificação padrão do repositório especificado para a ramificação especificada.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Para obter detalhes da API, consulte [UpdateDefaultBranch](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CCRepositoryDescription

O código de exemplo a seguir mostra como usar Update-CCRepositoryDescription.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera a descrição do repositório especificado.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription  
"This is an updated description."
```

- Para obter detalhes da API, consulte [UpdateRepositoryDescription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CCRepositoryName

O código de exemplo a seguir mostra como usar Update-CCRepositoryName.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera o nome do repositório especificado.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Para obter detalhes da API, consulte [UpdateRepositoryName](#) em Referência de AWS Tools for PowerShell cmdlet.

## CodeDeploy exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CodeDeploy.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

### Add-CDOnPremiseInstanceTag

O código de exemplo a seguir mostra como usar Add-CDOnPremiseInstanceTag.

Ferramentas para PowerShell

Exemplo 1: este exemplo adiciona uma tag de instância local com a chave e o valor especificados para a instância local especificada.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";  
"Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte [AddTagsToOnPremisesInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-CDApplication

O código de exemplo a seguir mostra como usar Get-CDApplication.

Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre o aplicativo especificado.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Saída:

ApplicationId	ApplicationName	CreateTime
LinkedToGitHub		

```

-----
-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
9:49:48 PM    False

```

- Para obter detalhes da API, consulte [GetApplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDApplicationBatch

O código de exemplo a seguir mostra como usar Get-CDApplicationBatch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre os aplicativos especificados.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
CodePipelineDemoApplication
```

Saída:

```

ApplicationId                ApplicationName                CreateTime
-----
LinkedToGitHub
-----
-----
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication    8/13/2015
5:53:26 PM    False

```

- Para obter detalhes da API, consulte [BatchGetApplications](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDApplicationList

O código de exemplo a seguir mostra como usar Get-CDApplicationList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista dos aplicativos disponíveis.

```
Get-CDApplicationList
```

Saída:

```
CodeDeployDemoApplication  
CodePipelineDemoApplication
```

- Para obter detalhes da API, consulte [ListApplications](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDApplicationRevision

O código de exemplo a seguir mostra como usar Get-CDApplicationRevision.

Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre a revisão do aplicativo especificada.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication -  
S3Location_Bucket MyBucket -Revision_RevisionType S3 -S3Location_Key 5xd27EX.zip -  
S3Location_BundleType zip -S3Location_ETag 4565c1ac97187f190c1a90265EXAMPLE  
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",  
RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Saída:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,  
RegisterTime = 07/20/2015 23:46:42
```

- Para obter detalhes da API, consulte [GetApplicationRevision](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDApplicationRevisionList

O código de exemplo a seguir mostra como usar Get-CDApplicationRevisionList.

Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre as revisões disponíveis para o aplicativo especificado.



```

ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
  CodeDeployDemoApplication -Deployed Ignore)) {
>>  If ($revision.RevisionType -Eq "S3") {
>>    Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
  + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
  $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>  }
>>  If ($revision.RevisionType -Eq "GitHub") {
>>    Write-Output ("Type = GitHub, CommitId = " +
  $revision.GitHubLocation.CommitId + ", Repository = " +
  $revision.GitHubLocation.Repository)
>>  }
>> }
>> }
>> }

```

### Saída:

```

Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo

```

- Para obter detalhes da API, consulte [ListApplicationRevisions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeployment

O código de exemplo a seguir mostra como usar Get-CDDeployment.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações resumidas sobre a implantação especificada.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

### Saída:

```

ApplicationName      : CodeDeployDemoApplication
CompleteTime         : 7/23/2015 11:26:04 PM
CreateTime           : 7/23/2015 11:24:43 PM

```

```
Creator           : user
DeploymentConfigName : CodeDeployDefault.OneAtATime
DeploymentGroupName : CodeDeployDemoFleet
DeploymentId       : d-QZMRGSTEX
DeploymentOverview : Amazon.CodeDeploy.Model.DeploymentOverview
Description        :
ErrorInformation   :
IgnoreApplicationStopFailures : False
Revision           : Amazon.CodeDeploy.Model.RevisionLocation
StartTime          : 1/1/0001 12:00:00 AM
Status             : Succeeded
```

Exemplo 2: Este exemplo obtém informações sobre o status das instâncias que estão participando da implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Saída:

```
Failed       : 0
InProgress   : 0
Pending      : 0
Skipped      : 0
Succeeded    : 3
```

Exemplo 3: Este exemplo obtém informações sobre a revisão do aplicativo para a implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Saída:

```
Bucket       : MyBucket
BundleType    : zip
ETag          : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key           : clzfqEX
Version       :
```

- Para obter detalhes da API, consulte [GetDeployment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentBatch

O código de exemplo a seguir mostra como usar Get-CDDeploymentBatch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre as implantações especificadas.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

### Saída:

```
ApplicationName      : CodeDeployDemoApplication
CompleteTime        : 7/23/2015 11:26:04 PM
CreateTime          : 7/23/2015 11:24:43 PM
Creator             : user
DeploymentConfigName : CodeDeployDefault.OneAtATime
DeploymentGroupName  : CodeDeployDemoFleet
DeploymentId         : d-QZMRGSTEX
DeploymentOverview   : Amazon.CodeDeploy.Model.DeploymentOverview
Description          :
ErrorInformation     :
IgnoreApplicationStopFailures : False
Revision            : Amazon.CodeDeploy.Model.RevisionLocation
StartTime           : 1/1/0001 12:00:00 AM
Status              : Succeeded

ApplicationName      : CodePipelineDemoApplication
CompleteTime        : 7/23/2015 6:07:30 PM
CreateTime          : 7/23/2015 6:06:29 PM
Creator             : user
DeploymentConfigName : CodeDeployDefault.OneAtATime
DeploymentGroupName  : CodePipelineDemoFleet
DeploymentId         : d-RR0T5KTEX
DeploymentOverview   : Amazon.CodeDeploy.Model.DeploymentOverview
Description          :
ErrorInformation     :
IgnoreApplicationStopFailures : False
Revision            : Amazon.CodeDeploy.Model.RevisionLocation
StartTime           : 1/1/0001 12:00:00 AM
Status              : Succeeded
```

- Para obter detalhes da API, consulte [BatchGetDeployments](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentConfig

O código de exemplo a seguir mostra como usar Get-CDDeploymentConfig.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações resumidas sobre a configuração de implantação especificada.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Saída:

CreateTime	DeploymentConfigId	DeploymentConfigName
MinimumHealthyHosts		
-----	-----	-----
-----		
10/3/2014 4:32:30 PM	518a3950-d034-46a1-9d2c-3c949EXAMPLE	ThreeQuartersHealthy
Amazon.CodeDeploy.Model.MinimumHealthyHosts		

Exemplo 2: Este exemplo obtém informações sobre a definição da configuração de implantação especificada.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName  
ThreeQuartersHealthy).MinimumHealthyHosts)
```

Saída:

Type	Value
----	-----
FLEET_PERCENT	75

- Para obter detalhes da API, consulte [GetDeploymentConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentConfigList

O código de exemplo a seguir mostra como usar Get-CDDeploymentConfigList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista das configurações de implantação disponíveis.

```
Get-CDDeploymentConfigList
```

Saída:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Para obter detalhes da API, consulte [ListDeploymentConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentGroup

O código de exemplo a seguir mostra como usar Get-CDDeploymentGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre o grupo de implantação especificado.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
ApplicationName           : CodeDeployDemoApplication
AutoScalingGroups         : {}
DeploymentConfigName      : CodeDeployDefault.OneAtATime
DeploymentGroupId         : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName       : CodeDeployDemoFleet
Ec2TagFilters             : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn           : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-09MWP7XTL8EX
```

```
TargetRevision : Amazon.CodeDeploy.Model.RevisionLocation
```

- Para obter detalhes da API, consulte [GetDeploymentGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentGroupList

O código de exemplo a seguir mostra como usar `Get-CDDeploymentGroupList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista de grupos de implantação para o aplicativo especificado.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationName      DeploymentGroups
NextToken
-----
-----
CodeDeployDemoApplication {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Para obter detalhes da API, consulte [ListDeploymentGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentInstance

O código de exemplo a seguir mostra como usar `Get-CDDeploymentInstance`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre a instância especificada para a implantação especificada.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Saída:

```
DeploymentId      : d-QZMRGSTEX
InstanceId       : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt    : 7/23/2015 11:25:24 PM
LifecycleEvents  : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Para obter detalhes da API, consulte [GetDeploymentInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentInstanceList

O código de exemplo a seguir mostra como usar `Get-CDDeploymentInstanceList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo obtém uma lista de IDs de instância para a implantação especificada.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Saída:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Para obter detalhes da API, consulte [ListDeploymentInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CDDeploymentList

O código de exemplo a seguir mostra como usar `Get-CDDeploymentList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista de IDs de implantação para o aplicativo e o grupo de implantação especificados.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
CodeDeployDemoFleet
```

**Saída:**

```
d-QZMRGSTEX  
d-RR0T5KTEX
```

- Para obter detalhes da API, consulte [ListDeployments](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CDOnPremiseInstance**

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstance`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo obtém informações sobre a instância local especificada.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

**Saída:**

```
DeregisterTime : 1/1/0001 12:00:00 AM  
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser  
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/  
AssetTag12010298EX_rDH556dxEX  
InstanceName   : AssetTag12010298EX  
RegisterTime   : 4/3/2015 6:36:24 PM  
Tags           : {Name}
```

- Para obter detalhes da API, consulte [GetOnPremisesInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CDOnPremiseInstanceBatch**

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstanceBatch`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo obtém informações sobre as instâncias locais especificadas.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```



**Saída:**

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes da API, consulte [BatchGetOnPremisesInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CDOnPremiseInstanceList**

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstanceList`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo obtém uma lista de nomes de instâncias locais disponíveis.

```
Get-CDOnPremiseInstanceList
```

**Saída:**

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Para obter detalhes da API, consulte [ListOnPremisesInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CDApplication

O código de exemplo a seguir mostra como usar `New-CDApplication`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo aplicativo com o nome especificado.

```
New-CDApplication -ApplicationName MyNewApplication
```

Saída:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Para obter detalhes da API, consulte [CreateApplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CDDeployment

O código de exemplo a seguir mostra como usar `New-CDDeployment`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova implantação para o aplicativo e o grupo de implantação especificados com a configuração de implantação e a revisão do aplicativo especificadas.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket  
-S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime -  
DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -  
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

Saída:

```
d-ZHR0G7UEX
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelas quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma implantação azul/verde.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket
-S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
-S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Saída:

```
d-ZHROG7UEX
```

- Para obter detalhes da API, consulte [CreateDeployment](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CDDeploymentConfig

O código de exemplo a seguir mostra como usar `New-CDDeploymentConfig`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova configuração de implantação com o nome e o comportamento especificados.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Saída:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Para obter detalhes da API, consulte [CreateDeploymentConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CDDeploymentGroup

O código de exemplo a seguir mostra como usar `New-CDDeploymentGroup`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um grupo de implantação com o nome especificado, grupo Auto Scaling, configuração de implantação, tag e função de serviço para o aplicativo especificado.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelas quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma implantação azul/verde.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key="
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Para obter detalhes da API, consulte [CreateDeploymentGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-CDApplicationRevision

O código de exemplo a seguir mostra como usar Register-CDApplicationRevision.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo registra uma revisão do aplicativo com o local especificado do Amazon S3, para o aplicativo especificado.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
MyBucket -S3Location_BundleType zip -S3Location_Key aws-codedeploy_linux-master.zip
-Revision_RevisionType S3
```

- Para obter detalhes da API, consulte [RegisterApplicationRevision](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-CDOnPremiseInstance

O código de exemplo a seguir mostra como usar Register-CDOnPremiseInstance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo registra uma instância local com o nome e o usuário do IAM especificados.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/  
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte [RegisterOnPremisesInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CDApplication

O código de exemplo a seguir mostra como usar Remove-CDApplication.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o aplicativo com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o aplicativo sem um aviso.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Para obter detalhes da API, consulte [DeleteApplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CDDeploymentConfig

O código de exemplo a seguir mostra como usar Remove-CDDeploymentConfig.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a configuração de implantação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro `-Force` para excluir a configuração de implantação sem um aviso.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Para obter detalhes da API, consulte [DeleteDeploymentConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CDDeploymentGroup

O código de exemplo a seguir mostra como usar `Remove-CDDeploymentGroup`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o grupo de implantação com o nome especificado para o aplicativo especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro `-Force` para excluir o grupo de implantação sem um aviso.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName  
MyNewDeploymentGroup
```

- Para obter detalhes da API, consulte [DeleteDeploymentGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CDOnPremiseInstanceTag

O código de exemplo a seguir mostra como usar `Remove-CDOnPremiseInstanceTag`.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo exclui a tag especificada para a instância local com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro `-Force` para excluir a tag sem um aviso.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =  
"Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte [RemoveTagsFromOnPremisesInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-CDDeployment

O código de exemplo a seguir mostra como usar Stop-CDDeployment.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo tenta interromper a implantação com o ID de implantação especificado.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Saída:

```
Status      StatusMessage
-----      -
Pending     Stopping Pending. Stopping to schedule commands in the deployment
instances
```

- Para obter detalhes da API, consulte [StopDeployment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-CDOnPremiseInstance

O código de exemplo a seguir mostra como usar Unregister-CDOnPremiseInstance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cancela o registro da instância local com o nome especificado.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte [DeregisterOnPremisesInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CDApplication

O código de exemplo a seguir mostra como usar Update-CDApplication.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera o nome do aplicativo especificado.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
MyNewApplication-2
```

- Para obter detalhes da API, consulte [UpdateApplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CDDeploymentGroup

O código de exemplo a seguir mostra como usar Update-CDDeploymentGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo altera o nome do grupo de implantação especificado para o aplicativo especificado.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
MyNewDeploymentGroup-2
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelas quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma implantação azul/verde.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
MyNewDeploymentGroup-2 -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Para obter detalhes da API, consulte [UpdateDeploymentGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## CodePipeline exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with CodePipeline.



Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### **Confirm-CPJob**

O código de exemplo a seguir mostra como usar `Confirm-CPJob`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém o status do trabalho especificado.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Saída:

```
Value  
-----  
InProgress
```

- Para obter detalhes da API, consulte [AcknowledgeJob](#) em Referência de AWS Tools for PowerShell cmdlet.

### **Disable-CPStageTransition**

O código de exemplo a seguir mostra como usar `Disable-CPStageTransition`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling temporarily." -StageName Beta -TransitionType Inbound
```

- Para obter detalhes da API, consulte [DisableStageTransition](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-CPStageTransition

O código de exemplo a seguir mostra como usar `Enable-CPStageTransition`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita a transição de entrada para o estágio especificado no pipeline especificado.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -TransitionType Inbound
```

- Para obter detalhes da API, consulte [EnableStageTransition](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPActionType

O código de exemplo a seguir mostra como usar `Get-CPActionType`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre todas as ações disponíveis para o proprietário especificado.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {  
    Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +  
$actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version = " +  
$actionType.Id.Version + ":")  
    Write-Output (" ActionConfigurationProperties:")
```

```

ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("        Description = " + $acp.Description)
    Write-Output ("        Key = " + $acp.Key)
    Write-Output ("        Queryable = " + $acp.Queryable)
    Write-Output ("        Required = " + $acp.Required)
    Write-Output ("        Secret = " + $acp.Secret)
}
Write-Output ("  InputArtifactDetails:")
Write-Output ("    MaximumCount = " +
$actionType.InputArtifactDetails.MaximumCount)
Write-Output ("    MinimumCount = " +
$actionType.InputArtifactDetails.MinimumCount)
Write-Output ("  OutputArtifactDetails:")
Write-Output ("    MaximumCount = " +
$actionType.OutputArtifactDetails.MaximumCount)
Write-Output ("    MinimumCount = " +
$actionType.OutputArtifactDetails.MinimumCount)
Write-Output ("  Settings:")
Write-Output ("    EntityUrlTemplate = " + $actionType.Settings.EntityUrlTemplate)
Write-Output ("    ExecutionUrlTemplate = " +
$actionType.Settings.ExecutionUrlTemplate)
}

```

### Saída:

```

For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
ActionConfigurationProperties:
  For ApplicationName:
    Description = The AWS Elastic Beanstalk Application name
    Key = True
    Queryable = False
    Required = True
    Secret = False
  For EnvironmentName:
    Description = The AWS Elastic Beanstalk Environment name
    Key = True
    Queryable = False
    Required = True
    Secret = False
InputArtifactDetails:
  MaximumCount = 1
  MinimumCount = 1

```

```

OutputArtifactDetails:
  MaximumCount = 0
  MinimumCount = 0
Settings:
  EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
  ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
ActionConfigurationProperties:
  For ApplicationName:
    Description = The AWS CodeDeploy Application name
    Key = True
    Queryable = False
    Required = True
    Secret = False
  For DeploymentGroupName:
    Description = The AWS CodeDeploy Deployment Group name
    Key = True
    Queryable = False
    Required = True
    Secret = False
InputArtifactDetails:
  MaximumCount = 1
  MinimumCount = 1
OutputArtifactDetails:
  MaximumCount = 0
  MinimumCount = 0
Settings:
  EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
applications/{Config:ApplicationName}/deployment-groups/{Config:DeploymentGroupName}
  ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}

```

- Para obter detalhes da API, consulte [ListActionTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPActionableJobList

O código de exemplo a seguir mostra como usar Get-CPActionableJobList.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações sobre todos os trabalhos acionáveis para a categoria de ação, proprietário, provedor, versão e parâmetros de consulta especificados.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
-ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
@{"ProjectName" = "MyProjectName"}
```

Saída:

AccountId	Data	Id
----- -----	----	--
80398EXAMPLE f57a0EXAMPLE	Amazon.CodePipeline.Model.JobData 3	0de392f5-712d-4f41-ace3-

- Para obter detalhes da API, consulte [PollForJobs](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPJobDetail

O código de exemplo a seguir mostra como usar `Get-CPJobDetail`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações gerais sobre o trabalho especificado.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Saída:

AccountId	Data	Id
-----	----	--
80398EXAMPLE	Amazon.CodePipeline.Model.JobData	f570dc12-5ef3-44bc-945a-6e133EXAMPLE

Exemplo 2: Este exemplo obtém informações detalhadas sobre o trabalho especificado.

```

$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
    $value = $jobData.ActionConfiguration.$key
    Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
    $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
    Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
    Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)

```

### Saída:

```

For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1

```

```
ArtifactCredentials:
  AccessKeyId = ASIAIEI3...IXI6YREX
  SecretAccessKey = cqAFDhEi...RdQyfa2u
  SessionToken = AQoDYXdz...5u+lsAU=
InputArtifacts:
  MyApp
OutputArtifacts:
  MyAppBuild
PipelineContext:
  Name = Build
  PipelineName = CodePipelineDemo
  Stage = Build
```

- Para obter detalhes da API, consulte [GetJobDetails](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPPipeline

O código de exemplo a seguir mostra como usar Get-CPPipeline.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações gerais sobre o pipeline especificado.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Exemplo 2: Este exemplo obtém informações detalhadas sobre o pipeline especificado.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
```

```

Write-Output (" Location = " + $pipeline.ArtifactStore.Location)
Write-Output (" Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
    Write-Output (" Name = " + $stage.Name)
    Write-Output (" Actions:")
    ForEach ($action in $stage.Actions) {
        Write-Output (" Name = " + $action.Name)
Write-Output (" Category = " + $action.ActionTypeId.Category)
Write-Output (" Owner = " + $action.ActionTypeId.Owner)
Write-Output (" Provider = " + $action.ActionTypeId.Provider)
Write-Output (" Version = " + $action.ActionTypeId.Version)
Write-Output (" Configuration:")
ForEach ($key in $action.Configuration.Keys) {
    $value = $action.Configuration.$key
    Write-Output (" " + $key + " = " + $value)
}
Write-Output (" InputArtifacts:")
ForEach ($ia in $action.InputArtifacts) {
    Write-Output (" " + $ia.Name)
}
ForEach ($oa in $action.OutputArtifacts) {
    Write-Output (" " + $oa.Name)
}
Write-Output (" RunOrder = " + $action.RunOrder)
}
}

```

### Saída:

```

Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
    Location = MyBucketName
    Type = S3
Stages:
    Name = Source
    Actions:
        Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub

```



```
    Version = 1
    Configuration:
      Branch = master
      OAuthToken = ****
      Owner = my-user-name
      Repo = MyRepoName
    InputArtifacts:
      MyApp
    RunOrder = 1
Name = Build
  Actions:
    Name = Build
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
    Configuration:
      ProjectName = MyProjectName
    InputArtifacts:
      MyApp
      MyAppBuild
    RunOrder = 1
Name = Beta
  Actions:
    Name = CodePipelineDemoFleet
    Category = Deploy
    Owner = AWS
    Provider = CodeDeploy
    Version = 1
    Configuration:
      ApplicationName = CodePipelineDemoApplication
      DeploymentGroupName = CodePipelineDemoFleet
    InputArtifacts:
      MyAppBuild
    RunOrder = 1
Name = TestStage
  Actions:
    Name = MyJenkinsTestAction
    Category = Test
    Owner = Custom
    Provider = MyCustomTestProvider
    Version = 1
    Configuration:
      ProjectName = MyJenkinsProjectName
```

```
InputArtifacts:
  MyAppBuild
  RunOrder = 1
```

- Para obter detalhes da API, consulte [GetPipeline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPPipelineList

O código de exemplo a seguir mostra como usar Get-CPPipelineList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém uma lista dos pipelines disponíveis.

```
Get-CPPipelineList
```

Saída:

Created	Name	Updated	Version
8/13/2015 10:17:54 PM	CodePipelineDemo	8/13/2015 10:17:54 PM	3
7/8/2015 2:41:53 AM	MyFirstPipeline	7/22/2015 9:06:37 PM	7

- Para obter detalhes da API, consulte [ListPipelines](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CPPipelineState

O código de exemplo a seguir mostra como usar Get-CPPipelineState.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém informações gerais sobre os estágios do pipeline especificado.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Saída:

```
Created          : 8/13/2015 10:17:54 PM
```

```
PipelineName      : CodePipelineDemo
PipelineVersion   : 1
StageStates       : {Source, Build, Beta, TestStage}
Updated           : 8/13/2015 10:17:54 PM
```

Exemplo 2: Este exemplo obtém informações detalhadas sobre o estado do pipeline especificado.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
    Write-Output ("For " + $stageState.StageName + ":")
    Write-Output ("  InboundTransitionState:")
    Write-Output ("    DisabledReason = " +
$stageState.InboundTransitionState.DisabledReason)
    Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
    Write-Output ("    LastChangedAt = " +
$stageState.InboundTransitionState.LastChangedAt)
    Write-Output ("    LastChangedBy = " +
$stageState.InboundTransitionState.LastChangedBy)
    Write-Output ("  ActionStates:")
    ForEach ($actionState in $stageState.ActionStates) {
        Write-Output ("    For " + $actionState.ActionName + ":")
    Write-Output ("      CurrentRevision:")
        Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
    Write-Output ("        RevisionChangeId = " +
$actionState.CurrentRevision.RevisionChangeId)
    Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
    Write-Output ("        EntityUrl = " + $actionState.EntityUrl)
    Write-Output ("      LatestExecution:")
        Write-Output ("        ErrorDetails:")
        Write-Output ("          Code = " +
$actionState.LatestExecution.ErrorDetails.Code)
    Write-Output ("          Message = " +
$actionState.LatestExecution.ErrorDetails.Message)
    Write-Output ("        ExternalExecutionId = " +
$actionState.LatestExecution.ExternalExecutionId)
    Write-Output ("        ExternalExecutionUrl = " +
$actionState.LatestExecution.ExternalExecutionUrl)
    Write-Output ("        LastStatusChange = " +
$actionState.LatestExecution.LastStatusChange)
    Write-Output ("        PercentComplete = " +
$actionState.LatestExecution.PercentComplete)
    Write-Output ("        Status = " + $actionState.LatestExecution.Status)
    Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
    Write-Output ("        RevisionUrl = " + $actionState.RevisionUrl)
```

```
}  
}
```

**Saída:**

```
For Source:  
  InboundTransitionState:  
    DisabledReason =  
    Enabled =  
    LastChangedAt =  
    LastChangedBy =  
  ActionStates:  
    For Source:  
      CurrentRevision:  
        Created =  
        RevisionChangeId =  
        RevisionId =  
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master  
      LatestExecution:  
        ErrorDetails:  
          Code =  
          Message =  
        ExternalExecutionId =  
        ExternalExecutionUrl =  
        LastStatusChange = 07/20/2015 23:28:45  
        PercentComplete = 0  
        Status = Succeeded  
        Summary =  
      RevisionUrl =  
For Build:  
  InboundTransitionState:  
    DisabledReason =  
    Enabled = True  
    LastChangedAt = 01/01/0001 00:00:00  
    LastChangedBy =  
  ActionStates:  
    For Build:  
      CurrentRevision:  
        Created =  
        RevisionChangeId =  
        RevisionId =  
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo  
      LatestExecution:
```

```
ErrorDetails:
  Code = TimeoutError
  Message = The action failed because a job worker exceeded its time limit.
If this is a custom action, make sure that the job worker is configured correctly.
  ExternalExecutionId =
  ExternalExecutionUrl =
  LastStatusChange = 07/21/2015 00:29:29
  PercentComplete = 0
  Status = Failed
  Summary =
  RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
        EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
          ExternalExecutionId = d-D5LTCZXEX
          ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
          LastStatusChange = 07/08/2015 22:07:42
          PercentComplete = 0
          Status = Succeeded
          Summary = Deployment Succeeded
          RevisionUrl =
    For TestStage:
      InboundTransitionState:
        DisabledReason =
        Enabled = True
        LastChangedAt = 01/01/0001 00:00:00
        LastChangedBy =
      ActionStates:
```

```

For MyJenkinsTestAction25:
  CurrentRevision:
    Created =
    RevisionChangeId =
    RevisionId =
  EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
  LatestExecution:
    ErrorDetails:
      Code =
      Message =
    ExternalExecutionId = 5
    ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
    LastStatusChange = 07/08/2015 22:09:03
    PercentComplete = 0
    Status = Succeeded
    Summary = Finished
  RevisionUrl =

```

- Para obter detalhes da API, consulte [GetPipelineState](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CPCustomActionType

O código de exemplo a seguir mostra como usar New-CPCustomActionType.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma nova ação personalizada com as propriedades especificadas.

```

New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
= "The name of the build project must be provided when this action is added
to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
= $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
-Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1

```

Saída:

```
ActionConfigurationProperties : {ProjectName}
```

```
Id : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails : Amazon.CodePipeline.Model.ArtifactDetails
Settings : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Para obter detalhes da API, consulte [CreateCustomActionType](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CPPipeline

O código de exemplo a seguir mostra como usar New-CPPipeline.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um novo pipeline com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
    Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
    "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "MyBucketName")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
    "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
    "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"
```

```
$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "MyBucketName"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

#### Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Para obter detalhes da API, consulte [CreatePipeline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CPCustomActionType

O código de exemplo a seguir mostra como usar Remove-CPCustomActionType.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a ação personalizada especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a ação personalizada sem um aviso.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```



- Para obter detalhes da API, consulte [DeleteCustomActionType](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CPPipeline

O código de exemplo a seguir mostra como usar Remove-CPPipeline.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o pipeline especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o pipeline sem um aviso.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte [DeletePipeline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-CPPipelineExecution

O código de exemplo a seguir mostra como usar Start-CPPipelineExecution.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo começa a executar o pipeline especificado.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte [StartPipelineExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CPPipeline

O código de exemplo a seguir mostra como usar Update-CPPipeline.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza o pipeline existente especificado com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration
```

```
$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
    Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
    "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "MyBucketName")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
    "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
    "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "MyBucketName"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

## Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Para obter detalhes da API, consulte [UpdatePipeline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de identidade do Amazon Cognito usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell Amazon Cognito Identity.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-CGIIIdentityPool**

O código de exemplo a seguir mostra como usar `Get-CGIIIdentityPool`.

### Ferramentas para PowerShell

Exemplo 1: recupera informações sobre um grupo de identidades específico por meio de seu id.

```
Get-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

### Saída:

```

LoggedAt           : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName :
IdentityPoolId     : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName   : CommonTests1
OpenIdConnectProviderARNs : {}
SupportedLoginProviders : {}
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 142
HttpStatusCode     : OK

```

- Para obter detalhes da API, consulte [DescribeIdentityPool](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CGIIIdentityPoolList

O código de exemplo a seguir mostra como usar `Get-CGIIIdentityPoolList`.

### Ferramentas para PowerShell

Exemplo 1: Recupera uma lista de grupos de identidades existentes.

```
Get-CGIIIdentityPoolList
```

### Saída:

```

IdentityPoolId           IdentityPoolName
-----
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1   CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2   Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3   CommonTests13

```

- Para obter detalhes da API, consulte [ListIdentityPools](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CGIIIdentityPoolRole

O código de exemplo a seguir mostra como usar `Get-CGIIIdentityPoolRole`.

### Ferramentas para PowerShell

Exemplo 1: Obtém as informações sobre as funções de um grupo de identidades específico.

```
Get-CGIIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Saída:

```
LoggedAt           : 8/12/2015 4:33:51 PM
IdentityPoolId     : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles              : {[unauthenticated, arn:aws:iam::123456789012:role/CommonTests1Role]}
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 165
HttpStatusCode     : OK
```

- Para obter detalhes da API, consulte [GetIdentityPoolRoles](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-CGIIIdentityPool

O código de exemplo a seguir mostra como usar `New-CGIIIdentityPool`.

### Ferramentas para PowerShell

Exemplo 1: Cria um novo grupo de identidades que permite identidades não autenticadas.

```
New-CGIIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName CommonTests13
```

Saída:

```
LoggedAt           : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
```

```
IdentityPoolId           : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName        : CommonTests13
OpenIdConnectProviderARNs : {}
SupportedLoginProviders  : {}
ResponseMetadata        : Amazon.Runtime.ResponseMetadata
ContentLength           : 136
HttpStatusCode           : OK
```

- Para obter detalhes da API, consulte [CreateIdentityPool](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-CGIIdentityPool

O código de exemplo a seguir mostra como usar `Remove-CGIIdentityPool`.

### Ferramentas para PowerShell

Exemplo 1: Exclui um grupo de identidades específico.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Para obter detalhes da API, consulte [DeleteIdentityPool](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-CGIIdentityPoolRole

O código de exemplo a seguir mostra como usar `Set-CGIIdentityPoolRole`.

### Ferramentas para PowerShell

Exemplo 1: configura o grupo de identidades específico para ter uma função do IAM não autenticada.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Para obter detalhes da API, consulte [SetIdentityPoolRoles](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-CGIIIdentityPool

O código de exemplo a seguir mostra como usar Update-CGIIIdentityPool.

### Ferramentas para PowerShell

Exemplo 1: Atualiza algumas das propriedades do Identity Pool, neste caso, o nome do Identity Pool.

```
Update-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

### Saída:

```
LoggedAt                : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName   :
IdentityPoolId          : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName        : NewPoolName
OpenIdConnectProviderARNs : {}
SupportedLoginProviders : {}
ResponseMetadata        : Amazon.Runtime.ResponseMetadata
ContentLength           : 135
HttpStatusCode          : OK
```

- Para obter detalhes da API, consulte [UpdateIdentityPool](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Config exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Config.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-CFGResourceTag

O código de exemplo a seguir mostra como usar Add-CFGResourceTag.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo associa a tag especificada ao ARN do recurso, que neste caso é config-rule/config-rule-16iyn0.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- Para obter detalhes da API, consulte [TagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-CFGAggregateComplianceByConfigRuleList

O código de exemplo a seguir mostra como usar Get-CFGAggregateComplianceByConfigRuleList.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca os detalhes da filtragem ConfigurationAggregator 'kaju' para a regra de configuração fornecida e expande/retorna a 'Conformidade' da regra.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
-Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Saída:

```
ComplianceContributorCount ComplianceType
```



```
-----
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Exemplo 2: Este exemplo busca detalhes do dado ConfigurationAggregator, filtra para a conta específica para todas as regiões cobertas pelo agregador e retorna ainda mais a conformidade de todas as regras.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
@{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Saída:

ConfigRuleName	Compliance
-----	-----
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK	NON_COMPLIANT
ec2-instance-no-public-ip	NON_COMPLIANT
desired-instance-type	NON_COMPLIANT

- Para obter detalhes da API, consulte [DescribeAggregateComplianceByConfigRules](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGAggregateComplianceDetailsByConfigRule

O código de exemplo a seguir mostra como usar Get-CFGAggregateComplianceDetailsByConfigRule.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna os resultados da avaliação selecionando a saída com resource-id e resource-type para a regra de AWS configuração " que estão no estado 'COMPATÍVELdesired-instance-type' para a conta, agregador, região e regra de configuração fornecidos

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
EvaluationResultQualifier
```

**Saída:**

```

ConfigRuleName      ResourceId      ResourceType
-----
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance

```

- Para obter detalhes da API, consulte [GetAggregateComplianceDetailsByConfigRule](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CFGAggregateConfigRuleComplianceSummary**

O código de exemplo a seguir mostra como usar `Get-CFGAggregateConfigRuleComplianceSummary`.

**Ferramentas para PowerShell**

Exemplo 1: Esse exemplo retorna o número de regras não compatíveis para o agregador especificado.

```

(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
  raj).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount

```

**Saída:**

```

CapExceeded CappedCount
-----
False      5

```

- Para obter detalhes da API, consulte [GetAggregateConfigRuleComplianceSummary](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CFGAggregateDiscoveredResourceCount**

O código de exemplo a seguir mostra como usar `Get-CFGAggregateDiscoveredResourceCount`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna a contagem de recursos para o agregador especificado filtrado para a região us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Saída:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
-----
{} 455
```

Exemplo 2: Esse exemplo retorna a contagem de recursos agrupada por RESOURCE\_TYPE para a região filtrada de um determinado agregador.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
Select-Object -ExpandProperty GroupedResourceCounts
```

Saída:

GroupName	ResourceCount
-----	-----
AWS::CloudFormation::Stack	12
AWS::CloudFront::Distribution	1
AWS::CloudTrail::Trail	1
AWS::DynamoDB::Table	1
AWS::EC2::EIP	2
AWS::EC2::FlowLog	2
AWS::EC2::InternetGateway	4
AWS::EC2::NatGateway	2
AWS::EC2::NetworkAcl	4
AWS::EC2::NetworkInterface	12
AWS::EC2::RouteTable	13
AWS::EC2::SecurityGroup	18
AWS::EC2::Subnet	16
AWS::EC2::VPC	4
AWS::EC2::VPCEndpoint	2
AWS::EC2::VPCPeeringConnection	1

```

AWS::IAM::Group                2
AWS::IAM::Policy               51
AWS::IAM::Role                 78
AWS::IAM::User                 7
AWS::Lambda::Function          3
AWS::RDS::DBSecurityGroup      1
AWS::S3::Bucket                3
AWS::SSM::AssociationCompliance 107
AWS::SSM::ManagedInstanceInventory 108

```

- Para obter detalhes da API, consulte [GetAggregateDiscoveredResourceCounts](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGAggregateDiscoveredResourceList

O código de exemplo a seguir mostra como usar `Get-CFGAggregateDiscoveredResourceList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna os identificadores de recursos para o determinado tipo de recurso agregado no agregador "Irlanda". Para ver a lista de tipos de recursos, consulte [https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TCFGServiceResourceTypeConfigService.html&tocid=AMAZON\\_.\\_ResourceType](https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TCFGServiceResourceTypeConfigService.html&tocid=AMAZON_._ResourceType)

```

Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)

```

### Saída:

```

ResourceId      : arn:aws:autoscaling:eu-
west-1:123456789012:autoScalingGroup:12e3b4fc-1234-1234-
a123-1d2ba3c45678:autoScalingGroupName/asg-1
ResourceName    : asg-1
ResourceType    : AWS::AutoScaling::AutoScalingGroup
SourceAccountId : 123456789012
SourceRegion    : eu-west-1

```

Exemplo 2: Esse exemplo retorna o tipo de recurso **AwsEC2SecurityGroup** chamado 'default' para o agregador especificado filtrado com a região us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

### Saída:

```
ResourceId      : sg-01234bd5dbfa67c89
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-0123a4ebbf56789be
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-4fc1d234
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1
```

- Para obter detalhes da API, consulte [ListAggregateDiscoveredResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGAggregateResourceConfig

O código de exemplo a seguir mostra como usar `Get-CFGAggregateResourceConfig`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna o item de configuração para o determinado recurso agregado e expande a configuração.

```
(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json
```

**Saída:**

```
{
  "description": "default VPC security group",
  "groupName": "default",
  "ipPermissions": [
    {
      "ipProtocol": "-1",
      "ipv6Ranges": [],
      "prefixListIds": [],
      "userIdGroupPairs": [
        {
          "groupId": "sg-4fc1d234",
          "userId": "123456789012"
        }
      ],
      "ipv4Ranges": [
        {
          "fromPort": 3389,
          "ipProtocol": "tcp",
          "ipv6Ranges": [],
          "prefixListIds": [],
          "toPort": 3389,
          "userIdGroupPairs": []
        }
      ],
      "ipv4Ranges": [
        {
          "cidrIp": "54.240.197.224/29",
          "description": "office subnet"
        },
        {
          "cidrIp": "72.21.198.65/32",
          "description": "home pc"
        }
      ],
      "ipRanges": [
        "54.240.197.224/29",
        "72.21.198.65/32"
      ],
      "ownerId": "123456789012",
      "groupId": "sg-4fc1d234",
      "ipPermissions": [
        {
          "ipProtocol": "-1",
          "ipv6Ranges": [],
          "prefixListIds": [],
          "userIdGroupPairs": []
        }
      ],
      "ipv4Ranges": [
        {
          "cidrIp": "0.0.0.0/0"
        }
      ],
      "ipRanges": [
        "0.0.0.0/0"
      ],
      "tags": [],
      "vpcId": "vpc-2d1c2e34"
    }
  ]
}
```

- Para obter detalhes da API, consulte [GetAggregateResourceconfig-service](#) no AWS Tools for PowerShell Cmdlet Reference.

**Get-CFGAggregateResourceConfigBatch**

O código de exemplo a seguir mostra como usar Get-CFGAggregateResourceConfigBatch.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo busca o item de configuração atual do recurso (identificado) presente no agregador fornecido.

```
$resIdentifier=[Amazon.ConfigService.Model.AggregateResourceIdentifier]@{
  ResourceId= "i-012e3cb4df567e8aa"
  ResourceName = "arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa"
  ResourceType = [Amazon.ConfigService.ResourceType]::AWSEC2Instance
  SourceAccountId = "123456789012"
  SourceRegion = "eu-west-1"
}

Get-CFGAggregateResourceConfigBatch -ResourceIdentifier $resIdentifier -
ConfigurationAggregatorName raju
```

**Saída:**

```
BaseConfigurationItems UnprocessedResourceIdentifiers
-----
```

```
{} {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Para obter detalhes da API, consulte [BatchGetAggregateResourceconfig-service](#) no AWS Tools for PowerShell Cmdlet Reference.

## Get-CFGAggregationAuthorizationList

O código de exemplo a seguir mostra como usar `Get-CFGAggregationAuthorizationList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera as autorizações concedidas aos agregadores.

```
Get-CFGAggregationAuthorizationList
```

Saída:

```
AggregationAuthorizationArn
  AuthorizedAccountId AuthorizedAwsRegion CreationTime
-----
-----
arn:aws:config-service:eu-west-1:123456789012:aggregation-
authorization/123456789012/eu-west-1 123456789012 eu-west-1
8/26/2019 12:55:27 AM
```

- Para obter detalhes da API, consulte [DescribeAggregationAuthorizations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGComplianceByConfigRule

O código de exemplo a seguir mostra como usar `Get-CFGComplianceByConfigRule`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera detalhes de conformidade da regra `ebs-optimized-instance`, para os quais não há resultados de avaliação atuais para a regra, portanto, retorna `INSUFFICIENT_DATA`

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

**Saída:**

```
ComplianceContributorCount ComplianceType
-----
INSUFFICIENT_DATA
```

Exemplo 2: Esse exemplo retorna o número de recursos não compatíveis para a regra ALB\_HTTP\_TO\_HTTPS\_REDIRECTION\_CHECK.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

**Saída:**

```
CapExceeded CappedCount
-----
False      2
```

- Para obter detalhes da API, consulte [DescribeComplianceByConfigRule](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CFGComplianceByResource**

O código de exemplo a seguir mostra como usar `Get-CFGComplianceByResource`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo verifica o tipo **AWS::SSM::ManagedInstanceInventory** de recurso quanto ao tipo de conformidade “COMPATÍVEL”.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
AWS::SSM::ManagedInstanceInventory
```

**Saída:**

```
Compliance                               ResourceId           ResourceType
-----
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
AWS::SSM::ManagedInstanceInventory
```



```
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
AWS::SSM::ManagedInstanceInventory
```

- Para obter detalhes da API, consulte [DescribeComplianceByResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGComplianceDetailsByConfigRule

O código de exemplo a seguir mostra como usar `Get-CFGComplianceDetailsByConfigRule`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém os resultados da avaliação da regra `access-keys-rotated` e retorna a saída agrupada por tipo de conformidade

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-Object ComplianceType
```

Saída:

```
Count Name                               Group
-----
      2 COMPLIANT                        {Amazon.ConfigService.Model.EvaluationResult,
Amazon.ConfigService.Model.EvaluationResult}
      5 NON_COMPLIANT                    {Amazon.ConfigService.Model.EvaluationResult,
Amazon.ConfigService.Model.EvaluationResult,
Amazon.ConfigService.Model.EvaluationRes...
```

Exemplo 2: Este exemplo consulta os detalhes de conformidade da regra `access-keys-rotated` para recursos COMPATÍVEIS.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
{$_ .EvaluationResultIdentifier.EvaluationResultQualifier}
```

Saída:

```
ConfigRuleName      ResourceId          ResourceType
-----
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
```

```
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Para obter detalhes da API, consulte [GetComplianceDetailsByConfigRule](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGComplianceDetailsByResource

O código de exemplo a seguir mostra como usar `Get-CFGComplianceDetailsByResource`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo de resultados de avaliação para o recurso fornecido.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType  
'AWS::IAM::User'
```

#### Saída:

```
Annotation           :  
ComplianceType       : COMPLIANT  
ConfigRuleInvokedTime : 8/25/2019 11:34:56 PM  
EvaluationResultIdentifier : Amazon.ConfigService.Model.EvaluationResultIdentifier  
ResultRecordedTime   : 8/25/2019 11:34:56 PM  
ResultToken          :
```

- Para obter detalhes da API, consulte [GetComplianceDetailsByResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGComplianceSummaryByConfigRule

O código de exemplo a seguir mostra como usar `Get-CFGComplianceSummaryByConfigRule`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna o número de regras do Config que não estão em conformidade.

```
Get-CFGComplianceSummaryByConfigRule -Select  
ComplianceSummary.NonCompliantResourceCount
```

#### Saída:

```
CapExceeded CappedCount
-----
False      9
```

- Para obter detalhes da API, consulte [GetComplianceSummaryByConfigRule](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGComplianceSummaryByResourceType

O código de exemplo a seguir mostra como usar `Get-CFGComplianceSummaryByResourceType`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna o número de recursos compatíveis ou não compatíveis e converte a saída em json.

```
Get-CFGComplianceSummaryByResourceType -Select
ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Para obter detalhes da API, consulte [GetComplianceSummaryByResourceType](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigRule

O código de exemplo a seguir mostra como usar `Get-CFGConfigRule`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as regras de configuração da conta, com propriedades selecionadas.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
ConfigRuleState
```

Saída:

```
ConfigRuleName                ConfigRuleId                ConfigRuleArn
-----
ALB_REDIRECTION_CHECK         config-rule-12iyn3         arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3 ACTIVE
access-keys-rotated          config-rule-aospfr         arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr ACTIVE
autoscaling-group-elb-healthcheck-required config-rule-cn1f2x         arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x ACTIVE
```

- Para obter detalhes da API, consulte [DescribeConfigRules](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigRuleEvaluationStatus

O código de exemplo a seguir mostra como usar `Get-CFGConfigRuleEvaluationStatus`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna as informações de status das regras de configuração fornecidas.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Saída:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                 :
```

```

LastErrorMessage      :
LastFailedEvaluationTime : 1/1/0001 12:00:00 AM
LastFailedInvocationTime : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn         : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId          : config-rule-z1s23b
ConfigRuleName        : vpc-flow-logs-enabled
FirstActivatedTime    : 8/14/2019 6:23:44 AM
FirstEvaluationStarted : True
LastErrorCode          :
LastErrorMessage      :
LastFailedEvaluationTime : 1/1/0001 12:00:00 AM
LastFailedInvocationTime : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM

```

- Para obter detalhes da API, consulte [DescribeConfigRuleEvaluationStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigurationAggregatorList

O código de exemplo a seguir mostra como usar `Get-CFGConfigurationAggregatorList`.

### Ferramentas para PowerShell

Exemplo 1: Essa amostra retorna todos os agregadores da região/conta.

```
Get-CFGConfigurationAggregatorList
```

Saída:

```

AccountAggregationSources      :
  {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn     : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabca1me
ConfigurationAggregatorName    : IrelandMaster
CreationTime                   : 8/25/2019 11:42:39 PM
LastUpdatedTime               : 8/25/2019 11:42:39 PM
OrganizationAggregationSource  :

```

```

AccountAggregationSources      : {}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-qubqabcd
ConfigurationAggregatorName   : raju
CreationTime                   : 8/11/2019 8:39:25 AM
LastUpdatedTime               : 8/11/2019 8:39:25 AM
OrganizationAggregationSource :
    Amazon.ConfigService.Model.OrganizationAggregationSource

```

- Para obter detalhes da API, consulte [DescribeConfigurationAggregators](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigurationAggregatorSourcesStatus

O código de exemplo a seguir mostra como usar Get-CFGConfigurationAggregatorSourcesStatus.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo exibe os campos solicitados para as fontes em determinado agregador.

```

Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
select SourceType, LastUpdateStatus, LastUpdateTime, SourceId

```

Saída:

SourceType	LastUpdateStatus	LastUpdateTime	SourceId
ORGANIZATION	SUCCEEDED	12/31/2019 7:45:06 AM	Organization
ACCOUNT	SUCCEEDED	12/31/2019 7:09:38 AM	612641234567
ACCOUNT	SUCCEEDED	12/31/2019 7:12:53 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:18:10 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:25:17 AM	933301234567
ACCOUNT	SUCCEEDED	12/31/2019 7:25:49 AM	612641234567
ACCOUNT	SUCCEEDED	12/31/2019 7:26:11 AM	612641234567

- Para obter detalhes da API, consulte [DescribeConfigurationAggregatorSourcesStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigurationRecorder

O código de exemplo a seguir mostra como usar Get-CFGConfigurationRecorder.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna os detalhes dos gravadores de configuração.

```
Get-CFGConfigurationRecorder | Format-List
```

#### Saída:

```
Name           : default
RecordingGroup  : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Para obter detalhes da API, consulte [DescribeConfigurationRecorders](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConfigurationRecorderStatus

O código de exemplo a seguir mostra como usar Get-CFGConfigurationRecorderStatus.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna o status dos gravadores de configuração.

```
Get-CFGConfigurationRecorderStatus
```

#### Saída:

```
LastErrorCode      :
LastErrorMessage   :
LastStartTime      : 10/11/2019 10:13:51 AM
LastStatus         : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime       : 10/11/2019 10:13:46 AM
Name               : default
Recording          : True
```

- Para obter detalhes da API, consulte [DescribeConfigurationRecorderStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGConformancePack

O código de exemplo a seguir mostra como usar Get-CFGConformancePack.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todos os pacotes de conformidade.

```
Get-CFGConformancePack
```

Saída:

```
ConformancePackArn      : arn:aws:config:eu-west-1:123456789012:conformance-
pack/dono/conformance-pack-p0acq8bpz
ConformancePackId      : conformance-pack-p0acabcde
ConformancePackInputParameters : {}
ConformancePackName    : dono
CreatedBy              :
DeliveryS3Bucket       : kt-ps-examples
DeliveryS3KeyPrefix    :
LastUpdateRequestedTime : 12/31/2019 8:45:31 AM
```

- Para obter detalhes da API, consulte [DescribeConformancePacks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-CFGDeliveryChannel

O código de exemplo a seguir mostra como usar Get-CFGDeliveryChannel.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera o canal de entrega da região e exibe detalhes.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
S3KeyPrefix,
@{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```



**Saída:**

```

Name      S3BucketName      S3KeyPrefix DeliveryFrequency
----      -
default  config-bucket-NA my      TwentyFour_Hours

```

- Para obter detalhes da API, consulte [DescribeDeliveryChannels](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-CFGResourceTag**

O código de exemplo a seguir mostra como usar Get-CFGResourceTag.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo lista as tags associadas para o determinado recurso

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

**Saída:**

```

Key      Value
---      -
Version  1.3

```

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-CFGConformancePack**

O código de exemplo a seguir mostra como usar Remove-CFGConformancePack.

**Ferramentas para PowerShell**

Exemplo 1: Esta amostra remove o pacote de conformidade fornecido, junto com todas as regras, ações de remediação e resultados de avaliação do pacote.

```
Remove-CFGConformancePack -ConformancePackName dono
```

## Saída:

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

- Para obter detalhes da API, consulte [DeleteConformancePack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-CFGConformancePack

O código de exemplo a seguir mostra como usar Write-CFGConformancePack.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um pacote de conformidade, buscando o modelo do arquivo yaml fornecido.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket kt-ps-examples
-TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- Para obter detalhes da API, consulte [PutConformancePack](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-CFGDeliveryChannel

O código de exemplo a seguir mostra como usar Write-CFGDeliveryChannel.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera a propriedade deliveryFrequency de um canal de entrega existente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName config-
bucket-NA -DeliveryChannel_S3KeyPrefix my
```

- Para obter detalhes da API, consulte [PutDeliveryChannel](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de Device Farm usando o Tools for PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Device Farm.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### New-DFUpload

O código de exemplo a seguir mostra como usar New-DFUpload.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um upload AWS do Device Farm para um aplicativo Android. Você pode obter o ARN do projeto a partir da saída de New-DFProject ou Get-DF.ProjectList Use o URL assinado na saída New-DFUpload para carregar um arquivo no Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
"arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Para obter detalhes da API, consulte [CreateUpload](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Directory Service exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Directory Service.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-DSIpRoute**

O código de exemplo a seguir mostra como usar Add-DSIpRoute.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando remove a tag de recurso atribuída ao ID de diretório especificado

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- Para obter detalhes da API, consulte [AddIpRoutes](#) em Referência de AWS Tools for PowerShell cmdlet.

#### **Add-DSResourceTag**

O código de exemplo a seguir mostra como usar Add-DSResourceTag.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando adiciona a tag de recurso ao ID do diretório especificado

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Para obter detalhes da API, consulte [AddTagsToResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Approve-DSTrust

O código de exemplo a seguir mostra como usar Approve-DSTrust.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo chama a operação da VerifyTrust API AWS Directory Service para o Trustid especificado.

```
Approve-DSTrust -TrustId t-9067157123
```

- Para obter detalhes da API, consulte [VerifyTrust](#) em Referência de AWS Tools for PowerShell cmdlet.

## Confirm-DSSharedDirectory

O código de exemplo a seguir mostra como usar Confirm-DSSharedDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo aceita uma solicitação de compartilhamento de diretório enviada pelo proprietário do diretório Conta da AWS.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Saída:

```
CreatedDateTime      : 12/30/2019 4:20:27 AM
LastUpdatedDateTime : 12/30/2019 4:21:40 AM
OwnerAccountId       : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId      : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod          :
```

```
ShareNotes      : This is test sharing
ShareStatus     : Sharing
```

- Para obter detalhes da API, consulte [AcceptSharedDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Connect-DSDirectory

O código de exemplo a seguir mostra como usar Connect-DSDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um AD Connector para se conectar a um diretório local.

```
Connect-DSDirectory -Name contoso.com -ConnectSettings_CustomerUserName
Administrator -Password $Password -ConnectSettings_CustomerDnsIp 172.31.36.96
-ShortName CONTOSO -Size Small -ConnectSettings_VpcId vpc-123459da -
ConnectSettings_SubnetId subnet-1234ccaa, subnet-5678ffbb
```

- Para obter detalhes da API, consulte [ConnectDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Deny-DSSharedDirectory

O código de exemplo a seguir mostra como usar Deny-DSSharedDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo rejeita uma solicitação de compartilhamento de diretório enviada da conta do proprietário do diretório.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Saída:

```
d-9067012345
```

- Para obter detalhes da API, consulte [RejectSharedDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-DSDirectoryShare

O código de exemplo a seguir mostra como usar `Disable-DSDirectoryShare`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo interrompe o compartilhamento de diretórios entre o proprietário do diretório e a conta do consumidor.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -  
UnshareTarget_Type ACCOUNT
```

Saída:

```
d-9067012345
```

- Para obter detalhes da API, consulte [UnshareDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-DSLADAPS

O código de exemplo a seguir mostra como usar `Disable-DSLADAPS`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa as chamadas seguras LDAP para o diretório especificado.

```
Disable-DSLADAPS -DirectoryId d-123456ijkl -Type Client
```

- Para obter detalhes da API, consulte [DisableLDAPs em Cmdlet Reference](#).AWS Tools for PowerShell

## Disable-DSRadius

O código de exemplo a seguir mostra como usar `Disable-DSRadius`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o servidor RADIUS configurado para um diretório AD Connector ou Microsoft AD.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Para obter detalhes da API, consulte [DisableRadius](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-DSSso

O código de exemplo a seguir mostra como usar `Disable-DSSso`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o login único para um diretório.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Para obter detalhes da API, consulte [DisableSso](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-DSDirectoryShare

O código de exemplo a seguir mostra como usar `Enable-DSDirectoryShare`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo compartilha um diretório específico em sua AWS conta com outra AWS conta usando o método Handshake.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -  
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Saída:

```
d-9067012345
```

- Para obter detalhes da API, consulte [ShareDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.



## Enable-DSLdapS

O código de exemplo a seguir mostra como usar Enable-DSLdapS.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo ativa o switch do diretório específico para sempre usar chamadas seguras LDAP.

```
Enable-DSLdapS -DirectoryId d-123456ijkl -Type Client
```

- Para obter detalhes da API, consulte [EnableLDAPs em Cmdlet Reference](#).AWS Tools for PowerShell

## Enable-DSRadius

O código de exemplo a seguir mostra como usar Enable-DSRadius.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita a autenticação multifator (MFA) com a configuração do servidor RADIUS fornecida para um AD Connector ou um diretório do Microsoft AD.

```
Enable-DSRadius -DirectoryId d-123456ijkl  
-RadiusSettings_AuthenticationProtocol PAP  
-RadiusSettings_DisplayLabel Radius  
-RadiusSettings_RadiusPort 1812  
-RadiusSettings_RadiusRetry 4  
-RadiusSettings_RadiusServer 10.4.185.113  
-RadiusSettings_RadiusTimeout 50  
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- Para obter detalhes da API, consulte [EnableRadius](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-DSSso

O código de exemplo a seguir mostra como usar Enable-DSSso.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita o login único para um diretório.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Para obter detalhes da API, consulte [EnableSso](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSCertificate

O código de exemplo a seguir mostra como usar Get-DSCertificate.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe informações sobre o certificado registrado para uma conexão LDAP segura.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Saída:

```
CertificateId      : c-906731e34f
CommonName         : contoso-EC2AMAZ-CTGG2NM-CA
ExpiryDateTime     : 4/15/2025 6:34:15 PM
RegisteredDateTime : 4/15/2020 6:38:56 PM
State              : Registered
StateReason        : Certificate registered successfully.
```

- Para obter detalhes da API, consulte [DescribeCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSCertificateList

O código de exemplo a seguir mostra como usar Get-DSCertificateList.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todos os certificados registrados para uma conexão LDAP segura para o diretório especificado.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Saída:

CertificateId	CommonName	ExpiryDateTime	State
c-906731e34f	contoso-EC2AMAZ-CTGG2NM-CA	4/15/2025 6:34:15 PM	Registered

- Para obter detalhes da API, consulte [ListCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSConditionalForwarder

O código de exemplo a seguir mostra como usar `Get-DSConditionalForwarder`.

Ferramentas para PowerShell

Exemplo 1: Esse comando obtém todos os encaminhadores condicionais configurados de determinado ID de diretório.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Saída:

DnsIpAddrs	RemoteDomainName	ReplicationScope
{172.31.77.239}	contoso.com	Domain

- Para obter detalhes da API, consulte [DescribeConditionalForwarders](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSDirectory

O código de exemplo a seguir mostra como usar `Get-DSDirectory`.

Ferramentas para PowerShell

Exemplo 1: Esse comando obtém informações sobre os diretórios que pertencem a essa conta.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Saída:

DirectoryId	Name	DnsIpAddrs	Type
-----	----	-----	----
d-123456abcd	abcd.example.com	{172.31.74.189, 172.31.13.145}	SimpleAD
d-123456efgh	wifi.example.com	{172.31.16.108, 172.31.10.56}	ADConnector
d-123456ijkl	lan2.example.com	{172.31.10.56, 172.31.16.108}	MicrosoftAD

- Para obter detalhes da API, consulte [DescribeDirectories](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSDirectoryLimit

O código de exemplo a seguir mostra como usar `Get-DSDirectoryLimit`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exibe as informações de limite de diretório para a região us-east-1.

```
Get-DSDirectoryLimit -Region us-east-1
```

Saída:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit       : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- Para obter detalhes da API, consulte [GetDirectoryLimits](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSDomainControllerList

O código de exemplo a seguir mostra como usar `Get-DSDomainControllerList`.

## Ferramentas para PowerShell

Exemplo 1: Esse comando obtém a lista detalhada dos controladores de domínio lançados para o ID de diretório mencionado

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Saída:

```
AvailabilityZone      : us-east-1b
DirectoryId          : d-123456ijkl
DnsIpAddr            : 172.31.16.108
DomainControllerId   : dc-1234567aa6
LaunchTime           : 4/4/2019 4:53:43 AM
Status               : Active
StatusLastUpdatedDateTime : 4/24/2019 1:37:54 PM
StatusReason         :
SubnetId             : subnet-1234kkaa
VpcId                : vpc-123459d

AvailabilityZone      : us-east-1d
DirectoryId          : d-123456ijkl
DnsIpAddr            : 172.31.10.56
DomainControllerId   : dc-1234567aa7
LaunchTime           : 4/4/2019 4:53:43 AM
Status               : Active
StatusLastUpdatedDateTime : 4/4/2019 5:14:31 AM
StatusReason         :
SubnetId             : subnet-5678ffbb
VpcId                : vpc-123459d
```

- Para obter detalhes da API, consulte [DescribeDomainControllers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSEventTopic

O código de exemplo a seguir mostra como usar `Get-DSEventTopic`.

## Ferramentas para PowerShell

Exemplo 1: Esse comando mostra informações do tópico SNS configurado para notificação enquanto o status do diretório é alterado.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Saída:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status         : Registered
TopicArn       : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName      : snstopicname
```

- Para obter detalhes da API, consulte [DescribeEventTopics](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSIpRouteList

O código de exemplo a seguir mostra como usar Get-DSIpRouteList.

Ferramentas para PowerShell

Exemplo 1: Esse comando obtém os blocos de endereços IP públicos configurados no Roteamento IP do Diretório

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Saída:

```
AddedDateTime      : 12/13/2019 12:27:22 PM
CidrIp             : 203.0.113.5/32
Description        : Public IP of On-Prem DNS Server
DirectoryId       : d-123456ijkl
IpRouteStatusMsg  : Added
IpRouteStatusReason :
```

- Para obter detalhes da API, consulte [ListIpRoutes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSLdapSSetting

O código de exemplo a seguir mostra como usar Get-DSLdapSSetting.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o status da segurança LDAP para o diretório especificado.

```
Get-DSLdapSetting -DirectoryId d-123456ijkl
```

Saída:

```
LastUpdatedDateTime  LDAPStatus LDAPStatusReason
-----
4/15/2020 6:51:03 PM Enabled      LDAPS is enabled successfully.
```

- Para obter detalhes da API, consulte [DescribeDapsSettings](#) em Cmdlet Reference.AWS Tools for PowerShell

## Get-DSLogSubscriptionList

O código de exemplo a seguir mostra como usar Get-DSLogSubscriptionList.

## Ferramentas para PowerShell

Exemplo 1: Esse comando obtém as informações de assinaturas de log do ID de diretório especificado

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Saída:

```
DirectoryId  LogGroupName
SubscriptionCreatedDateTime
-----
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
AM
```

- Para obter detalhes da API, consulte [ListLogSubscriptions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSResourceTag

O código de exemplo a seguir mostra como usar Get-DSResourceTag.

### Ferramentas para PowerShell

Exemplo 1: Este comando obtém todas as tags do diretório especificado.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Saída:

```
Key    Value
---    -
myTag  myTagValue
```

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSSchemaExtension

O código de exemplo a seguir mostra como usar Get-DSSchemaExtension.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todas as extensões de esquema aplicadas a um diretório do Microsoft AD.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Saída:

```
Description           : ManagedADSchemaExtension
DirectoryId           : d-123456ijkl
EndDateTime           : 4/12/2020 10:30:49 AM
SchemaExtensionId     : e-9067306643
SchemaExtensionStatus : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartDateTime         : 4/12/2020 10:28:42 AM
```



- Para obter detalhes da API, consulte [ListSchemaExtensions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSSharedDirectory

O código de exemplo a seguir mostra como usar Get-DSSharedDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém os diretórios compartilhados da sua conta AWS

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Saída:

```
CreatedDateTime      : 12/30/2019 4:34:37 AM
LastUpdatedDateTime  : 12/30/2019 4:35:22 AM
OwnerAccountId       : 123456781234
OwnerDirectoryId     : d-123456ijkl
SharedAccountId      : 123456784321
SharedDirectoryId    : d-9067012345
ShareMethod          : HANDSHAKE
ShareNotes           : This is a test Sharing
ShareStatus          : Shared
```

- Para obter detalhes da API, consulte [DescribeSharedDirectories](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSSnapshot

O código de exemplo a seguir mostra como usar Get-DSSnapshot.

### Ferramentas para PowerShell

Exemplo 1: Esse comando obtém informações sobre os instantâneos de diretório especificados que pertencem a essa conta.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Saída:

```

DirectoryId : d-123456ijkl
Name       :
SnapshotId : s-9064bd1234
StartTime  : 12/13/2019 6:33:01 PM
Status     : Completed
Type       : Auto

DirectoryId : d-123456ijkl
Name       :
SnapshotId : s-9064bb4321
StartTime  : 12/9/2019 9:48:11 PM
Status     : Completed
Type       : Auto

```

- Para obter detalhes da API, consulte [DescribeSnapshots](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSSnapshotLimit

O código de exemplo a seguir mostra como usar `Get-DSSnapshotLimit`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando obtém os limites manuais de instantâneos para um diretório especificado.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Saída:

```

ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
-----
0                            5                            False

```

- Para obter detalhes da API, consulte [GetSnapshotLimits](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DSTrust

O código de exemplo a seguir mostra como usar `Get-DSTrust`.

## Ferramentas para PowerShell

Exemplo 1: Esse comando obtém as informações das relações de confiança criadas para o ID de diretório especificado.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Saída:

```
CreatedDateTime      : 7/5/2019 4:55:42 AM
DirectoryId         : d-123456abcd
LastUpdatedDateTime : 7/5/2019 4:56:04 AM
RemoteDomainName    : contoso.com
SelectiveAuth       : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection      : One-Way: Incoming
TrustId             : t-9067157123
TrustState          : Created
TrustStateReason    :
TrustType           : Forest
```

- Para obter detalhes da API, consulte [DescribeTrusts](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSAlias

O código de exemplo a seguir mostra como usar New-DSAlias.

## Ferramentas para PowerShell

Exemplo 1: Esse comando cria um alias para um diretório e atribui o alias ao id de diretório especificado.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Saída:

```
Alias      DirectoryId
-----      -
myorgname d-123456ijkl
```

- Para obter detalhes da API, consulte [CreateAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSComputer

O código de exemplo a seguir mostra como usar `New-DSComputer`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo objeto de computador do Active Directory.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password $Password
```

Saída:

```
ComputerAttributes          ComputerId
-----
ComputerName
-----
-----
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
ADMemberServer
```

- Para obter detalhes da API, consulte [CreateComputer](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSConditionalForwarder

O código de exemplo a seguir mostra como usar `New-DSConditionalForwarder`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um encaminhador condicional no ID de diretório especificado AWS .

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Para obter detalhes da API, consulte [CreateConditionalForwarder](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSDirectory

O código de exemplo a seguir mostra como usar New-DSDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo diretório Simple AD.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -  
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para obter detalhes da API, consulte [CreateDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSLogSubscription

O código de exemplo a seguir mostra como usar New-DSLogSubscription.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma assinatura para encaminhar registros de segurança do controlador de domínio do Directory Service em tempo real para o grupo de CloudWatch registros da Amazon especificado em seu Conta da AWS.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/  
d-123456ijkl-lan2.example.com
```

- Para obter detalhes da API, consulte [CreateLogSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSMicrosoftAD

O código de exemplo a seguir mostra como usar New-DSMicrosoftAD.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo Microsoft AD Directory em Nuvem AWS.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -  
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para obter detalhes da API, consulte [CreateMicrosoftAD](#) em Referência de AWS Tools for PowerShell Cmdlet.

## New-DSSnapshot

O código de exemplo a seguir mostra como usar New-DSSnapshot.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um instantâneo do diretório

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Para obter detalhes da API, consulte [CreateSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DSTrust

O código de exemplo a seguir mostra como usar New-DSTrust.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria confiança bidirecional em toda a floresta entre seu diretório gerenciado AWS do Microsoft AD e o Microsoft Active Directory local existente.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection  
Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr  
172.31.36.96
```

Saída:

```
t-9067157123
```

- Para obter detalhes da API, consulte [CreateTrust](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-DSCertificate

O código de exemplo a seguir mostra como usar Register-DSCertificate.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo registra um certificado para conexão LDAP segura.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Saída:

```
c-906731e350
```

- Para obter detalhes da API, consulte [RegisterCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-DSEventTopic

O código de exemplo a seguir mostra como usar Register-DSEventTopic.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo associa um diretório como editor a um tópico do SNS.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para obter detalhes da API, consulte [RegisterEventTopic](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSConditionalForwarder

O código de exemplo a seguir mostra como usar Remove-DSConditionalForwarder.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o encaminhador condicional que foi configurado para seu AWS Diretório.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
contoso.com
```

- Para obter detalhes da API, consulte [DeleteConditionalForwarder](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSDirectory

O código de exemplo a seguir mostra como usar Remove-DSDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui um AWS diretório de serviços de diretório (Simple AD/Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Para obter detalhes da API, consulte [DeleteDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSIpRoute

O código de exemplo a seguir mostra como usar Remove-DSIpRoute.

### Ferramentas para PowerShell

Exemplo 1: Esse comando remove o IP especificado das rotas IP configuradas do Directory-ID.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Para obter detalhes da API, consulte [RemovelpRoutes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSLogSubscription

O código de exemplo a seguir mostra como usar Remove-DSLogSubscription.

### Ferramentas para PowerShell

Exemplo 1: Esse comando remove a assinatura de log do ID de diretório especificado

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```



- Para obter detalhes da API, consulte [DeleteLogSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSResourceTag

O código de exemplo a seguir mostra como usar Remove-DSResourceTag.

Ferramentas para PowerShell

Exemplo 1: Esse comando remove a tag de recurso atribuída ao ID de diretório especificado

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Para obter detalhes da API, consulte [RemoveTagsFromResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSSnapshot

O código de exemplo a seguir mostra como usar Remove-DSSnapshot.

Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o instantâneo criado manualmente.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Para obter detalhes da API, consulte [DeleteSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DSTrust

O código de exemplo a seguir mostra como usar Remove-DSTrust.

Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a relação de confiança existente entre seu diretório AWS gerenciado do AD e um domínio externo.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

**Saída:**

```
t-9067157123
```

- Para obter detalhes da API, consulte [DeleteTrust](#) em Referência de AWS Tools for PowerShell cmdlet.

**Reset-DSUserPassword**

O código de exemplo a seguir mostra como usar `Reset-DSUserPassword`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo redefine a senha do usuário do Active Directory chamado `ADUser` no AWS Managed microsoft AD ou Simple AD Directory

```
Reset-DSUserPassword -UserName ADUser -DirectoryId d-123456ijkl -NewPassword  
$Password
```

- Para obter detalhes da API, consulte [ResetUserPassword](#) em Referência de AWS Tools for PowerShell cmdlet.

**Restore-DSFromSnapshot**

O código de exemplo a seguir mostra como usar `Restore-DSFromSnapshot`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo restaura um diretório usando um instantâneo de diretório existente.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Para obter detalhes da API, consulte [RestoreFromSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

**Set-DSDomainControllerCount**

O código de exemplo a seguir mostra como usar `Set-DSDomainControllerCount`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo define o número do controlador de domínio como 3 para o ID de diretório especificado.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Para obter detalhes da API, consulte [UpdateNumberOfDomainControllers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-DSSchemaExtension

O código de exemplo a seguir mostra como usar `Start-DSSchemaExtension`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo aplica uma extensão de esquema a um diretório do Microsoft AD.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Saída:

```
e-9067306643
```

- Para obter detalhes da API, consulte [StartSchemaExtension](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-DSSchemaExtension

O código de exemplo a seguir mostra como usar `Stop-DSSchemaExtension`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela uma extensão de esquema em andamento para um diretório do Microsoft AD.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Para obter detalhes da API, consulte [CancelSchemaExtension](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-DSCertificate

O código de exemplo a seguir mostra como usar `Unregister-DSCertificate`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui do sistema o certificado que foi registrado para uma conexão LDAP segura.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Para obter detalhes da API, consulte [DeregisterCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-DSEventTopic

O código de exemplo a seguir mostra como usar `Unregister-DSEventTopic`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o diretório especificado como editor do tópico SNS especificado.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para obter detalhes da API, consulte [DeregisterEventTopic](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-DSConditionalForwarder

O código de exemplo a seguir mostra como usar `Update-DSConditionalForwarder`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza um encaminhador condicional que foi configurado para seu AWS diretório.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Para obter detalhes da API, consulte [UpdateConditionalForwarder](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-DSRadius

O código de exemplo a seguir mostra como usar Update-DSRadius.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza as informações do servidor RADIUS para um diretório AD Connector ou Microsoft AD.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Para obter detalhes da API, consulte [UpdateRadius](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-DSTrust

O código de exemplo a seguir mostra como usar Update-DSTrust.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza o SelectiveAuth parâmetro do trust-id especificado de Desativado para Ativado.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Saída:

```
RequestId                                TrustId
-----                                -
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- Para obter detalhes da API, consulte [UpdateTrust](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS DMS exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS DMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **New-DMSReplicationTask**

O código de exemplo a seguir mostra como usar New-DMSReplicationTask.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova tarefa de replicação do AWS Database Migration Service que usa CdcStartTime em vez de CdcStartPosition. O MigrationType é definido como "full-load-and-cdc", o que significa que a tabela de destino deve estar vazia. A nova tarefa é marcada com uma tag que tem uma chave de Stage e um valor-chave de Test. Para obter mais informações sobre os valores usados por esse cmdlet, consulte Creating a Task ([https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Tasks.Creating.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html)) no AWS Database Migration Service User Guide.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE" `
  -CdcStartTime "2019-08-08T12:12:12" `
  -CdcStopPosition "server_time:2019-08-09T12:12:12" `
  -MigrationType "full-load-and-cdc" `
```

```
-ReplicationTaskIdentifier "task1" `
-ReplicationTaskSetting "" `
-SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE" `
-TableMapping "file:///home/testuser/table-mappings.json" `
-Tag @{"Key"="Stage";"Value"="Test"} `
-TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Para obter detalhes da API, consulte [CreateReplicationTask](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do DynamoDB usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-DDBIndexSchema**

O código de exemplo a seguir mostra como usar Add-DDBIndexSchema.

### Ferramentas para PowerShell

Exemplo 1: cria um TableSchema objeto vazio e adiciona uma nova definição de índice secundário local a ele antes de gravar o TableSchema objeto no pipeline.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Saída:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----
-----	
{LastPostDateTime}	{}
{LastPostIndex}	

Exemplo 2: adiciona uma nova definição de índice secundário local ao TableSchema objeto fornecido antes de gravar o TableSchema objeto de volta no pipeline. O TableSchema objeto também pode ser fornecido usando o parâmetro -Schema.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Saída:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----
-----	
{LastPostDateTime}	{}
{LastPostIndex}	

- Para obter detalhes da API, consulte [Add-DDB IndexSchema](#) na Referência do AWS Tools for PowerShell Cmdlet.

## Add-DDBKeySchema

O código de exemplo a seguir mostra como usar Add-DDBKeySchema.

### Ferramentas para PowerShell

Exemplo 1: cria um TableSchema objeto vazio e adiciona entradas de definição de chave e atributo a ele usando os dados-chave especificados antes de gravar o TableSchema objeto no



pipeline. O tipo de chave é declarado como 'HASH' por padrão; use o KeyType parâmetro - com um valor de 'RANGE' para declarar uma chave de intervalo.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Saída:

```
AttributeSchema                                KeySchema
  LocalSecondaryIndexSchema                    -----
-----
  -----
{ForumName}                                    {ForumName}
  {}
```

Exemplo 2: adiciona novas entradas de definição de chave e atributo ao TableSchema objeto fornecido antes de gravar o TableSchema objeto no pipeline. O tipo de chave é declarado como 'HASH' por padrão; use o KeyType parâmetro - com um valor de 'RANGE' para declarar uma chave de intervalo. O TableSchema objeto também pode ser fornecido usando o parâmetro - Schema.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Saída:

```
AttributeSchema                                KeySchema
  LocalSecondaryIndexSchema                    -----
-----
  -----
{ForumName}                                    {ForumName}
  {}
```

- Para obter detalhes da API, consulte [Add-DDB KeySchema](#) na Referência do AWS Tools for PowerShell Cmdlet.

## ConvertFrom-DDBItem

O código de exemplo a seguir mostra como usar ConvertFrom-DDBItem.

## Ferramentas para PowerShell

Exemplo 1: `ConvertFrom-DDBItem` é usado para converter o resultado de `Get-DDBItem` de uma tabela de hash do DynamoDB em uma tabela de hash de tipos comuns, como string e `AttributeValues` double.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

Name	Value
----	----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- Para obter detalhes da API, consulte [ConvertFrom-DDBItem na AWS Tools for PowerShell Referência do Cmdlet](#).

## ConvertTo-DDBItem

O código de exemplo a seguir mostra como usar `ConvertTo-DDBItem`.

## Ferramentas para PowerShell

Exemplo 1: Um exemplo de conversão de uma tabela de hash em um dicionário de valores de atributos do DynamoDB.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem
```

Key	Value
---	-----
SongTitle	Amazon.DynamoDBv2.Model.AttributeValue
Artist	Amazon.DynamoDBv2.Model.AttributeValue

Exemplo 2: Um exemplo de conversão de uma tabela de hash em um dicionário de valores de atributos do DynamoDB.

```
@{
    MyMap      = @{
        MyString = 'my string'
    }
    MyStringSet = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string'))))
    )
    MyList1     = @('my', 'string')
    MyList2     = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3     = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

Saída:

Key	Value
---	-----
MyStringSet	Amazon.DynamoDBv2.Model.AttributeValue
MyList1	Amazon.DynamoDBv2.Model.AttributeValue
MyNumericSet	Amazon.DynamoDBv2.Model.AttributeValue
MyList2	Amazon.DynamoDBv2.Model.AttributeValue
MyBinarySet	Amazon.DynamoDBv2.Model.AttributeValue
MyMap	Amazon.DynamoDBv2.Model.AttributeValue
MyList3	Amazon.DynamoDBv2.Model.AttributeValue

- Para obter detalhes da API, consulte [ConvertTo-DDBItem na AWS Tools for PowerShell Referência do Cmdlet](#).

## Get-DDBBatchItem

O código de exemplo a seguir mostra como usar Get-DDBBatchItem.

## Ferramentas para PowerShell

Exemplo 1: obtém o item com o SongTitle “Somewhere Down The Road” das tabelas 'Music' e 'Songs' do DynamoDB.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-DDBItem
```

Saída:

Name	Value
----	-----
Artist	No One You Know
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
CriticRating	10
Genre	Country
Price	1.94
Artist	No One You Know
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
CriticRating	10
Genre	Country
Price	1.94

- Para obter detalhes da API, consulte [BatchGetItem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DDBItem

O código de exemplo a seguir mostra como usar Get-DDBItem.

### Ferramentas para PowerShell

Exemplo 1: retorna o item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

Name	Value
----	-----
Genre	Country
SongTitle	Somewhere Down The Road
Price	1.94
Artist	No One You Know
CriticRating	9
AlbumTitle	Somewhat Famous

- Para obter detalhes da API, consulte [GetItem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DDBTable

O código de exemplo a seguir mostra como usar Get-DDBTable.

### Ferramentas para PowerShell

Exemplo 1: exibe detalhes da tabela especificada.

```
Get-DDBTable -TableName "myTable"
```

- Para obter detalhes da API, consulte [DescribeTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-DDBTableList

O código de exemplo a seguir mostra como usar `Get-DDBTableList`.

### Ferramentas para PowerShell

Exemplo 1: exibe detalhes de todas as tabelas, iterando automaticamente até que o serviço indique que não existem mais tabelas.

```
Get-DDBTableList
```

Exemplo 2: itera manualmente detalhes de todas as tabelas, exibindo até dez tabelas por chamada até que o serviço indique que não existem mais tabelas.

```
$nextToken = $null
do {
    Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
    $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [ListTables](#) em Referência de AWS Tools for PowerShell cmdlet.

## Invoke-DDBQuery

O código de exemplo a seguir mostra como usar `Invoke-DDBQuery`.

### Ferramentas para PowerShell

Exemplo 1: invoca uma consulta que retorna itens do DynamoDB com o especificado e o artista. `SongTitle`

```
$invokeDDBQuery = @{
    TableName = 'Music'
```

```

KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
ExpressionAttributeValues = @{
    ':SongTitle' = 'Somewhere Down The Road'
    ':Artist' = 'No One You Know'
} | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem

```

**Saída:**

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- Para ter detalhes da API, consulte [Query](#) em AWS Tools for PowerShell Cmdlet Reference.

**Invoke-DDBScan**

O código de exemplo a seguir mostra como usar Invoke-DDBScan.

**Ferramentas para PowerShell**

Exemplo 1: exibe todos os itens da tabela Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

**Saída:**

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous
Genre	Country

Artist	No One You Know
Price	1.98
CriticRating	8.4
SongTitle	My Dog Spot
AlbumTitle	Hey Now

Exemplo 2: Retorna itens na tabela Música com um valor CriticRating maior ou igual a nove.

```
$scanFilter = @{
    CriticRating = [Amazon.DynamoDBv2.Model.Condition]{
        AttributeValueList = @( @{N = '9'} )
        ComparisonOperator = 'GE'
    }
}
Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Saída:

Name	Value
----	-----
Genre	Country
Artist	No One You Know
Price	1.94
CriticRating	9
SongTitle	Somewhere Down The Road
AlbumTitle	Somewhat Famous

- Para ter detalhes da API, consulte [Scan](#) em AWS Tools for PowerShell Cmdlet Reference.

## New-DDBTable

O código de exemplo a seguir mostra como usar New-DDBTable.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
```



```
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

**Saída:**

```
AttributeDefinitions : {ForumName, Subject}
TableName            : Thread
KeySchema            : {ForumName, Subject}
TableStatus          : CREATING
CreationDateTime     : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes       : 0
ItemCount            : 0
LocalSecondaryIndexes : {}
```

Exemplo 2: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). Um índice secundário local também é definido. A chave do índice secundário local será definida automaticamente a partir da chave de hash primária na tabela (ForumName). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
  "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

**Saída:**

```
AttributeDefinitions : {ForumName, LastPostDateTime, Subject}
TableName            : Thread
KeySchema            : {ForumName, Subject}
TableStatus          : CREATING
CreationDateTime     : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes       : 0
ItemCount            : 0
LocalSecondaryIndexes : {LastPostIndex}
```

Exemplo 3: Este exemplo mostra como usar um único pipeline para criar uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave) e um índice secundário local. O Add-DDB KeySchema e o Add-DDB IndexSchema criam um novo TableSchema objeto para você se um não for fornecido pelo pipeline ou pelo parâmetro -Schema.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
    -RangeKeyName "LastPostDateTime" `
    -RangeKeyDataType "S" `
    -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions : {ForumName, LastPostDateTime, Subject}
TableName            : Thread
KeySchema            : {ForumName, Subject}
TableStatus          : CREATING
CreationDateTime     : 10/28/2013 4:39:49 PM
ProvisionedThroughput : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes       : 0
ItemCount            : 0
LocalSecondaryIndexes : {LastPostIndex}
```

- Para obter detalhes da API, consulte [CreateTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-DDBTableSchema

O código de exemplo a seguir mostra como usar New-DDBTableSchema.

### Ferramentas para PowerShell

Exemplo 1: Cria um TableSchema objeto vazio pronto para aceitar definições de chave e índice para uso na criação de uma nova tabela do Amazon DynamoDB. O objeto retornado pode ser canalizado para os cmdlets Add-DDBKeySchema, Add-DDB IndexSchema e New-DDBtable ou passado para eles usando o parâmetro -Schema em cada cmdlet.

```
New-DDBTableSchema
```

Saída:

```
AttributeSchema                                KeySchema
  LocalSecondaryIndexSchema
-----
-----
{}                                             {}
  {}
```

- Para obter detalhes da API, consulte [New-DDB TableSchema](#) em AWS Tools for PowerShell Cmdlet Reference.

## Remove-DDBItem

O código de exemplo a seguir mostra como usar Remove-DDBItem.

Ferramentas para PowerShell

Exemplo 1: remove o item do DynamoDB que corresponde à chave fornecida.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Para obter detalhes da API, consulte [Deleteltem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-DDBTable

O código de exemplo a seguir mostra como usar Remove-DDBTable.

Ferramentas para PowerShell

Exemplo 1: exclui a tabela especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable"
```

Exemplo 2: exclui a tabela especificada. A confirmação não será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Para obter detalhes da API, consulte [DeleteTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-DDBBatchItem

O código de exemplo a seguir mostra como usar Set-DDBBatchItem.

### Ferramentas para PowerShell

Exemplo 1: cria um item ou substitui um item por um novo item nas tabelas Music e Songs do DynamoDB.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item
```

Saída:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Para obter detalhes da API, consulte [BatchWriteItem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-DDBItem

O código de exemplo a seguir mostra como usar Set-DDBItem.

### Ferramentas para PowerShell

Exemplo 1: cria um item ou substitui um item por um novo item.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Para obter detalhes da API, consulte [PutItem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-DDBItem

O código de exemplo a seguir mostra como usar Update-DDBItem.

### Ferramentas para PowerShell

Exemplo 1: define o atributo de gênero como 'Rap' no item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
```

```
UpdateExpression = 'set Genre = :val1'
ExpressionAttributeValue = (@{
    ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
})
}
Update-DDBItem @updateDdbItem
```

Saída:

Name	Value
----	-----
Genre	Rap

- Para obter detalhes da API, consulte [UpdateItem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-DDBTable

O código de exemplo a seguir mostra como usar Update-DDBTable.

### Ferramentas para PowerShell

Exemplo 1: atualiza os valores de throughput provisionado da tabela especificada.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Para obter detalhes da API, consulte [UpdateTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon EC2 usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-EC2CapacityReservation

O código de exemplo a seguir mostra como usar Add-EC2CapacityReservation.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova reserva de capacidade com os atributos especificados

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

## Saída:

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate           : 3/28/2019 9:29:41 AM
EbsOptimized         : True
EndDate              : 1/1/0001 12:00:00 AM
EndDateType          : unlimited
EphemeralStorage     : False
InstanceMatchCriteria : open
InstancePlatform     : Windows
InstanceType         : m4.xlarge
State                 : active
Tags                  : {}
Tenancy               : default
TotalInstanceCount   : 2
```

- Para obter detalhes da API, consulte [CreateCapacityReservation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-EC2InternetGateway

O código de exemplo a seguir mostra como usar Add-EC2InternetGateway.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa o gateway de Internet especificado à VPC especificada.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: Este exemplo cria uma VPC e um gateway da Internet e, em seguida, conecta o gateway da Internet à VPC.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16  
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Para obter detalhes da API, consulte [AttachInternetGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-EC2NetworkInterface

O código de exemplo a seguir mostra como usar Add-EC2NetworkInterface.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa a interface de rede especificada à instância especificada.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -  
DeviceIndex 1
```

Saída:

```
eni-attach-1a2b3c4d
```

- Para obter detalhes da API, consulte [AttachNetworkInterface](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-EC2Volume

O código de exemplo a seguir mostra como usar Add-EC2Volume.



## Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa o volume especificado à instância especificada e o expõe com o nome do dispositivo especificado.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Saída:

```
AttachTime      : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device          : /dev/sdh
InstanceId      : i-1a2b3c4d
State          : attaching
VolumeId       : vol-12345678
```

- Para obter detalhes da API, consulte [AttachVolume](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-EC2VpnGateway

O código de exemplo a seguir mostra como usar Add-EC2VpnGateway.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa o gateway privado virtual especificado à VPC especificada.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Saída:

```
State      VpcId
-----
attaching  vpc-12345678
```

- Para obter detalhes da API, consulte [AttachVpnGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Approve-EC2VpcPeeringConnection

O código de exemplo a seguir mostra como usar `Approve-EC2VpcPeeringConnection`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo aprova o `pcx-1dfad234b56ff78be` solicitado `VpcPeeringConnectionId`

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

### Saída:

```
AccepterVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime       : 1/1/0001 12:00:00 AM
RequesterVpcInfo     : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status               : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                 : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Para obter detalhes da API, consulte [AcceptVpcPeeringConnection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Confirm-EC2ProductInstance

O código de exemplo a seguir mostra como usar `Confirm-EC2ProductInstance`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo determina se o código do produto especificado está associado à instância especificada.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Para obter detalhes da API, consulte [ConfirmProductInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Copy-EC2Image

O código de exemplo a seguir mostra como usar `Copy-EC2Image`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”. Se -Region não for especificada, a região padrão atual será usada como a região de destino.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2  
-Name "Copy of ami-12345678"
```

Saída:

```
ami-87654321
```

- Para obter detalhes da API, consulte [CopyImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Copy-EC2Snapshot

O código de exemplo a seguir mostra como usar Copy-EC2Snapshot.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo copia o snapshot especificado da região da UE (Irlanda) para a região Oeste dos EUA (Oregon).

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-west-2
```

Exemplo 2: Se você definir uma região padrão e omitir o parâmetro Região, a região de destino padrão será a região padrão.

```
Set-DefaultAWSRegion us-west-2  
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Para obter detalhes da API, consulte [CopySnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## Deny-EC2VpcPeeringConnection

O código de exemplo a seguir mostra como usar Deny-EC2VpcPeeringConnection.

## Ferramentas para PowerShell

Exemplo 1: O exemplo acima nega a solicitação de ID de solicitação VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Para obter detalhes da API, consulte [RejectVpcPeeringConnection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-EC2VgwRoutePropagation

O código de exemplo a seguir mostra como usar Disable-EC2VgwRoutePropagation.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo impede que o VGW propague automaticamente as rotas para a tabela de roteamento especificada.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte [DisableVgwRoutePropagation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-EC2VpcClassicLink

O código de exemplo a seguir mostra como usar Disable-EC2VpcClassicLink.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o EC2 VpcClassicLink para o vpc-01e23c4a5d6db78e9. Ele retorna Verdadeiro ou Falso

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Para obter detalhes da API, consulte [DisableVpcClassicLink](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-EC2VpcClassicLinkDnsSupport

O código de exemplo a seguir mostra como usar `Disable-EC2VpcClassicLinkDnsSupport`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o suporte de ClassicLink DNS para o `vpc-0b12d3456a7e8910d`

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Para obter detalhes da API, consulte [DisableVpcClassicLinkDnsSupport](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-EC2InternetGateway

O código de exemplo a seguir mostra como usar `Dismount-EC2InternetGateway`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo separa o gateway de Internet especificado da VPC especificada.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte [DetachInternetGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-EC2NetworkInterface

O código de exemplo a seguir mostra como usar `Dismount-EC2NetworkInterface`.

Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove o anexo especificado entre uma interface de rede e uma instância.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Para obter detalhes da API, consulte [DetachNetworkInterface](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-EC2Volume

O código de exemplo a seguir mostra como usar Dismount-EC2Volume.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo separa o volume especificado.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Saída:

```
AttachTime      : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device          : /dev/sdh
InstanceId      : i-1a2b3c4d
State           : detaching
VolumeId       : vol-12345678
```

Exemplo 2: Você também pode especificar o ID da instância e o nome do dispositivo para garantir que você esteja desanexando o volume correto.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Para obter detalhes da API, consulte [DetachVolume](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-EC2VpnGateway

O código de exemplo a seguir mostra como usar Dismount-EC2VpnGateway.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo separa o gateway privado virtual especificado da VPC especificada.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte [DetachVpnGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2CapacityReservation

O código de exemplo a seguir mostra como usar `Edit-EC2CapacityReservation`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica o `CapacityReservationId` `cr-0c1f2345db6f7cdba` alterando a contagem de instâncias para 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

Saída:

```
True
```

- Para obter detalhes da API, consulte [ModifyCapacityReservation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2Host

O código de exemplo a seguir mostra como usar `Edit-EC2Host`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica as `AutoPlacement` configurações para desativadas para o host dedicado `h-01e23f4cd567890f3`

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Saída:

```
Successful          Unsuccessful  
-----  
{h-01e23f4cd567890f3} {}
```

- Para obter detalhes da API, consulte [ModifyHosts](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2IdFormat

O código de exemplo a seguir mostra como usar `Edit-EC2IdFormat`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo ativa o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Exemplo 2: Este exemplo desativa o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Para obter detalhes da API, consulte [ModifyIdFormat](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2ImageAttribute

O código de exemplo a seguir mostra como usar `Edit-EC2ImageAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo atualiza a descrição da AMI especificada.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemplo 2: Esse exemplo torna a AMI pública (por exemplo, para que qualquer Conta da AWS possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

Exemplo 3: Esse exemplo torna a AMI privada (por exemplo, para que somente você, como proprietário, possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```



Exemplo 4: Este exemplo concede permissão de lançamento ao especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

Exemplo 5: Este exemplo remove a permissão de lançamento do especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- Para obter detalhes da API, consulte [ModifyImageAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2InstanceAttribute

O código de exemplo a seguir mostra como usar Edit-EC2InstanceAttribute.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo modifica o tipo de instância da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Exemplo 2: Este exemplo habilita redes aprimoradas para a instância especificada, especificando "simple" como o valor do parâmetro de suporte de rede de virtualização de E/S raiz única (SR-IOV), -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Exemplo 3: Esse exemplo modifica os grupos de segurança da instância especificada. A instância deve estar em uma VPC. Você deve especificar a ID de cada grupo de segurança, não o nome.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",  
"sg-45678901" )
```

Exemplo 4: Esse exemplo permite a otimização de E/S do EBS para a instância especificada. Esse recurso não está disponível em todos os tipos de instância. Taxas de uso adicionais se aplicam ao usar uma instância otimizada para EBS.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemplo 5: Este exemplo permite a verificação de origem/destino para a instância especificada. Para que uma instância NAT realize NAT, o valor deve ser 'false'.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Exemplo 6: Este exemplo desativa o encerramento da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Exemplo 7: Esse exemplo altera a instância especificada para que ela seja encerrada quando o desligamento for iniciado a partir da instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior  
terminate
```

- Para obter detalhes da API, consulte [ModifyInstanceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2InstanceCreditSpecification

O código de exemplo a seguir mostra como usar `Edit-EC2InstanceCreditSpecification`.

### Ferramentas para PowerShell

Exemplo 1: Isso habilita créditos ilimitados de T2, por exemplo, i-01234567890abcdef.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest  
$Credit.InstanceId = "i-01234567890abcdef"  
$Credit.CpuCredits = "unlimited"  
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Para obter detalhes da API, consulte [ModifyInstanceCreditSpecification](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2NetworkInterfaceAttribute

O código de exemplo a seguir mostra como usar `Edit-EC2NetworkInterfaceAttribute`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica a interface de rede especificada para que o anexo especificado seja excluído no encerramento.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Exemplo 2: Este exemplo modifica a descrição da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
description"
```

Exemplo 3: Este exemplo modifica o grupo de segurança da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
sg-1a2b3c4d
```

Exemplo 4: Este exemplo desativa a verificação de origem/destino para a interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
>false
```

- Para obter detalhes da API, consulte [ModifyNetworkInterfaceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2ReservedInstance

O código de exemplo a seguir mostra como usar Edit-EC2ReservedInstance.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo modifica a zona de disponibilidade, a contagem de instâncias e a plataforma das instâncias reservadas especificadas.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"
```

```
Edit-EC2ReservedInstance `
-ReservedInstancesId @"FE32132D-70D5-4795-B400-AE435EXAMPLE", "0CC556F3-7AB8-4C00-
B0E5-98666EXAMPLE" `
-TargetConfiguration $config
```

- Para obter detalhes da API, consulte [ModifyReservedInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2SnapshotAttribute

O código de exemplo a seguir mostra como usar Edit-EC2SnapshotAttribute.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo torna público o snapshot especificado definindo seu CreateVolumePermission atributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission -OperationType Add -GroupName all
```

- Para obter detalhes da API, consulte [ModifySnapshotAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2SpotFleetRequest

O código de exemplo a seguir mostra como usar Edit-EC2SpotFleetRequest.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza a capacidade alvo da solicitação de frota spot especificada.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Saída:

```
True
```

- Para obter detalhes da API, consulte [ModifySpotFleetRequest](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2SubnetAttribute

O código de exemplo a seguir mostra como usar `Edit-EC2SubnetAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Exemplo 2: Este exemplo desativa o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Para obter detalhes da API, consulte [ModifySubnetAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2VolumeAttribute

O código de exemplo a seguir mostra como usar `Edit-EC2VolumeAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica o atributo especificado do volume especificado. As operações de E/S do volume são retomadas automaticamente após serem suspensas devido a dados potencialmente inconsistentes.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Para obter detalhes da API, consulte [ModifyVolumeAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-EC2VpcAttribute

O código de exemplo a seguir mostra como usar `Edit-EC2VpcAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita o suporte para nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemplo 2: Este exemplo desativa o suporte para nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemplo 3: Este exemplo permite o suporte à resolução de DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemplo 4: Este exemplo desativa o suporte à resolução de DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Para obter detalhes da API, consulte [ModifyVpcAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-EC2VgwRoutePropagation

O código de exemplo a seguir mostra como usar `Enable-EC2VgwRoutePropagation`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo permite que o VGW especificado propague rotas automaticamente para a tabela de roteamento especificada.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte [EnableVgwRoutePropagation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-EC2VolumeIO

O código de exemplo a seguir mostra como usar `Enable-EC2VolumeIO`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita operações de E/S para o volume especificado, se as operações de E/S estiverem desativadas.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Para obter detalhes da API, consulte [EnableVolumeIo](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-EC2VpcClassicLink

O código de exemplo a seguir mostra como usar `Enable-EC2VpcClassicLink`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita a VPC `vpc-0123456b789b0d12f` para ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Saída:

```
True
```

- Para obter detalhes da API, consulte [EnableVpcClassicLink](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-EC2VpcClassicLinkDnsSupport

O código de exemplo a seguir mostra como usar `Enable-EC2VpcClassicLinkDnsSupport`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo permite que o `vpc-0b12d3456a7e8910d` ofereça suporte à resolução de nome de host DNS para ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Para obter detalhes da API, consulte [EnableVpcClassicLinkDnsSupport](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2AccountAttribute

O código de exemplo a seguir mostra como usar `Get-EC2AccountAttribute`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve se você pode executar instâncias no EC2-Classico e no EC2-VPC na região ou somente no EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Saída:

```
AttributeValue
-----
EC2
VPC
```

Exemplo 2: Este exemplo descreve sua VPC padrão ou é “nenhuma” se você não tiver uma VPC padrão na região.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Saída:

```
AttributeValue
-----
vpc-12345678
```

Exemplo 3: Este exemplo descreve o número máximo de instâncias sob demanda que você pode executar.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Saída:

```
AttributeValue
-----
20
```

- Para obter detalhes da API, consulte [DescribeAccountAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.



## Get-EC2Address

O código de exemplo a seguir mostra como usar Get-EC2Address.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o endereço IP elástico especificado para instâncias no EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Saída:

```
AllocationId      : eipalloc-12345678
AssociationId     : eipassoc-12345678
Domain           : vpc
InstanceId       : i-87654321
NetworkInterfaceId : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress  : 10.0.2.172
PublicIp         : 198.51.100.2
```

Exemplo 2: Este exemplo descreve seus endereços IP elásticos para instâncias em uma VPC. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Exemplo 3: Este exemplo descreve o endereço IP elástico especificado para instâncias no EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Saída:

```
AllocationId      :
AssociationId     :
Domain           : standard
InstanceId       : i-12345678
NetworkInterfaceId :
NetworkInterfaceOwnerId :
```

```
PrivateIpAddress      :
PublicIp              : 203.0.113.17
```

Exemplo 4: Este exemplo descreve seus endereços IP elásticos para instâncias no EC2-Classic. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Exemplo 5: Este exemplo descreve todos os seus endereços IP elásticos.

```
Get-EC2Address
```

Exemplo 6: Este exemplo retorna o IP público e privado para o ID da instância fornecido no filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Saída:

```
PrivateIpAddress PublicIp
-----
10.0.0.99         63.36.5.227
```

Exemplo 7: Este exemplo recupera todos os IPs elásticos com seu ID de alocação, ID de associação e IDs de instância

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
AllocationId, PublicIp
```

Saída:

```
InstanceId           AssociationId           AllocationId           PublicIp
-----
17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
17.212.124.77
17.212.225.7
eipalloc-012345678eeabcfad
```

```
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
37.215.222.270
```

Exemplo 8: Este exemplo busca uma lista de endereços IP do EC2 que correspondem à chave de tag 'Category' com o valor 'Prod'

```
Get-EC2Address -Filter @{"Name"="tag:Category";Values="Prod"}
```

Saída:

```
AllocationId      : eipalloc-0123f456f81a01b58
AssociationId     : eipassoc-0d1b23a456d103810
CustomerOwnedIp  :
CustomerOwnedIpv4Pool :
Domain           : vpc
InstanceId       : i-012e3cb4df567e1aa
NetworkBorderGroup : eu-west-1
NetworkInterfaceId : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress  : 192.168.1.84
PublicIp         : 34.250.81.29
PublicIpv4Pool   : amazon
Tags             : {Category, Name}
```

- Para obter detalhes da API, consulte [DescribeAddresses](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2AvailabilityZone

O código de exemplo a seguir mostra como usar Get-EC2AvailabilityZone.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as zonas de disponibilidade da região atual que estão disponíveis para você.

```
Get-EC2AvailabilityZone
```

Saída:

Messages	RegionName	State	ZoneName
-----	-----	-----	-----
{}	us-west-2	available	us-west-2a
{}	us-west-2	available	us-west-2b
{}	us-west-2	available	us-west-2c

Exemplo 2: Este exemplo descreve todas as zonas de disponibilidade que estão em estado de comprometimento. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Exemplo 3: Com a PowerShell versão 2, você deve usar `New-Object` para criar o filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Para obter detalhes da API, consulte [DescribeAvailabilityZones](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2BundleTask

O código de exemplo a seguir mostra como usar `Get-EC2BundleTask`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a tarefa de pacote especificada.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Exemplo 2: Este exemplo descreve as tarefas do pacote cujo estado é “concluído” ou “falhado”.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
```

```
$filter.Values = @( "complete", "failed" )  
  
Get-EC2BundleTask -Filter $filter
```

- Para obter detalhes da API, consulte [DescribeBundleTasks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2CapacityReservation

O código de exemplo a seguir mostra como usar Get-EC2CapacityReservation.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve uma ou mais de suas reservas de capacidade para a região

```
Get-EC2CapacityReservation -Region eu-west-1
```

Saída:

```
AvailabilityZone      : eu-west-1b  
AvailableInstanceCount : 2  
CapacityReservationId : cr-0c1f2345db6f7cdba  
CreateDate           : 3/28/2019 9:29:41 AM  
EbsOptimized         : True  
EndDate              : 1/1/0001 12:00:00 AM  
EndDateType          : unlimited  
EphemeralStorage     : False  
InstanceMatchCriteria : open  
InstancePlatform     : Windows  
InstanceType         : m4.xlarge  
State                : active  
Tags                 : {}  
Tenancy              : default  
TotalInstanceCount   : 2
```

- Para obter detalhes da API, consulte [DescribeCapacityReservations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ConsoleOutput

O código de exemplo a seguir mostra como usar Get-EC2ConsoleOutput.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém a saída do console para a instância Linux especificada. A saída do console é codificada.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Saída:

```
InstanceId      Output
-----
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Exemplo 2: Este exemplo armazena a saída codificada do console em uma variável e, em seguida, a decodifica.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- Para obter detalhes da API, consulte [GetConsoleOutput](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2CustomerGateway

O código de exemplo a seguir mostra como usar Get-EC2CustomerGateway.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o gateway do cliente especificado.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
BgpAsn          : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress        : 203.0.113.12
State            : available
Tags             : {}
Type             : ipsec.1
```

Exemplo 2: Este exemplo descreve qualquer gateway de cliente cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Exemplo 3: Este exemplo descreve todos os gateways de seus clientes.

```
Get-EC2CustomerGateway
```

- Para obter detalhes da API, consulte [DescribeCustomerGateways](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2DhcpOption

O código de exemplo a seguir mostra como usar Get-EC2DhcpOption.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista seus conjuntos de opções de DHCP.

```
Get-EC2DhcpOption
```

Saída:

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

Exemplo 2: Este exemplo obtém detalhes de configuração para o conjunto de opções DHCP especificado.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

**Saída:**

Key	Values
---	-----
domain-name	{abc.local}
domain-name-servers	{10.0.0.101, 10.0.0.102}

- Para obter detalhes da API, consulte [DescribeDhcpOptions](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2FlowLog**

O código de exemplo a seguir mostra como usar Get-EC2FlowLog.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve um ou mais registros de fluxo com o tipo de destino de log 's3'

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

**Saída:**

```
CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :
DeliverLogsStatus      : SUCCESS
FlowLogId              : f1-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination         : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType     : s3
LogGroupName           :
ResourceId             : eni-01d2dda3456b7e890
TrafficType            : ALL
```

- Para obter detalhes da API, consulte [DescribeFlowLogs](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2Host**

O código de exemplo a seguir mostra como usar Get-EC2Host.



## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna os detalhes do host EC2

```
Get-EC2Host
```

Saída:

```
AllocationTime      : 3/23/2019 4:55:22 PM
AutoPlacement       : off
AvailabilityZone     : eu-west-1b
AvailableCapacity   : Amazon.EC2.Model.AvailableCapacity
ClientToken         :
HostId              : h-01e23f4cd567890f1
HostProperties       : Amazon.EC2.Model.HostProperties
HostReservationId   :
Instances           : {}
ReleaseTime         : 1/1/0001 12:00:00 AM
State               : available
Tags                : {}
```

Exemplo 2: Este exemplo consulta o host AvailableInstanceCapacity h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Saída:

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11
```

- Para obter detalhes da API, consulte [DescribeHosts](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-EC2HostReservationOffering

O código de exemplo a seguir mostra como usar Get-EC2HostReservationOffering.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as reservas de host dedicado que estão disponíveis para compra para o determinado filtro 'instance-family', onde está PaymentOption " NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |  
Where-Object PaymentOption -eq NoUpfront
```

### Saída:

```
CurrencyCode      :  
Duration          : 94608000  
HourlyPrice       : 1.307  
InstanceFamily    : m4  
OfferingId        : hro-0c1f234567890d9ab  
PaymentOption     : NoUpfront  
UpfrontPrice      : 0.000  
  
CurrencyCode      :  
Duration          : 31536000  
HourlyPrice       : 1.830  
InstanceFamily    : m4  
OfferingId        : hro-04ad12aaaf34b5a67  
PaymentOption     : NoUpfront  
UpfrontPrice      : 0.000
```

- Para obter detalhes da API, consulte [DescribeHostReservationOfferings](#) sem Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2HostReservationPurchasePreview

O código de exemplo a seguir mostra como usar Get-EC2HostReservationPurchasePreview.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo mostra uma compra de reserva com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet  
h-01e23f4cd567890f1
```

**Saída:**

```

CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
-----
{}          1.307          0.000

```

- Para obter detalhes da API, consulte [GetHostReservationPurchasePreview](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2IdFormat**

O código de exemplo a seguir mostra como usar Get-EC2IdFormat.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve o formato de ID para o tipo de recurso especificado.

```
Get-EC2IdFormat -Resource instance
```

**Saída:**

```

Resource      UseLongIds
-----
instance      False

```

Exemplo 2: Este exemplo descreve os formatos de ID para todos os tipos de recursos que oferecem suporte a IDs mais longos.

```
Get-EC2IdFormat
```

**Saída:**

```

Resource      UseLongIds
-----
reservation   False
instance      False

```

- Para obter detalhes da API, consulte [DescribeIdFormat](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2IdentityIdFormat

O código de exemplo a seguir mostra como usar Get-EC2IdentityIdFormat.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna o formato de ID do recurso 'imagem' para a função fornecida

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
image
```

Saída:

```
Deadline           Resource UseLongIds
-----
8/2/2018 11:30:00 PM image      True
```

- Para obter detalhes da API, consulte [DescribeIdentityIdFormat](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Image

O código de exemplo a seguir mostra como usar Get-EC2Image.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve a AMI especificada.

```
Get-EC2Image -ImageId ami-12345678
```

Saída:

```
Architecture       : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate       : 2014-10-20T00:56:28.000Z
Description        : My image
Hypervisor         : xen
ImageId            : ami-12345678
ImageLocation      : 123456789012/my-image
ImageOwnerAlias    :
ImageType          : machine
```

```
KernelId      :  
Name          : my-image  
OwnerId       : 123456789012  
Platform      :  
ProductCodes  : {}  
Public        : False  
RamdiskId     :  
RootDeviceName : /dev/xvda  
RootDeviceType : ebs  
SriovNetSupport : simple  
State         : available  
StateReason    :  
Tags          : {Name}  
VirtualizationType : hvm
```

Exemplo 2: Este exemplo descreve as AMIs que você possui.

```
Get-EC2Image -owner self
```

Exemplo 3: Este exemplo descreve as AMIs públicas que executam o Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemplo 4: Este exemplo descreve todas as AMIs públicas na região 'us-west-2'.

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

- Para obter detalhes da API, consulte [Descrever Imagens](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ImageAttribute

O código de exemplo a seguir mostra como usar Get-EC2ImageAttribute.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo obtém a descrição da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Saída:

```
BlockDeviceMappings : {}
Description           : My image description
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      :
```

Exemplo 2: Esse exemplo obtém as permissões de execução para a AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Saída:

```
BlockDeviceMappings : {}
Description           :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {all}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      :
```

Exemplo 3: Este exemplo testa se a rede avançada está habilitada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Saída:

```
BlockDeviceMappings : {}
Description           :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      : simple
```

- Para obter detalhes da API, consulte [DescribeImageAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ImageByName

O código de exemplo a seguir mostra como usar Get-EC2ImageByName.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o conjunto completo de nomes de filtros que são atualmente suportados.

```
Get-EC2ImageByName
```

### Saída:

```
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
```

Exemplo 2: Esse exemplo descreve a AMI especificada. Usar esse comando para localizar uma AMI é útil porque AWS lança novas AMIs do Windows com as atualizações mais recentes a cada mês. Você pode especificar o 'Imageld' New-EC2Instance para iniciar uma instância usando a AMI atual para o filtro especificado.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

Saída:

```
Architecture      : x86_64
BlockDeviceMappings : {/dev/sda1, xvdca, xvdcb, xvdcc...}
CreationDate      : yyyy.mm.ddThh:mm:ss.000Z
Description       : Microsoft Windows Server 2016 with Desktop Experience Locale
                   English AMI provided by Amazon
Hypervisor        : xen
ImageId           : ami-xxxxxxxx
ImageLocation     : amazon/Windows_Server-2016-English-Full-Base-yyyy.mm.dd
ImageOwnerAlias   : amazon
ImageType        : machine
KernelId         :
Name              : Windows_Server-2016-English-Full-Base-yyyy.mm.dd
OwnerId          : 801119661308
Platform         : Windows
ProductCodes     : {}
Public           : True
RamdiskId        :
RootDeviceName   : /dev/sda1
RootDeviceType   : ebs
SriovNetSupport  : simple
State            : available
StateReason      :
Tags             : {}
```



```
VirtualizationType : hvm
```

- Para obter detalhes da API, consulte [Get-EC2ImageByName](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ImportImageTask

O código de exemplo a seguir mostra como usar Get-EC2ImportImageTask.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a tarefa de importação de imagem especificada.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Saída:

```
Architecture      : x86_64
Description       : Windows Image 2
Hypervisor        :
ImageId           : ami-1a2b3c4d
ImportTaskId      : import-ami-hgfedcba
LicenseType       : AWS
Platform          : Windows
Progress          :
SnapshotDetails   : {/dev/sda1}
Status            : completed
StatusMessage     :
```

Exemplo 2: Este exemplo descreve todas as suas tarefas de importação de imagens.

```
Get-EC2ImportImageTask
```

Saída:

```
Architecture      :
Description       : Windows Image 1
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
```

```

Platform      : Windows
Progress      :
SnapshotDetails : {}
Status        : deleted
StatusMessage  : User initiated task cancelation

Architecture  : x86_64
Description   : Windows Image 2
Hypervisor    :
ImageId       : ami-1a2b3c4d
ImportTaskId  : import-ami-hgfedcba
LicenseType   : AWS
Platform      : Windows
Progress      :
SnapshotDetails : {/dev/sda1}
Status        : completed
StatusMessage  :

```

- Para obter detalhes da API, consulte [DescribeImportImageTasks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ImportSnapshotTask

O código de exemplo a seguir mostra como usar Get-EC2ImportSnapshotTask.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a tarefa de importação de snapshot especificada.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Saída:

```

Description          ImportTaskId          SnapshotTaskDetail
-----
Disk Image Import 1  import-snap-abcdefgh
Amazon.EC2.Model.SnapshotTaskDetail

```

Exemplo 2: Este exemplo descreve todas as suas tarefas de importação de instantâneos.

```
Get-EC2ImportSnapshotTask
```

Saída:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2	import-snap-hgfedcba	Amazon.EC2.Model.SnapshotTaskDetail

- Para obter detalhes da API, consulte [DescribeImportSnapshotTask](#) sem Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Instance

O código de exemplo a seguir mostra como usar Get-EC2Instance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve a instância especificada.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Saída:

```
AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken         : T1eEy1448154045270
EbsOptimized        : False
Hypervisor          : xen
IamInstanceProfile  : Amazon.EC2.Model.IamInstanceProfile
ImageId             : ami-12345678
InstanceId          : i-12345678
InstanceLifecycle   :
InstanceType        : t2.micro
KernelId            :
KeyName             : my-key-pair
```

```
LaunchTime           : 12/4/2015 4:44:40 PM
Monitoring           : Amazon.EC2.Model.Monitoring
NetworkInterfaces    : {ip-10-0-2-172.us-west-2.compute.internal}
Placement            : Amazon.EC2.Model.Placement
Platform             : Windows
PrivateDnsName       : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress     : 10.0.2.172
ProductCodes         : {}
PublicDnsName        :
PublicIpAddress      :
RamdiskId            :
RootDeviceName       : /dev/sda1
RootDeviceType       : ebs
SecurityGroups       : {default}
SourceDestCheck      : True
SpotInstanceRequestId :
SriovNetSupport      :
State                : Amazon.EC2.Model.InstanceState
StateReason          :
StateTransitionReason :
SubnetId             : subnet-12345678
Tags                 : {Name}
VirtualizationType   : hvm
VpcId                : vpc-12345678
```

Exemplo 2: Este exemplo descreve todas as suas instâncias na região atual, agrupadas por reserva. Para ver os detalhes da instância, expanda a coleção de instâncias em cada objeto de reserva.

```
Get-EC2Instance
```

Saída:

```
GroupNames          : {}
Groups              : {}
Instances           : {}
OwnerId             : 123456789012
RequesterId        : 226008221399
ReservationId       : r-c5df370c

GroupNames          : {}
Groups              : {}
```

```
Instances      : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Exemplo 3: Este exemplo ilustra o uso de um filtro para consultar instâncias do EC2 em uma sub-rede específica de uma VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}).Instances
```

Saída:

```
InstanceId           InstanceType Platform PrivateIpAddress PublicIpAddress
SecurityGroups SubnetId           VpcId
-----
-----
i-01af...82cf180e19 t2.medium    Windows 10.0.0.98      ...
    subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows 10.0.0.53      ...
    subnet-1a2b3c4d vpc-1a2b3c4d
```

- Para obter detalhes da API, consulte [DescribeInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2InstanceAttribute

O código de exemplo a seguir mostra como usar `Get-EC2InstanceAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve o tipo de instância da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Saída:

```
InstanceType           : t2.micro
```

Exemplo 2: Este exemplo descreve se a rede avançada está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Saída:

```
SriovNetSupport           : simple
```

Exemplo 3: Este exemplo descreve os grupos de segurança da instância especificada.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Saída:

```
GroupId  
-----  
sg-12345678  
sg-45678901
```

Exemplo 4: Esse exemplo descreve se a otimização do EBS está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Saída:

```
EbsOptimized             : False
```

Exemplo 5: Esse exemplo descreve o atributo `disableApiTermination` da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Saída:

```
DisableApiTermination    : False
```

Exemplo 6: Esse exemplo descreve o atributo `instanceInitiatedShutdownComportamento` da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
instanceInitiatedShutdownBehavior
```

Saída:

```
InstanceInitiatedShutdownBehavior : stop
```

- Para obter detalhes da API, consulte [DescribeInstanceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2InstanceMetadata

O código de exemplo a seguir mostra como usar Get-EC2InstanceMetadata.

### Ferramentas para PowerShell

Exemplo 1: lista as categorias disponíveis de metadados de instância que podem ser consultados.

```
Get-EC2InstanceMetadata -ListCategory
```

Saída:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
```

```
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Exemplo 2: retorna o ID da Amazon Machine Image (AMI) que foi usada para iniciar a instância.

```
Get-EC2InstanceMetadata -Category AmiId
```

Saída:

```
ami-b2e756ca
```

Exemplo 3: Este exemplo consulta o documento de identidade formatado em JSON para a instância.

```
Get-EC2InstanceMetadata -Category IdentityDocument
{
  "availabilityZone" : "us-west-2a",
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : null,
  "version" : "2017-09-30",
  "instanceId" : "i-01ed50f7e2607f09e",
  "billingProducts" : [ "bp-6ba54002" ],
  "instanceType" : "t2.small",
  "pendingTime" : "2018-03-07T16:26:04Z",
  "imageId" : "ami-b2e756ca",
  "privateIp" : "10.0.0.171",
  "accountId" : "111122223333",
  "architecture" : "x86_64",
  "kernelId" : null,
  "ramdiskId" : null,
  "region" : "us-west-2"
}
```

Exemplo 4: Este exemplo usa uma consulta de caminho para obter os macs da interface de rede para a instância.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```



Saída:

```
02:80:7f:ef:4c:e0/
```

Exemplo 5: Se houver uma função do IAM associada à instância, retornará informações sobre a última vez em que o perfil da instância foi atualizado, incluindo a LastUpdated data da instância InstanceProfileArn, InstanceProfileId e.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Saída:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Para obter detalhes da API, consulte [Get-EC2InstanceMetadata](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2InstanceStatus

O código de exemplo a seguir mostra como usar Get-EC2InstanceStatus.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve o status da instância especificada.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Saída:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
```

```
SystemStatus      : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Saída:

```
Code    Name
----    -
16     running
```

```
$status.Status
```

Saída:

```
Details          Status
-----          -
{reachability}  ok
```

```
$status.SystemStatus
```

Saída:

```
Details          Status
-----          -
{reachability}  ok
```

- Para obter detalhes da API, consulte [DescribeInstanceStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2InternetGateway

O código de exemplo a seguir mostra como usar Get-EC2InternetGateway.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o gateway de Internet especificado.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

**Saída:**

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}

Exemplo 2: Este exemplo descreve todos os seus gateways de Internet.

```
Get-EC2InternetGateway
```

**Saída:**

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- Para obter detalhes da API, consulte [DescribeInternetGateways](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2KeyPair**

O código de exemplo a seguir mostra como usar `Get-EC2KeyPair`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve o par de chaves especificado.

```
Get-EC2KeyPair -KeyName my-key-pair
```

**Saída:**

KeyFingerprint	KeyName
-----	-----
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f	my-key-pair

Exemplo 2: Este exemplo descreve todos os seus pares de chaves.

```
Get-EC2KeyPair
```

- Para obter detalhes da API, consulte [DescribeKeyPairs](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2NetworkAcl

O código de exemplo a seguir mostra como usar Get-EC2NetworkAcl.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a rede ACL especificada.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault   : False
NetworkAclId : acl-12345678
Tags        : {Name}
VpcId       : vpc-12345678
```

Exemplo 2: Este exemplo descreve as regras para a rede ACL especificada.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Saída:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
```

```
Protocol      : -1
RuleAction    : deny
RuleNumber    : 32767
```

Exemplo 3: Este exemplo descreve todas as suas ACLs de rede.

```
Get-EC2NetworkAcl
```

- Para obter detalhes da API, consulte [DescribeNetworkAcls](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2NetworkInterface

O código de exemplo a seguir mostra como usar Get-EC2NetworkInterface.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Association      :
Attachment       : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone  : us-west-2c
Description      :
Groups           : {my-security-group}
MacAddress        : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId          : 123456789012
PrivateDnsName   : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId      :
RequesterManaged : False
SourceDestCheck  : True
Status           : in-use
SubnetId         : subnet-1a2b3c4d
TagSet           : {}
VpcId            : vpc-12345678
```

Exemplo 2: Este exemplo descreve todas as suas interfaces de rede.

```
Get-EC2NetworkInterface
```

- Para obter detalhes da API, consulte [DescribeNetworkInterfaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2NetworkInterfaceAttribute

O código de exemplo a seguir mostra como usar Get-EC2NetworkInterfaceAttribute.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

Saída:

```
Attachment           : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Exemplo 2: Este exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

Saída:

```
Description          : My description
```

Exemplo 3: Este exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

Saída:

```
Groups                : {my-security-group}
```

Exemplo 4: Este exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

Saída:

```
SourceDestCheck      : True
```

- Para obter detalhes da API, consulte [DescribeNetworkInterfaceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2PasswordData

O código de exemplo a seguir mostra como usar Get-EC2PasswordData.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descriptografa a senha que o Amazon EC2 atribuiu à conta do administrador para a instância especificada do Windows. Quando um arquivo pem foi especificado, a configuração da opção -Decrypt é automaticamente assumida.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 2: ( PowerShell somente para Windows) inspeciona a instância para determinar o nome do par de chaves usado para iniciar a instância e, em seguida, tenta encontrar os dados do par de chaves correspondente no repositório de configuração do AWS Toolkit for Visual Studio. Se os dados do par de chaves forem encontrados, a senha será descriptografada.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 3: retorna os dados da senha criptografada da instância.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Saída:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Para obter detalhes da API, consulte [GetPasswordData](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2PlacementGroup

O código de exemplo a seguir mostra como usar `Get-EC2PlacementGroup`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o grupo de posicionamento especificado.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- Para obter detalhes da API, consulte [DescribePlacementGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2PrefixList

O código de exemplo a seguir mostra como usar `Get-EC2PrefixList`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo busca o disponível Serviços da AWS em um formato de lista de prefixos para a região



```
Get-EC2PrefixList
```

Saída:

Cidrs	PrefixListId	PrefixListName
-----	-----	-----
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23}	p1-6fa54006	com.amazonaws.eu-west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}	p1-6da54004	com.amazonaws.eu-west-1.s3

- Para obter detalhes da API, consulte [DescribePrefixLists](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Region

O código de exemplo a seguir mostra como usar Get-EC2Region.

Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as regiões que estão disponíveis para você.

```
Get-EC2Region
```

Saída:

Endpoint	RegionName
-----	-----
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

- Para obter detalhes da API, consulte [DescribeRegions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2RouteTable

O código de exemplo a seguir mostra como usar Get-EC2RouteTable.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve todas as suas tabelas de rotas.

```
Get-EC2RouteTable
```

Saída:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateTable
State                   : active
VpcPeeringConnectionId :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId :
```

Exemplo 2: Este exemplo retorna detalhes da tabela de rotas especificada.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Exemplo 3: Este exemplo descreve as tabelas de rotas para a VPC especificada.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Saída:

```
Associations      : {rtbassoc-12345678}
PropagatingVgws  : {}
Routes           : {, }
RouteTableId     : rtb-1a2b3c4d
Tags             : {}
VpcId            : vpc-1a2b3c4d
```

- Para obter detalhes da API, consulte [DescribeRouteTables](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ScheduledInstance

O código de exemplo a seguir mostra como usar Get-EC2ScheduledInstance.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a instância agendada especificada.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

#### Saída:

```
AvailabilityZone      : us-west-2b
CreateDate            : 1/25/2016 1:43:38 PM
HourlyPrice           : 0.095
InstanceCount        : 1
InstanceType         : c4.large
NetworkPlatform      : EC2-VPC
NextSlotStartTime     : 1/31/2016 1:00:00 AM
Platform             : Linux/UNIX
PreviousSlotEndTime   :
Recurrence            : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId   : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours  : 32
TermEndDate           : 1/31/2017 1:00:00 AM
TermStartDate         : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Exemplo 2: Este exemplo descreve todas as suas instâncias programadas.

```
Get-EC2ScheduledInstance
```

- Para obter detalhes da API, consulte [DescribeScheduledInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2ScheduledInstanceAvailability

O código de exemplo a seguir mostra como usar Get-EC2ScheduledInstanceAvailability.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve uma programação que ocorre toda semana no domingo, começando na data especificada.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency  
Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -  
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -  
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

### Saída:

```
AvailabilityZone           : us-west-2b  
AvailableInstanceCount    : 20  
FirstSlotStartTime        : 1/31/2016 8:00:00 AM  
HourlyPrice               : 0.095  
InstanceType              : c4.large  
MaxTermDurationInDays    : 366  
MinTermDurationInDays    : 366  
NetworkPlatform          : EC2-VPC  
Platform                  : Linux/UNIX  
PurchaseToken             : eyJ2IjoiMSIsInMiOjEsImMiOi...  
Recurrence                : Amazon.EC2.Model.ScheduledInstanceRecurrence  
SlotDurationInHours      : 23  
TotalScheduledInstanceHours : 1219  
...
```

Exemplo 2: Para restringir os resultados, você pode adicionar filtros para critérios como sistema operacional, rede e tipo de instância.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-  
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Para obter detalhes da API, consulte [DescribeScheduledInstanceAvailability](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SecurityGroup

O código de exemplo a seguir mostra como usar Get-EC2SecurityGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o grupo de segurança especificado para uma VPC. Ao trabalhar com grupos de segurança pertencentes a uma VPC, você deve usar o ID do grupo de segurança (- GroupId parâmetro), não o nome (- GroupName parâmetro), para referenciar o grupo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

#### Saída:

```
Description      : default VPC security group
GroupId          : sg-12345678
GroupName        : default
IpPermissions    : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId         : 123456789012
Tags             : {}
VpcId           : vpc-12345678
```

Exemplo 2: Este exemplo descreve o grupo de segurança especificado para o EC2-Classic. Ao trabalhar com grupos de segurança para o EC2-Classic, você pode usar o nome do grupo (- GroupName parâmetro) ou o ID do grupo (- GroupId parâmetro) para fazer referência ao grupo de segurança.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

#### Saída:

```
Description      : my security group
GroupId          : sg-45678901
GroupName        : my-security-group
IpPermissions    : {Amazon.EC2.Model.IpPermission, Amazon.EC2.Model.IpPermission}
```

```
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Exemplo 3: Este exemplo recupera todos os grupos de segurança do vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Para obter detalhes da API, consulte [DescribeSecurityGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Snapshot

O código de exemplo a seguir mostra como usar Get-EC2Snapshot.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o instantâneo especificado.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

### Saída:

```
DataEncryptionKeyId :
Description          : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
                      vol-12345678
Encrypted            : False
KmsKeyId             :
OwnerAlias           :
OwnerId              : 123456789012
Progress             : 100%
SnapshotId           : snap-12345678
StartTime            : 10/23/2014 6:01:28 AM
State                 : completed
StateMessage         :
Tags                 : {}
VolumeId             : vol-12345678
VolumeSize           : 8
```

Exemplo 2: Este exemplo descreve os instantâneos que têm uma tag “Nome”.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Exemplo 3: Este exemplo descreve os instantâneos que têm uma tag 'Nome' com o valor 'TestValue'.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
  $_.Tags.Value -eq "TestValue" }
```

Exemplo 4: Este exemplo descreve todos os seus instantâneos.

```
Get-EC2Snapshot -Owner self
```

- Para obter detalhes da API, consulte [DescribeSnapshots](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SnapshotAttribute

O código de exemplo a seguir mostra como usar `Get-EC2SnapshotAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o atributo especificado do instantâneo especificado.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Saída:

CreateVolumePermissions	ProductCodes	SnapshotId
{}	{}	snap-12345678

Exemplo 2: Este exemplo descreve o atributo especificado do instantâneo especificado.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
  CreateVolumePermission).CreateVolumePermissions
```

Saída:

Group	UserId
-------	--------

```
-----  
-----  
all
```

- Para obter detalhes da API, consulte [DescribeSnapshotAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SpotDatafeedSubscription

O código de exemplo a seguir mostra como usar `Get-EC2SpotDatafeedSubscription`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve o feed de dados da sua instância spot.

```
Get-EC2SpotDatafeedSubscription
```

Saída:

```
Bucket   : my-s3-bucket  
Fault    :  
OwnerId  : 123456789012  
Prefix   : spotdata  
State    : Active
```

- Para obter detalhes da API, consulte [DescribeSpotDatafeedSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SpotFleetInstance

O código de exemplo a seguir mostra como usar `Get-EC2SpotFleetInstance`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as instâncias associadas à solicitação de frota spot especificada.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-  
aa30-494c-8788-1cee4EXAMPLE
```

Saída:



InstanceId	InstanceType	SpotInstanceRequestId
-----	-----	-----
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- Para obter detalhes da API, consulte [DescribeSpotFleetInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SpotFleetRequest

O código de exemplo a seguir mostra como usar Get-EC2SpotFleetRequest.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
| format-list
```

Saída:

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime           : 12/26/2015 8:23:33 AM
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Exemplo 2: Este exemplo descreve todas as suas solicitações de frota spot.

```
Get-EC2SpotFleetRequest
```

- Para obter detalhes da API, consulte [DescribeSpotFleetRequests](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SpotFleetRequestHistory

O código de exemplo a seguir mostra como usar Get-EC2SpotFleetRequestHistory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o histórico da solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

**Saída:**

```
HistoryRecords      : {Amazon.EC2.Model.HistoryRecord,
  Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

**Saída:**

EventInformation	EventType	Timestamp
-----	-----	-----
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:05 AM

- Para obter detalhes da API, consulte [DescribeSpotFleetRequestHistory](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2SpotInstanceRequest**

O código de exemplo a seguir mostra como usar Get-EC2SpotInstanceRequest.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve a solicitação de instância spot especificada.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

**Saída:**

```
ActualBlockHourlyPrice :  
AvailabilityZoneGroup  :  
BlockDurationMinutes  : 0  
CreateTime             : 4/8/2015 2:51:33 PM  
Fault                  :  
InstanceId             : i-12345678  
LaunchedAvailabilityZone : us-west-2b  
LaunchGroup           :  
LaunchSpecification    : Amazon.EC2.Model.LaunchSpecification  
ProductDescription     : Linux/UNIX  
SpotInstanceRequestId  : sir-12345678  
SpotPrice              : 0.020000  
State                  : active  
Status                 : Amazon.EC2.Model.SpotInstanceStatus  
Tags                   : {Name}  
Type                   : one-time
```

Exemplo 2: Este exemplo descreve todas as suas solicitações de instância spot.

```
Get-EC2SpotInstanceRequest
```

- Para obter detalhes da API, consulte [DescribeSpotInstanceRequests](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2SpotPriceHistory

O código de exemplo a seguir mostra como usar Get-EC2SpotPriceHistory.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo obtém as últimas 10 entradas no histórico de preços spot para o tipo de instância e a zona de disponibilidade especificados. Observe que o valor especificado para o AvailabilityZone parâmetro - deve ser válido para o valor da região fornecido ao parâmetro - Region do cmdlet (não mostrado no exemplo) ou definido como padrão no shell. Este exemplo de comando pressupõe que uma região padrão de 'us-west-2' tenha sido definida no ambiente.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -  
MaxResult 10
```

Saída:

```
AvailabilityZone : us-west-2a
InstanceType    : c3.large
Price           : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp       : 12/25/2015 7:39:49 AM

AvailabilityZone : us-west-2a
InstanceType    : c3.large
Price           : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp       : 12/25/2015 7:38:29 AM

AvailabilityZone : us-west-2a
InstanceType    : c3.large
Price           : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp       : 12/25/2015 6:57:13 AM
...
```

- Para obter detalhes da API, consulte [DescribeSpotPriceHistory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Subnet

O código de exemplo a seguir mostra como usar Get-EC2Subnet.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a sub-rede especificada.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz         : False
MapPublicIpOnLaunch  : False
State                 : available
SubnetId              : subnet-1a2b3c4d
```

```
Tags           : {}
VpcId          : vpc-12345678
```

Exemplo 2: Este exemplo descreve todas as suas sub-redes.

```
Get-EC2Subnet
```

- Para obter detalhes da API, consulte [DescribeSubnets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Tag

O código de exemplo a seguir mostra como usar Get-EC2Tag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca as tags para o tipo de recurso 'image'

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Saída:

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

Exemplo 2: Este exemplo busca todas as tags de todos os recursos e as agrupa por tipo de recurso

```
Get-EC2Tag | Group-Object resourcetype
```

Saída:

Count	Name	Group
-----	-----	-----
9	subnet	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}

```

53 instance           {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription...}
 3 route-table       {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
 5 security-group    {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription...}
30 volume            {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription...}
 1 internet-gateway  {Amazon.EC2.Model.TagDescription}
 3 network-interface {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
 4 elastic-ip        {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
 1 dhcp-options      {Amazon.EC2.Model.TagDescription}
 2 image              {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
 3 vpc                {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```

Exemplo 3: Este exemplo exibe todos os recursos com a tag 'autodelete' com o valor 'no' para a região em questão

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Saída:

Key	ResourceId	ResourceType	Value
---	-----	-----	----
auto-delete	i-0f1bce234d5dd678b	instance	no
auto-delete	vol-01d234aa5678901a2	volume	no
auto-delete	vol-01234bfb5def6f7b8	volume	no
auto-delete	vol-01ccb23f4c5e67890	volume	no

Exemplo 4: este exemplo obtém todos os recursos com a tag “exclusão automática” com valor “nenhum” e filtros adicionais no próximo canal para analisar somente os tipos de recursos de “instância” e, eventualmente, cria a tag “ThisInstance” para cada recurso da instância, com o valor sendo o próprio ID da instância

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"} |
Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -ResourceId
$_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Exemplo 5: Este exemplo busca tags para todos os recursos da instância, bem como para as chaves "Name", e as exibe em formato de tabela

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
@{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Saída:

```
ResourceId          Name-Tag
-----
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2Volume

O código de exemplo a seguir mostra como usar Get-EC2Volume.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o volume EBS especificado.

```
Get-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Attachments      : {}
AvailabilityZone  : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
```

```
Size           : 30
SnapshotId    : snap-12345678
State         : in-use
Tags          : {}
VolumeId      : vol-12345678
VolumeType    : standard
```

Exemplo 2: Este exemplo descreve seus volumes do EBS que têm o status 'disponível'.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Saída:

```
Attachments    : {}
AvailabilityZone : us-west-2c
CreateTime     : 12/21/2015 2:31:29 PM
Encrypted      : False
Iops           : 60
KmsKeyId       :
Size           : 20
SnapshotId    : snap-12345678
State         : available
Tags          : {}
VolumeId      : vol-12345678
VolumeType    : gp2
...
```

Exemplo 3: Este exemplo descreve todos os seus volumes do EBS.

```
Get-EC2Volume
```

- Para obter detalhes da API, consulte [DescribeVolumes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VolumeAttribute

O código de exemplo a seguir mostra como usar `Get-EC2VolumeAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o atributo especificado do volume especificado.



```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Saída:

```
AutoEnableIO    ProductCodes    VolumeId
-----
False           {}              vol-12345678
```

- Para obter detalhes da API, consulte [DescribeVolumeAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VolumeStatus

O código de exemplo a seguir mostra como usar Get-EC2VolumeStatus.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o status do volume especificado.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Saída:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Saída:

```
Details                Status
-----
{io-enabled, io-performance}  ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

**Saída:**

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- Para obter detalhes da API, consulte [DescribeVolumeStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-EC2Vpc**

O código de exemplo a seguir mostra como usar Get-EC2Vpc.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve a VPC especificada.

```
Get-EC2Vpc -VpcId vpc-12345678
```

**Saída:**

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

Exemplo 2: Este exemplo descreve a VPC padrão (só pode haver uma por região). Se sua conta oferecer suporte ao EC2-Classic nessa região, não há VPC padrão.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

**Saída:**

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
```

```
IsDefault      : True
State          : available
Tags           : {}
VpcId          : vpc-45678901
```

Exemplo 3: Este exemplo descreve as VPCs que correspondem ao filtro especificado (ou seja, têm um CIDR que corresponde ao valor '10.0.0.0/16' e estão no estado 'disponível').

```
Get-EC2Vpc -Filter @{{Name="cidr";
Values="10.0.0.0/16"}},@{{Name="state";Values="available"}}
```

Exemplo 4: Este exemplo descreve todas as suas VPCs.

```
Get-EC2Vpc
```

- Para obter detalhes da API, consulte [DescribeVpcs](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpcAttribute

O código de exemplo a seguir mostra como usar Get-EC2VpcAttribute.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o atributo enableDnsSupport ".

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Saída:

```
EnableDnsSupport
-----
True
```

Exemplo 2: Este exemplo descreve o atributo enableDnsHostnames ".

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Saída:

```
EnableDnsHostnames
-----
True
```

- Para obter detalhes da API, consulte [DescribeVpcAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpcClassicLink

O código de exemplo a seguir mostra como usar `Get-EC2VpcClassicLink`.

### Ferramentas para PowerShell

Exemplo 1: O exemplo acima retorna todas as VPCs com seu `ClassicLinkEnabled` estado para a região

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Saída:

```
ClassicLinkEnabled Tags    VpcId
-----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}      vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}      vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Para obter detalhes da API, consulte [DescribeVpcClassicLink](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpcClassicLinkDnsSupport

O código de exemplo a seguir mostra como usar `Get-EC2VpcClassicLinkDnsSupport`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o status de suporte de ClassicLink DNS das VPCs para a região eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Saída:

```
ClassicLinkDnsSupported VpcId
-----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Para obter detalhes da API, consulte [DescribeVpcClassicLinkDnsSupport](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpcEndpoint

O código de exemplo a seguir mostra como usar `Get-EC2VpcEndpoint`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve um ou mais dos seus VPC endpoints para a região eu-west-1. Em seguida, ele canaliza a saída para o próximo comando, que seleciona a `VpcEndpointId` propriedade e retorna o ID da VPC da matriz como matriz de string

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Saída:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Exemplo 2: Este exemplo descreve todos os endpoints vpc da região eu-west-1 e seleciona `VpcEndpointId`, e as propriedades para apresentá-los em formato `VpcId` tabular `ServiceName` `PrivateDnsEnabled`

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Saída:

```

VpcEndpointId      VpcId      ServiceName
PrivateDnsEnabled
-----
-----
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
    True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
    True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
    True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
    True

```

Exemplo 3: Este exemplo exporta o documento de política do VPC Endpoint vpce-01a2ab3f4f5cc6f7d em um arquivo json

```

Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json

```

- Para obter detalhes da API, consulte [DescribeVpcEndpoints](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpcEndpointService

O código de exemplo a seguir mostra como usar Get-EC2VpcEndpointService.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o serviço de endpoint EC2 VPC com o filtro fornecido, neste caso com.amazonaws.eu-west-1.ecs. Além disso, ele também expande a ServiceDetails propriedade e exibe os detalhes

```

Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails

```

Saída:

```

AcceptanceRequired      : False
AvailabilityZones       : {eu-west-1a, eu-west-1b, eu-west-1c}

```

```
BaseEndpointDnsNames      : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                      : amazon
PrivateDnsName            : ecs.eu-west-1.amazonaws.com
ServiceName               : com.amazonaws.eu-west-1.ecs
ServiceType               : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Exemplo 2: Este exemplo recupera todos os serviços do EC2 VPC Endpoint e retorna o “ssm” correspondente ServiceNames

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
  Servicenames | Where-Object { -match "ssm"}
```

Saída:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmessages
```

- Para obter detalhes da API, consulte [DescribeVpcEndpointServices](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpnConnection

O código de exemplo a seguir mostra como usar Get-EC2VpnConnection.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve a conexão VPN especificada.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     : Amazon.EC2.Model.VpnConnectionOptions
Routes                      : {Amazon.EC2.Model.VpnStaticRoute}
State                       : available
Tags                       : {}
Type                       : ipsec.1
```

```
VgwTelemetry           : {Amazon.EC2.Model.VgwTelemetry,  
  Amazon.EC2.Model.VgwTelemetry}  
VpnConnectionId       : vpn-12345678  
VpnGatewayId          : vgw-1a2b3c4d
```

Exemplo 2: Este exemplo descreve qualquer conexão VPN cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "pending", "available" )  
  
Get-EC2VpnConnection -Filter $filter
```

Exemplo 3: Este exemplo descreve todas as suas conexões VPN.

```
Get-EC2VpnConnection
```

- Para obter detalhes da API, consulte [DescribeVpnConnections](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EC2VpnGateway

O código de exemplo a seguir mostra como usar Get-EC2VpnGateway.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o gateway privado virtual especificado.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {vpc-12345678}  
VpnGatewayId    : vgw-1a2b3c4d
```



Exemplo 2: Este exemplo descreve qualquer gateway privado virtual cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Exemplo 3: Este exemplo descreve todos os seus gateways privados virtuais.

```
Get-EC2VpnGateway
```

- Para obter detalhes da API, consulte [DescribeVpnGateways](#) em Referência de AWS Tools for PowerShell cmdlet.

## Grant-EC2SecurityGroupEgress

O código de exemplo a seguir mostra como usar Grant-EC2SecurityGroupEgress.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo define uma regra de saída para o grupo de segurança especificado para EC2-VPC. A regra concede acesso ao intervalo de endereços IP especificado na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo concede acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte [AuthorizeSecurityGroupEgress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Grant-EC2SecurityGroupIngress

O código de exemplo a seguir mostra como usar Grant-EC2SecurityGroupIngress.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo define regras de entrada para um grupo de segurança para EC2-VPC. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). Observe que você deve identificar grupos de segurança para EC2-VPC usando o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")
```

```
$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 3: Este exemplo define regras de entrada para um grupo de segurança do EC2-Classic. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
  $ip2 )
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
  $ip2 )
```

Exemplo 5: Este exemplo concede acesso à porta TCP 8081 do grupo de segurança de origem especificado (sg-1a2b3c4d) ao grupo de segurança especificado (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
```

```
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Exemplo 6: Este exemplo adiciona o CIDR 5.5.5.5/32 às regras de entrada do grupo de segurança sg-1234abcd para tráfego da porta TCP 22 com uma descrição.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Para obter detalhes da API, consulte [AuthorizeSecurityGroupIngress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Import-EC2Image

O código de exemplo a seguir mostra como usar `Import-EC2Image`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo importa uma imagem de máquina virtual de disco único do bucket do Amazon S3 especificado para o Amazon EC2 com um token de idempotência. O exemplo exige que exista uma função de serviço de importação de VM com o nome padrão 'vmimport', com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico Pré-requisitos de importação da VM. Para usar um papel personalizado, especifique o nome do papel usando o **-RoleName** parâmetro.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "myVirtualMachineImages"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"
```

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

### Saída:

```
Architecture      :
Description       : Windows 2008 Standard Image
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
Progress          : 2
SnapshotDetails   : {}
Status            : active
StatusMessage     : pending
```

- Para obter detalhes da API, consulte [ImportImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Import-EC2KeyPair

O código de exemplo a seguir mostra como usar `Import-EC2KeyPair`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo importa uma chave pública para o EC2. A primeira linha armazena o conteúdo do arquivo de chave pública (\*.pub) na variável. **\$publickey** Em seguida, o exemplo converte o formato UTF8 do arquivo de chave pública em uma string codificada em Base64 e armazena a string convertida na variável. **\$pkbase64** Na última linha, a chave pública convertida é importada para o EC2. O cmdlet retorna a impressão digital e o nome da chave como resultados.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
```

```
$pkbase64 =
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Saída:

```
KeyFingerprint                                KeyName
-----
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Para obter detalhes da API, consulte [ImportKeyPair](#) em Referência de AWS Tools for PowerShell cmdlet.

## Import-EC2Snapshot

O código de exemplo a seguir mostra como usar `Import-EC2Snapshot`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo importa uma imagem de disco de VM no formato 'VMDK' para um snapshot do Amazon EBS. O exemplo requer uma função de serviço de importação de VM com o nome padrão 'vmimport', com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico em <http://docs.aws.amazon.com/2/latest/VM-VM-Import-Prerequisites.html>. **AWSEC WindowsGuide ImportPrerequisites** Para usar um papel personalizado, especifique o nome do papel usando o **-RoleName** parâmetro.

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "myVirtualMachineImages"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Saída:

```
Description                                ImportTaskId                                SnapshotTaskDetail
```

```
-----  
Disk Image Import      import-snap-abcdefgh  
Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes da API, consulte [ImportSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## Move-EC2AddressToVpc

O código de exemplo a seguir mostra como usar Move-EC2AddressToVpc.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo move uma instância do EC2 com um endereço IP público de 12.345.67.89 para a plataforma EC2-VPC na região Leste dos EUA (Norte da Virgínia).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Exemplo 2: Este exemplo canaliza os resultados de um Get-EC2Instance comando para o Move-EC2AddressToVpc cmdlet. O Get-EC2Instance comando obtém uma instância especificada pelo ID da instância e retorna a propriedade de endereço IP público da instância.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-  
EC2AddressToVpc
```

- Para obter detalhes da API, consulte [MoveAddressToVpc](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Address

O código de exemplo a seguir mostra como usar New-EC2Address.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo aloca um endereço IP elástico para usar com uma instância em uma VPC.

```
New-EC2Address -Domain Vpc
```

**Saída:**

AllocationId	Domain	PublicIp
-----	-----	-----
eipalloc-12345678	vpc	198.51.100.2

Exemplo 2: Este exemplo aloca um endereço IP elástico para usar com uma instância no EC2-Classic.

```
New-EC2Address
```

**Saída:**

AllocationId	Domain	PublicIp
-----	-----	-----
	standard	203.0.113.17

- Para obter detalhes da API, consulte [AllocateAddress](#) em Referência de AWS Tools for PowerShell cmdlet.

**New-EC2CustomerGateway**

O código de exemplo a seguir mostra como usar New-EC2CustomerGateway.

**Ferramentas para PowerShell**

Exemplo 1: Esse exemplo cria o gateway do cliente especificado.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

**Saída:**

```
BgpAsn           : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress        : 203.0.113.12
State            : available
Tags             : {}
Type             : ipsec.1
```



- Para obter detalhes da API, consulte [CreateCustomerGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2DhcpOption

O código de exemplo a seguir mostra como usar `New-EC2DhcpOption`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria o conjunto especificado de opções DHCP. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Saída:

DhcpConfigurations	DhcpOptionsId	Tags
-----	-----	----
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}

Exemplo 2: Com a PowerShell versão 2, você deve usar `New-Object` para criar cada opção DHCP.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Saída:

DhcpConfigurations	DhcpOptionsId	Tags
-----	-----	----
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}

- Para obter detalhes da API, consulte [CreateDhcpOptions](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2FlowLog

O código de exemplo a seguir mostra como usar New-EC2FlowLog.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um log de fluxo do EC2 para a sub-rede subnet-1d234567 até o cloud-watch-log nome 'subnet1-log' para todo o tráfego 'REJECT' usando as permissões da função 'Admin'

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Saída:

ClientToken	FlowLogIds	Unsuccessful
----- m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw=	----- {f1-012fc34eed5678c9d}	----- {}

- Para obter detalhes da API, consulte [CreateFlowLogs](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Host

O código de exemplo a seguir mostra como usar New-EC2Host.

### Ferramentas para PowerShell

Exemplo 1: este exemplo aloca um host dedicado à sua conta para o tipo de instância e a zona de disponibilidade específicos

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -Quantity 1
```

Saída:

```
h-01e23f4cd567890f3
```

- Para obter detalhes da API, consulte [AllocateHosts](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2HostReservation

O código de exemplo a seguir mostra como usar `New-EC2HostReservation`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo compra a oferta de reserva `hro-0c1f23456789d0ab` com configurações que correspondem às do seu host dedicado `h-01e23f4cd567890f1`

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet  
h-01e23f4cd567890f1
```

Saída:

```
ClientToken      :  
CurrencyCode     :  
Purchase         : {hr-0123f4b5d67bedc89}  
TotalHourlyPrice : 1.307  
TotalUpfrontPrice : 0.000
```

- Para obter detalhes da API, consulte [PurchaseHostReservation](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Image

O código de exemplo a seguir mostra como usar `New-EC2Image`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma AMI com o nome e a descrição especificados, a partir da instância especificada. O Amazon EC2 tenta desligar completamente a instância antes de criar a imagem e reinicia a instância após a conclusão.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web  
server AMI"
```

Exemplo 2: Esse exemplo cria uma AMI com o nome e a descrição especificados, a partir da instância especificada. O Amazon EC2 cria a imagem sem desligar e reiniciar a instância; portanto, a integridade do sistema de arquivos na imagem criada não pode ser garantida.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web server AMI" -NoReboot $true
```

Exemplo 3: Esse exemplo cria uma AMI com três volumes. O primeiro volume é baseado em um snapshot do Amazon EBS. O segundo volume é um volume vazio de 100 GiB do Amazon EBS. O terceiro volume é um volume de armazenamento de instâncias. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
  "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
  $ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
  sdc";VirtualName="ephemeral0"})
```

- Para obter detalhes da API, consulte [CreateImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Instance

O código de exemplo a seguir mostra como usar New-EC2Instance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo executa uma única instância da AMI especificada no EC2-Classic ou em uma VPC padrão.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Exemplo 2: Esse exemplo executa uma única instância da AMI especificada em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

Exemplo 3: Para adicionar um volume do EBS ou um volume de armazenamento de instâncias, defina um mapeamento de dispositivos de blocos e adicione-o ao comando. Este exemplo adiciona um volume de armazenamento de instâncias.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Exemplo 4: Para especificar uma das AMIs atuais do Windows, obtenha sua ID de AMI usando `Get-EC2ImageByName`. Este exemplo executa uma instância da AMI base atual para Windows Server 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Exemplo 5: executa uma instância no ambiente de host dedicado especificado.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID
h-1a2b3c4d5e6f1a2b3
```

Exemplo 6: Essa solicitação inicia duas instâncias e aplica uma tag com uma chave de servidor web e um valor de produção às instâncias. A solicitação também aplica uma tag com uma chave de centro de custos e um valor de cc123 aos volumes criados (nesse caso, o volume raiz de cada instância).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)
```

```
New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

- Para obter detalhes da API, consulte [RunInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2InstanceExportTask

O código de exemplo a seguir mostra como usar New-EC2InstanceExportTask.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exporta uma instância parada, **i-0800b00a00EXAMPLE**, como um disco rígido virtual (VHD) para o bucket do S3. **testbucket-export-instances-2019** O ambiente de destino é **Microsoft**, e o parâmetro region é adicionado porque a instância está na **us-east-1** região, enquanto a AWS região padrão do usuário não é us-east-1. Para obter o status da tarefa de exportação, copie o **ExportTaskId** valor dos resultados desse comando e execute **Get-EC2ExportTask -ExportTaskId export\_task\_ID\_from\_results**.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "testbucket-export-
instances-2019" -TargetEnvironment Microsoft -Region us-east-1
```

Saída:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Para obter detalhes da API, consulte [CreateInstanceExportTask](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2InternetGateway

O código de exemplo a seguir mostra como usar New-EC2InternetGateway.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um gateway de Internet.

```
New-EC2InternetGateway
```

Saída:

Attachments	InternetGatewayId	Tags
-----	-----	----
{}	igw-1a2b3c4d	{}

- Para obter detalhes da API, consulte [CreateInternetGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2KeyPair

O código de exemplo a seguir mostra como usar `New-EC2KeyPair`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um par de chaves e captura a chave privada RSA codificada por PEM em um arquivo com o nome especificado. Quando você estiver usando PowerShell, a codificação deve ser definida como `ascii` para gerar uma chave válida. Para obter mais informações, consulte Criar, exibir e excluir pares de chaves do Amazon EC2 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html>) no Guia do usuário da interface de linha de AWS comando.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Para obter detalhes da API, consulte [CreateKeyPair](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2NetworkACL

O código de exemplo a seguir mostra como usar `New-EC2NetworkACL`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma rede ACL para a VPC especificada.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Saída:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Para obter detalhes da API, consulte [CreateNetworkAcl](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2NetworkAclEntry

O código de exemplo a seguir mostra como usar New-EC2NetworkAclEntry.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma entrada para a rede ACL especificada. A regra permite tráfego de entrada de qualquer lugar (0.0.0.0/0) na porta UDP 53 (DNS) em qualquer sub-rede associada.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -RuleAction
allow
```

- Para obter detalhes da API, consulte [CreateNetworkAclEntry](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2NetworkInterface

O código de exemplo a seguir mostra como usar New-EC2NetworkInterface.



## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria a interface de rede especificada.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Saída:

```
Association      :
Attachment       :
AvailabilityZone  : us-west-2c
Description      : my network interface
Groups           : {my-security-group}
MacAddress       : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId          : 123456789012
PrivateDnsName   : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.17
PrivateIpAddresses : {}
RequesterId      :
RequesterManaged : False
SourceDestCheck  : True
Status           : pending
SubnetId         : subnet-1a2b3c4d
TagSet           : {}
VpcId            : vpc-12345678
```

- Para obter detalhes da API, consulte [CreateNetworkInterface](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2PlacementGroup

O código de exemplo a seguir mostra como usar New-EC2PlacementGroup.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um grupo de posicionamento com o nome especificado.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Para obter detalhes da API, consulte [CreatePlacementGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Route

O código de exemplo a seguir mostra como usar New-EC2Route.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria a rota especificada para a tabela de rotas especificada. A rota corresponde a todo o tráfego e o envia para o gateway de Internet especificado.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId igw-1a2b3c4d
```

Saída:

```
True
```

- Para obter detalhes da API, consulte [CreateRoute](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2RouteTable

O código de exemplo a seguir mostra como usar New-EC2RouteTable.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma tabela de rotas para a VPC especificada.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Saída:

```
Associations      : {}  
PropagatingVgws  : {}  
Routes           : {}  
RouteTableId     : rtb-1a2b3c4d  
Tags             : {}
```

```
VpcId : vpc-12345678
```

- Para obter detalhes da API, consulte [CreateRouteTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2ScheduledInstance

O código de exemplo a seguir mostra como usar `New-EC2ScheduledInstance`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo inicia a instância agendada especificada.

```
New-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012 -
InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678 `
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Para obter detalhes da API, consulte [RunScheduledInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2ScheduledInstancePurchase

O código de exemplo a seguir mostra como usar `New-EC2ScheduledInstancePurchase`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo compra uma instância programada.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiJEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Saída:

```
AvailabilityZone : us-west-2b
```

```
CreateDate           : 1/25/2016 1:43:38 PM
HourlyPrice          : 0.095
InstanceCount       : 1
InstanceType        : c4.large
NetworkPlatform     : EC2-VPC
NextSlotStartTime   : 1/31/2016 1:00:00 AM
Platform            : Linux/UNIX
PreviousSlotEndTime :
Recurrence          : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours : 32
TermEndDate         : 1/31/2017 1:00:00 AM
TermStartDate       : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Para obter detalhes da API, consulte [PurchaseScheduledInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2SecurityGroup

O código de exemplo a seguir mostra como usar New-EC2SecurityGroup.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um grupo de segurança para a VPC especificada.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

Saída:

```
sg-12345678
```

Exemplo 2: Este exemplo cria um grupo de segurança para o EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Saída:

```
sg-45678901
```

- Para obter detalhes da API, consulte [CreateSecurityGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Snapshot

O código de exemplo a seguir mostra como usar New-EC2Snapshot.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um instantâneo do volume especificado.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Saída:

```
DataEncryptionKeyId :  
Description          : This is a test  
Encrypted           : False  
KmsKeyId            :  
OwnerAlias          :  
OwnerId             : 123456789012  
Progress            :  
SnapshotId          : snap-12345678  
StartTime           : 12/22/2015 1:28:42 AM  
State               : pending  
StateMessage        :  
Tags                : {}  
VolumeId            : vol-12345678  
VolumeSize          : 20
```

- Para obter detalhes da API, consulte [CreateSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2SpotDatafeedSubscription

O código de exemplo a seguir mostra como usar New-EC2SpotDatafeedSubscription.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um feed de dados da instância spot.

```
New-EC2SpotDatafeedSubscription -Bucket my-s3-bucket -Prefix spotdata
```

#### Saída:

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- Para obter detalhes da API, consulte [CreateSpotDatafeedSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Subnet

O código de exemplo a seguir mostra como usar New-EC2Subnet.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma sub-rede com o CIDR especificado.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

#### Saída:

```
AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz         : False
MapPublicIpOnLaunch  : False
State                 : pending
SubnetId              : subnet-1a2b3c4d
Tag                   : {}
VpcId                 : vpc-12345678
```

- Para obter detalhes da API, consulte [CreateSubnet](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Tag

O código de exemplo a seguir mostra como usar New-EC2Tag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona uma única tag ao recurso especificado. A chave da tag é 'myTag' e o valor da tag é 'myTagValue'. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Exemplo 2: Este exemplo atualiza ou adiciona as tags especificadas ao recurso especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
@{ Key="test"; Value="anotherTagValue" } )
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Para obter detalhes da API, consulte [CreateTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Volume

O código de exemplo a seguir mostra como usar New-EC2Volume.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria o volume especificado.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Saída:

```
Attachments      : {}
AvailabilityZone  : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size            : 50
SnapshotId      :
State           : creating
Tags            : {}
VolumeId        : vol-12345678
VolumeType      : gp2
```

Exemplo 2: Esse exemplo de solicitação cria um volume e aplica uma tag com uma chave de pilha e um valor de produção.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Para obter detalhes da API, consulte [CreateVolume](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2Vpc

O código de exemplo a seguir mostra como usar New-EC2Vpc.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma VPC com o CIDR especificado. A Amazon VPC também cria o seguinte para a VPC: um conjunto de opções DHCP padrão, uma tabela de rotas principal e uma ACL de rede padrão.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Saída:



```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- Para obter detalhes da API, consulte [CreateVpcem](#) Referência de AWS Tools for PowerShell cmdlet.

## New-EC2VpcEndpoint

O código de exemplo a seguir mostra como usar `New-EC2VpcEndpoint`.

### Ferramentas para PowerShell

Exemplo 1: neste exemplo, crie um novo VPC Endpoint para o serviço `com.amazonaws.eu-west-1.s3` na VPC `vpc-0fc1ff23f45b678eb`

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

Saída:

```
ClientToken VpcEndpoint
-----
                Amazon.EC2.Model.VpcEndpoint
```

- Para obter detalhes da API, consulte [CreateVpcEndpoint](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2VpnConnection

O código de exemplo a seguir mostra como usar `New-EC2VpnConnection`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado. A saída inclui as informações de configuração que seu administrador de rede precisa, no formato XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d
```

### Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId           : cgw-1a2b3c4d
Options                     :
Routes                      : {}
State                       : pending
Tags                        : {}
Type                        :
VgwTelemetry                : {}
VpnConnectionId            : vpn-12345678
VpnGatewayId                : vgw-1a2b3c4d
```

Exemplo 2: Este exemplo cria a conexão VPN e captura a configuração em um arquivo com o nome especificado.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Exemplo 3: Este exemplo cria uma conexão VPN, com roteamento estático, entre o gateway privado virtual especificado e o gateway do cliente especificado.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Para obter detalhes da API, consulte [CreateVpnConnection](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2VpnConnectionRoute

O código de exemplo a seguir mostra como usar `New-EC2VpnConnectionRoute`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria a rota estática especificada para a conexão VPN especificada.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

- Para obter detalhes da API, consulte [CreateVpnConnectionRoute](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EC2VpnGateway

O código de exemplo a seguir mostra como usar New-EC2VpnGateway.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria o gateway privado virtual especificado.

```
New-EC2VpnGateway -Type ipsec.1
```

Saída:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte [CreateVpnGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-EC2Address

O código de exemplo a seguir mostra como usar Register-EC2Address.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo associa o endereço IP elástico especificado à instância especificada em uma VPC.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Saída:

```
eipassoc-12345678
```

Exemplo 2: Este exemplo associa o endereço IP elástico especificado à instância especificada no EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte [AssociateAddress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-EC2DhcpOption

O código de exemplo a seguir mostra como usar `Register-EC2DhcpOption`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo associa o conjunto de opções de DHCP especificado à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: Este exemplo associa as opções padrão de DHCP definidas à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte [AssociateDhcpOptions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-EC2Image

O código de exemplo a seguir mostra como usar `Register-EC2Image`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo registra uma AMI usando o arquivo de manifesto especificado no Amazon S3.

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/image.manifest.xml -  
Name my-web-server-ami
```

- Para obter detalhes da API, consulte [RegisterImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-EC2PrivateIpAddress

O código de exemplo a seguir mostra como usar Register-EC2PrivateIpAddress.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo atribui o endereço IP privado secundário especificado à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress  
10.0.0.82
```

Exemplo 2: Este exemplo cria dois endereços IP privados secundários e os atribui à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -  
SecondaryPrivateIpAddressCount 2
```

- Para obter detalhes da API, consulte [AssignPrivateIpAddresses](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-EC2RouteTable

O código de exemplo a seguir mostra como usar Register-EC2RouteTable.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo associa a tabela de rotas especificada à sub-rede especificada.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
rtbassoc-12345678
```

- Para obter detalhes da API, consulte [AssociateRouteTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Address

O código de exemplo a seguir mostra como usar `Remove-EC2Address`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo libera o endereço IP elástico especificado para instâncias em uma VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Exemplo 2: Este exemplo libera o endereço IP elástico especificado para instâncias no EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Para obter detalhes da API, consulte [ReleaseAddress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2CapacityReservation

O código de exemplo a seguir mostra como usar `Remove-EC2CapacityReservation`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela a reserva de capacidade `cr-0c1f2345db6f7cdba`

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation (CancelCapacityReservation)"
on target "cr-0c1f2345db6f7cdba".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
True
```

- Para obter detalhes da API, consulte [CancelCapacityReservation](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2CustomerGateway**

O código de exemplo a seguir mostra como usar `Remove-EC2CustomerGateway`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo exclui o gateway do cliente especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on Target
"cgw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteCustomerGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2DhcpOption**

O código de exemplo a seguir mostra como usar `Remove-EC2DhcpOption`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o conjunto de opções DHCP especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
"dopt-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteDhcpOptions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2FlowLog

O código de exemplo a seguir mostra como usar Remove-EC2FlowLog.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o FlowLogId fl-01a2b3456a789c01 fornecido

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
"fl-01a2b3456a789c01".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteFlowLogs](#) em Referência de AWS Tools for PowerShell cmdlet.



## Remove-EC2Host

O código de exemplo a seguir mostra como usar Remove-EC2Host.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo libera o ID de host fornecido h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
"h-0badafd1dcb2f3456".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

Successful                Unsuccessful
-----
{h-0badafd1dcb2f3456} {}
```

- Para obter detalhes da API, consulte [ReleaseHosts](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Instance

O código de exemplo a seguir mostra como usar Remove-EC2Instance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo encerra a instância especificada (a instância pode estar em execução ou no estado “interrompido”). O cmdlet solicitará a confirmação antes de continuar; use a opção -Force para suprimir a solicitação.

```
Remove-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                InstanceId                PreviousState
```

```
-----  
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes da API, consulte [TerminateInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2InternetGateway

O código de exemplo a seguir mostra como usar `Remove-EC2InternetGateway`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o gateway de Internet especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

### Saída:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on Target  
"igw-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- Para obter detalhes da API, consulte [DeleteInternetGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2KeyPair

O código de exemplo a seguir mostra como usar `Remove-EC2KeyPair`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o par de chaves especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteKeyPair](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2NetworkAcl**

O código de exemplo a seguir mostra como usar `Remove-EC2NetworkAcl`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo exclui a rede ACL especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
"acl-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteNetworkAcl](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2NetworkAclEntry**

O código de exemplo a seguir mostra como usar `Remove-EC2NetworkAclEntry`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a regra especificada da rede ACL especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on Target
"acl-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteNetworkAclEntry](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2NetworkInterface

O código de exemplo a seguir mostra como usar Remove-EC2NetworkInterface.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a interface de rede especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on Target
"eni-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteNetworkInterface](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2PlacementGroup

O código de exemplo a seguir mostra como usar Remove-EC2PlacementGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o grupo de posicionamento especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
"my-placement-group".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeletePlacementGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Route

O código de exemplo a seguir mostra como usar Remove-EC2Route.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a rota especificada da tabela de rotas especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteRoute](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2RouteTable**

O código de exemplo a seguir mostra como usar Remove-EC2RouteTable.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo exclui a tabela de rotas especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
"rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteRouteTable](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-EC2SecurityGroup**

O código de exemplo a seguir mostra como usar Remove-EC2SecurityGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o grupo de segurança especificado para EC2-VPC. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
"sg-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

Exemplo 2: Este exemplo exclui o grupo de segurança especificado para o EC2-Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Para obter detalhes da API, consulte [DeleteSecurityGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Snapshot

O código de exemplo a seguir mostra como usar Remove-EC2Snapshot.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o instantâneo especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
"snap-12345678".
```

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

- Para obter detalhes da API, consulte [DeleteSnapshot](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2SpotDatafeedSubscription

O código de exemplo a seguir mostra como usar `Remove-EC2SpotDatafeedSubscription`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo exclui o feed de dados da sua instância spot. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2SpotDatafeedSubscription
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
(DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteSpotDatafeedSubscription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Subnet

O código de exemplo a seguir mostra como usar `Remove-EC2Subnet`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a sub-rede especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```



## Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target "subnet-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteSubnet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Tag

O código de exemplo a seguir mostra como usar Remove-EC2Tag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Exemplo 2: Este exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Exemplo 3: Este exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Exemplo 4: Este exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Para obter detalhes da API, consulte [DeleteTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Volume

O código de exemplo a seguir mostra como usar Remove-EC2Volume.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo separa o volume especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target "vol-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- Para obter detalhes da API, consulte [DeleteVolume](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2Vpc

O código de exemplo a seguir mostra como usar Remove-EC2Vpc.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo exclui a VPC especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteVpc](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2VpnConnection

O código de exemplo a seguir mostra como usar `Remove-EC2VpnConnection`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a conexão VPN especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
"vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteVpnConnection](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2VpnConnectionRoute

O código de exemplo a seguir mostra como usar `Remove-EC2VpnConnectionRoute`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a rota estática especificada da conexão VPN especificada. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
Target "vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteVpnConnectionRoute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EC2VpnGateway

O código de exemplo a seguir mostra como usar Remove-EC2VpnGateway.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o gateway privado virtual especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
"vgw-1a2b3c4d".
```

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

- Para obter detalhes da API, consulte [DeleteVpnGateway](#) em Referência de AWS Tools for PowerShell cmdlet.

## Request-EC2SpotFleet

O código de exemplo a seguir mostra como usar Request-EC2SpotFleet.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma solicitação de frota spot na zona de disponibilidade com o menor preço para o tipo de instância especificado. Se sua conta suportar somente EC2-VPC, a frota spot executa as instâncias na zona de disponibilidade de menor preço que tem uma sub-rede padrão. Se sua conta suportar o EC2-Classic, a frota spot executa as instâncias no EC2-Classic na zona de disponibilidade de menor preço. Observe que o preço pago não excederá o preço spot especificado para a solicitação.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Para obter detalhes da API, consulte [RequestSpotFleet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Request-EC2SpotInstance

O código de exemplo a seguir mostra como usar Request-EC2SpotInstance.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo solicita uma instância spot única na sub-rede especificada. Observe que o grupo de segurança deve ser criado para a VPC que contém a sub-rede especificada e deve ser especificado por ID usando a interface de rede. Ao especificar uma interface de rede, você deve incluir a ID da sub-rede usando a interface de rede.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

### Saída:

```
ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 12/26/2015 7:44:10 AM
Fault                       :
InstanceId                  :
LaunchedAvailabilityZone    :
LaunchGroup                 :
LaunchSpecification         : Amazon.EC2.Model.LaunchSpecification
ProductDescription          : Linux/UNIX
SpotInstanceRequestId       : sir-12345678
SpotPrice                   : 0.050000
State                       : open
Status                      : Amazon.EC2.Model.SpotInstanceStatus
Tags                       : {}
Type                       : one-time
```

- Para obter detalhes da API, consulte [RequestSpotInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Reset-EC2ImageAttribute

O código de exemplo a seguir mostra como usar Reset-EC2ImageAttribute.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo redefine o atributo 'launchPermission' para seu valor padrão. Por padrão, as AMIs são privadas.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Para obter detalhes da API, consulte [ResetImageAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Reset-EC2InstanceAttribute

O código de exemplo a seguir mostra como usar Reset-EC2InstanceAttribute.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo redefine o atributo sriovNetSupport " para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Exemplo 2: Esse exemplo redefine o atributo 'ebsOptimized' para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Exemplo 3: Esse exemplo redefine o atributo sourceDestCheck " para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Exemplo 4: Esse exemplo redefine o atributo disableApiTermination " para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Exemplo 5: Esse exemplo redefine o atributo "instanceInitiatedShutdownComportamento" para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- Para obter detalhes da API, consulte [ResetInstanceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Reset-EC2NetworkInterfaceAttribute

O código de exemplo a seguir mostra como usar `Reset-EC2NetworkInterfaceAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo redefine a verificação de origem/destino para a interface de rede especificada.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Para obter detalhes da API, consulte [ResetNetworkInterfaceAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Reset-EC2SnapshotAttribute

O código de exemplo a seguir mostra como usar `Reset-EC2SnapshotAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo redefine o atributo especificado do instantâneo especificado.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- Para obter detalhes da API, consulte [ResetSnapshotAttribute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Restart-EC2Instance

O código de exemplo a seguir mostra como usar `Restart-EC2Instance`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo reinicializa a instância especificada.

```
Restart-EC2Instance -InstanceId i-12345678
```



- Para obter detalhes da API, consulte [RebootInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Revoke-EC2SecurityGroupEgress

O código de exemplo a seguir mostra como usar `Revoke-EC2SecurityGroupEgress`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a regra do grupo de segurança especificado para EC2-VPC. Isso revoga o acesso ao intervalo de endereços IP especificado na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar `New-Object` para criar o `IpPermission` objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 80  
$ip.ToPort = 80  
$ip.IpRanges.Add("203.0.113.0/24")  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo revoga o acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair  
$ug.GroupId = "sg-1a2b3c4d"  
$ug.UserId = "123456789012"  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission  
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte [RevokeSecurityGroupEgress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Revoke-EC2SecurityGroupIngress

O código de exemplo a seguir mostra como usar `Revoke-EC2SecurityGroupIngress`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para EC2-VPC. Observe que você deve identificar grupos de segurança para EC2-VPC usando o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar `New-Object` para criar o `IpPermission` objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 22  
$ip.ToPort = 22  
$ip.IpRanges.Add("203.0.113.0/24")  
  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para o EC2-Classic. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }  
  
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Exemplo 4: Com a PowerShell versão 2, você deve usar `New-Object` para criar o `IpPermission` objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 22
```

```
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Para obter detalhes da API, consulte [RevokeSecurityGroupIngress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Send-EC2InstanceStatus

O código de exemplo a seguir mostra como usar Send-EC2InstanceStatus.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo relata o feedback de status da instância especificada.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
unresponsive
```

- Para obter detalhes da API, consulte [ReportInstanceStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-EC2NetworkAclAssociation

O código de exemplo a seguir mostra como usar Set-EC2NetworkAclAssociation.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo associa a ACL de rede especificada à sub-rede da associação de ACL de rede especificada.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
aclassoc-1a2b3c4d
```

Saída:

```
aclassoc-87654321
```

- Para obter detalhes da API, consulte [ReplaceNetworkAclAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-EC2NetworkAclEntry

O código de exemplo a seguir mostra como usar Set-EC2NetworkAclEntry.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo substitui a entrada especificada para a rede ACL especificada. A nova regra permite tráfego de entrada do endereço especificado para qualquer sub-rede associada.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100  
-Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -  
RuleAction allow
```

- Para obter detalhes da API, consulte [ReplaceNetworkAclEntry](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-EC2Route

O código de exemplo a seguir mostra como usar Set-EC2Route.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo substitui a rota especificada pela tabela de rotas especificada. A nova rota envia o tráfego especificado para o gateway privado virtual especificado.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId  
vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte [ReplaceRoute](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-EC2RouteTableAssociation

O código de exemplo a seguir mostra como usar Set-EC2RouteTableAssociation.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo associa a tabela de rotas especificada à sub-rede para a associação da tabela de rotas especificada.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
rtbassoc-12345678
```

Saída:

```
rtbassoc-87654321
```

- Para obter detalhes da API, consulte [ReplaceRouteTableAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-EC2Instance

O código de exemplo a seguir mostra como usar Start-EC2Instance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo inicia a instância especificada.

```
Start-EC2Instance -InstanceId i-12345678
```

Saída:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

Exemplo 2: Este exemplo inicia as instâncias especificadas.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Exemplo 3: Este exemplo inicia o conjunto de instâncias que estão atualmente paradas. Os objetos Instance retornados por Get-EC2Instance são canalizados para Start-EC2Instance. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
| Start-EC2Instance
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Para obter detalhes da API, consulte [StartInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-EC2InstanceMonitoring

O código de exemplo a seguir mostra como usar Start-EC2InstanceMonitoring.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo permite o monitoramento detalhado da instância especificada.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId      Monitoring
-----
i-12345678     Amazon.EC2.Model.Monitoring
```

- Para obter detalhes da API, consulte [MonitorInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-EC2ImportTask

O código de exemplo a seguir mostra como usar Stop-EC2ImportTask.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela a tarefa de importação especificada (importação de instantâneo ou imagem). Se necessário, um motivo pode ser fornecido usando o -**CancelReason** parâmetro.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Para obter detalhes da API, consulte [CancelImportTask](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-EC2Instance

O código de exemplo a seguir mostra como usar Stop-EC2Instance.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo interrompe a instância especificada.

```
Stop-EC2Instance -InstanceId i-12345678
```

Saída:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- Para obter detalhes da API, consulte [StopInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-EC2InstanceMonitoring

O código de exemplo a seguir mostra como usar Stop-EC2InstanceMonitoring.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo desativa o monitoramento detalhado da instância especificada.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- Para obter detalhes da API, consulte [UnmonitorInstances](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-EC2SpotFleetRequest

O código de exemplo a seguir mostra como usar Stop-EC2SpotFleetRequest.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela a solicitação de frota spot especificada e encerra as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Exemplo 2: Este exemplo cancela a solicitação de frota spot especificada sem encerrar as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Para obter detalhes da API, consulte [CancelSpotFleetRequests](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-EC2SpotInstanceRequest

O código de exemplo a seguir mostra como usar Stop-EC2SpotInstanceRequest.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela a solicitação de instância spot especificada.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
SpotInstanceRequestId    State
-----
sir-12345678             cancelled
```

- Para obter detalhes da API, consulte [CancelSpotInstanceRequests](#) em Referência de AWS Tools for PowerShell cmdlet.



## Unregister-EC2Address

O código de exemplo a seguir mostra como usar `Unregister-EC2Address`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo dissocia o endereço IP elástico especificado da instância especificada em uma VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Exemplo 2: Este exemplo dissocia o endereço IP elástico especificado da instância especificada no EC2-Classic.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte [DisassociateAddress](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-EC2Image

O código de exemplo a seguir mostra como usar `Unregister-EC2Image`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela o registro da AMI especificada.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Para obter detalhes da API, consulte [DeregisterImage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-EC2PrivateIpAddress

O código de exemplo a seguir mostra como usar `Unregister-EC2PrivateIpAddress`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela a atribuição do endereço IP privado especificado da interface de rede especificada.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress 10.0.0.82
```

- Para obter detalhes da API, consulte [UnassignPrivateIpAddresses](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-EC2RouteTable

O código de exemplo a seguir mostra como usar `Unregister-EC2RouteTable`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a associação especificada entre uma tabela de rotas e uma sub-rede.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Para obter detalhes da API, consulte [DisassociateRouteTable](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon ECR usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon ECR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Get-ECRLoginCommand

O código de exemplo a seguir mostra como usar Get-ECRLoginCommand.

#### Ferramentas para PowerShell

Exemplo 1: Retorna um PSObject contendo informações de login que podem ser usadas para autenticação em qualquer registro do Amazon ECR ao qual seu diretor do IAM tenha acesso. As credenciais e o endpoint da região necessários para que a chamada obtenha o token de autorização são obtidos dos padrões do shell (configurados pelos cmdlets ou). **Set-AWSCredential/Set-DefaultAWSRegion Initialize-AWSDefaultConfiguration** Você pode usar a propriedade Command com Invoke-Expression para fazer login no registro especificado ou usar as credenciais retornadas em outras ferramentas que exigem login.

```
Get-ECRLoginCommand
```

#### Saída:

```
Username      : AWS
Password      : eyJwYX1sb2Fk...kRBVEffS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
               eyJwYX1sb2Fk...kRBVEffS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

Exemplo 2: recupera um PSObject contendo informações de login que você usa como entrada para um comando docker login. Você pode especificar qualquer URI de registro do Amazon ECR para autenticação, desde que seu diretor do IAM tenha acesso a esse registro.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- Para obter detalhes da API, consulte [Get-ECR LoginCommand](#) em AWS Tools for PowerShell Cmdlet Reference.

## Exemplos do Amazon ECS usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-ECSClusterDetail**

O código de exemplo a seguir mostra como usar `Get-ECSClusterDetail`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet descreve um ou mais dos seus clusters do ECS.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

### Saída:

```
LoggedAt      : 12/27/2019 9:27:41 PM
Clusters      : {LAB-ECS-CL}
Failures      : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength  : 396
HttpStatusCode : OK
```

- Para obter detalhes da API, consulte [DescribeClusters](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ECSClusterList

O código de exemplo a seguir mostra como usar `Get-ECSClusterList`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet retorna uma lista de clusters ECS existentes.

```
Get-ECSClusterList
```

Saída:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL  
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Para obter detalhes da API, consulte [ListClusters](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ECSClusterService

O código de exemplo a seguir mostra como usar `Get-ECSClusterService`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo lista todos os serviços em execução no seu cluster padrão.

```
Get-ECSClusterService
```

Exemplo 2: Este exemplo lista todos os serviços em execução no cluster especificado.

```
Get-ECSClusterService -Cluster myCluster
```

Exemplo 3: Este exemplo lista os serviços em execução no cluster especificado, buscando no máximo 10 detalhes do serviço por vez.

```
$nextToken = $null  
do  
{  
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
}
```

```
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [ListServices](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ECSService

O código de exemplo a seguir mostra como usar Get-ECSService.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo mostra como recuperar detalhes de um serviço específico do seu cluster padrão.

```
Get-ECSService -Service my-http-service
```

Exemplo 2: Esse exemplo mostra como recuperar detalhes de um serviço específico em execução no cluster nomeado.

```
Get-ECSService -Cluster myCluster -Service my-http-service
```

- Para obter detalhes da API, consulte [DescribeServices](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ECSCluster

O código de exemplo a seguir mostra como usar New-ECSCluster.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet cria um novo cluster do Amazon ECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{"Name"="containerInsights";  
Value="enabled"}
```

Saída:

```
ActiveServicesCount      : 0  
Attachments              : {}
```

```

AttachmentsStatus      :
CapacityProviders      : {}
ClusterArn             : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName           : LAB-ECS-CL
DefaultCapacityProviderStrategy : {}
PendingTasksCount     : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount     : 0
Settings               : {containerInsights}
Statistics             : {}
Status                 : ACTIVE
Tags                   : {}

```

- Para obter detalhes da API, consulte [CreateCluster](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ECSService

O código de exemplo a seguir mostra como usar `New-ECSService`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo de comando cria um serviço em seu cluster padrão chamado ``ecs-simple-service``. O serviço usa a definição de tarefa ``ecs-demo`` e mantém 10 instâncias dessa tarefa.

```

New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10

```

Exemplo 2: Esse exemplo de comando cria um serviço por trás de um balanceador de carga em seu cluster padrão chamado ``ecs-simple-service``. O serviço usa a definição de tarefa ``ecs-demo`` e mantém 10 instâncias dessa tarefa.

```

$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb

```

- Para obter detalhes da API, consulte [CreateService](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ECSCluster

O código de exemplo a seguir mostra como usar `Remove-ECSCluster`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet exclui o cluster ECS especificado. Você deve cancelar o registro de todas as instâncias de contêiner desse cluster antes de excluí-las.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Saída:

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

- Para obter detalhes da API, consulte [DeleteCluster](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ECSService

O código de exemplo a seguir mostra como usar `Remove-ECSService`.

### Ferramentas para PowerShell

Exemplo 1: exclui o serviço chamado 'my-http-service' no cluster padrão. O serviço deve ter uma contagem desejada e uma contagem contínua de 0 antes que você possa excluí-lo. Você será solicitado a confirmar antes que o comando continue. Para ignorar o prompt de confirmação, adicione a opção `-Force`.

```
Remove-ECSService -Service my-http-service
```

Exemplo 2: exclui o serviço chamado 'my-http-service' no cluster nomeado.



```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Para obter detalhes da API, consulte [DeleteService](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-ECSClusterSetting

O código de exemplo a seguir mostra como usar Update-ECSClusterSetting.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet modifica as configurações a serem usadas em um cluster ECS.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";  
Value="disabled"}
```

Saída:

```
ActiveServicesCount      : 0  
Attachments              : {}  
AttachmentsStatus       :  
CapacityProviders       : {}  
ClusterArn               : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-  
ECS-CL  
ClusterName              : LAB-ECS-CL  
DefaultCapacityProviderStrategy : {}  
PendingTasksCount       : 0  
RegisteredContainerInstancesCount : 0  
RunningTasksCount       : 0  
Settings                 : {containerInsights}  
Statistics                : {}  
Status                   : ACTIVE  
Tags                     : {}
```

- Para obter detalhes da API, consulte [UpdateClusterSettings](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-ECSService

O código de exemplo a seguir mostra como usar Update-ECSService.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo de comando atualiza o serviço my-http-service `` para usar a definição de tarefa amazon-ecs-sample ``.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Exemplo 2: Este exemplo de comando atualiza a contagem desejada do serviço my-http-service `` para 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Para obter detalhes da API, consulte [UpdateService](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon EFS usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon EFS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Edit-EFSMountTargetSecurityGroup**

O código de exemplo a seguir mostra como usar Edit-EFSMountTargetSecurityGroup.

## Ferramentas para PowerShell

Exemplo 1: atualiza os grupos de segurança em vigor para o destino de montagem especificado. Até 5 podem ser especificados, no formato “sg-xxxxxxx”.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- Para obter detalhes da API, consulte [ModifyMountTargetSecurityGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EFSFileSystem

O código de exemplo a seguir mostra como usar Get-EFSFileSystem.

## Ferramentas para PowerShell

Exemplo 1: retorna a coleção de todos os sistemas de arquivos pertencentes à conta do chamador na região.

```
Get-EFSFileSystem
```

Saída:

```
CreationTime      : 5/26/2015 4:02:38 PM
CreationToken     : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId      : fs-1a2b3c4d
LifeCycleState    : available
Name              :
NumberOfMountTargets : 0
OwnerId           : 123456789012
SizeInBytes       : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime      : 5/26/2015 4:06:23 PM
CreationToken     : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId      : fs-4d3c2b1a
...
```

Exemplo 2: Retorna os detalhes do sistema de arquivos especificado.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Exemplo 3: retorna os detalhes de um sistema de arquivos usando o token de criação de idempotência que foi especificado no momento em que o sistema de arquivos foi criado.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Para obter detalhes da API, consulte [DescribeFileSystems](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EFSMountTarget

O código de exemplo a seguir mostra como usar `Get-EFSMountTarget`.

### Ferramentas para PowerShell

Exemplo 1: retorna a coleção de destinos de montagem associados ao sistema de arquivos especificado.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Saída:

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifeCycleState    : available
MountTargetId     : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId           : 123456789012
SubnetId          : subnet-1a2b3c4d
```

- Para obter detalhes da API, consulte [DescribeMountTargets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EFSMountTargetSecurityGroup

O código de exemplo a seguir mostra como usar `Get-EFSMountTargetSecurityGroup`.

### Ferramentas para PowerShell

Exemplo 1: Retorna os IDs dos grupos de segurança atualmente atribuídos à interface de rede associada ao destino de montagem.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Saída:

```
sg-1a2b3c4d
```

- Para obter detalhes da API, consulte [DescribeMountTargetSecurityGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EFSTag

O código de exemplo a seguir mostra como usar Get-EFSTag.

Ferramentas para PowerShell

Exemplo 1: retorna a coleção de tags atualmente associadas ao sistema de arquivos especificado.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Saída:

```
Key          Value
---          -
Name         My File System
tagkey1      tagvalue1
tagkey2      tagvalue2
```

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EFSFileSystem

O código de exemplo a seguir mostra como usar New-EFSFileSystem.

Ferramentas para PowerShell

Exemplo 1: Cria um novo sistema de arquivos vazio. O token usado para garantir a criação de idempotentes será gerado automaticamente e poderá ser acessado a partir do **CreationToken** membro do objeto retornado.

```
New-EFSFileSystem
```

Saída:

```
CreationTime      : 5/26/2015 4:02:38 PM
CreationToken     : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId      : fs-1a2b3c4d
LifecycleState    : creating
Name              :
NumberOfMountTargets : 0
OwnerId           : 123456789012
SizeInBytes       : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Exemplo 2: cria um sistema de arquivos novo e vazio usando um token personalizado para garantir uma criação idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Para obter detalhes da API, consulte [CreateFileSystem](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EFSMountTarget

O código de exemplo a seguir mostra como usar `New-EFSMountTarget`.

### Ferramentas para PowerShell

Exemplo 1: Cria um novo destino de montagem para um sistema de arquivos. A sub-rede especificada será usada para determinar a Virtual Private Cloud (VPC) na qual o destino de montagem será criado e o endereço IP que será atribuído automaticamente (do intervalo de endereços da sub-rede). O endereço IP atribuído pode ser usado para montar esse sistema de arquivos em uma instância do Amazon EC2. Como nenhum grupo de segurança foi especificado, a interface de rede criada para o destino está associada ao grupo de segurança padrão para a VPC da sub-rede.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
FileSystemId      : fs-1a2b3c4d
IpAddress         : 10.0.0.131
LifeCycleState   : creating
MountTargetId    : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId          : 123456789012
SubnetId         : subnet-1a2b3c4d
```

Exemplo 2: Cria um novo destino de montagem para o sistema de arquivos especificado com endereço IP atribuído automaticamente. A interface de rede criada para o destino de montagem está associada aos grupos de segurança especificados (até 5, no formato “sg-xxxxxxx”, podem ser especificados).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Exemplo 3: Cria um novo destino de montagem para o sistema de arquivos especificado com o endereço IP especificado.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
10.0.0.131
```

- Para obter detalhes da API, consulte [CreateMountTarget](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EFSTag

O código de exemplo a seguir mostra como usar New-EFSTag.

### Ferramentas para PowerShell

Exemplo 1: Aplica a coleção de tags ao sistema de arquivos especificado. Se uma tag com a chave especificada já existir no sistema de arquivos, o valor da tag será atualizado.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
@{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Exemplo 2: Define a tag de nome para o sistema de arquivos especificado. Esse valor é retornado junto com outros detalhes do sistema de arquivos quando o FileSystem cmdlet Get-EFS é usado.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Para obter detalhes da API, consulte [CreateTag](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EFSFileSystem

O código de exemplo a seguir mostra como usar `Remove-EFSFileSystem`.

### Ferramentas para PowerShell

Exemplo 1: Exclui o sistema de arquivos especificado que não está mais em uso (se o sistema de arquivos tiver destinos de montagem, eles devem ser removidos primeiro). Você será solicitado a confirmar antes que o cmdlet continue. Para suprimir a confirmação, use a opção. **-Force**

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Para obter detalhes da API, consulte [DeleteFileSystem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EFSMountTarget

O código de exemplo a seguir mostra como usar `Remove-EFSMountTarget`.

### Ferramentas para PowerShell

Exemplo 1: Exclui o destino de montagem especificado. A confirmação será solicitada antes que a operação continue. Para suprimir o prompt, use o **-Force** switch. Observe que essa operação interrompe à força qualquer montagem do sistema de arquivos por meio do destino - talvez você queira considerar a desmontagem do sistema de arquivos antes de executar esse comando, se possível.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Para obter detalhes da API, consulte [DeleteMountTarget](#) em Referência de AWS Tools for PowerShell cmdlet.



## Remove-EFSTag

O código de exemplo a seguir mostra como usar Remove-EFSTag.

### Ferramentas para PowerShell

Exemplo 1: Exclui a coleção de uma ou mais tags de um sistema de arquivos. Você será solicitado a confirmar antes que o cmdlet continue. Para suprimir a confirmação, use a opção. - **Force**

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Para obter detalhes da API, consulte [DeleteTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon EKS usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon EKS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### Add-EKSResourceTag

O código de exemplo a seguir mostra como usar Add-EKSResourceTag.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet associa as tags especificadas a um recurso com o ResourceArn especificado.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- Para obter detalhes da API, consulte [TagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSCluster

O código de exemplo a seguir mostra como usar Get-EKSCluster.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um cluster do Amazon EKS.

```
Get-EKSCluster -Name "PROD"
```

Saída:

```
Arn                : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/25/2019 6:46:17 AM
Endpoint            : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity            : Amazon.EKS.Model.Identity
Logging             : Amazon.EKS.Model.Logging
Name                : PROD
PlatformVersion     : eks.7
ResourcesVpcConfig : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678912:role/eks-iam-role
Status              : ACTIVE
Tags                : {}
Version             : 1.14
```

- Para obter detalhes da API, consulte [DescribeCluster](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSClusterList

O código de exemplo a seguir mostra como usar `Get-EKSClusterList`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet lista os clusters do Amazon EKS Conta da AWS em sua região especificada.

```
Get-EKSClusterList
```

Saída:

```
PROD
```

- Para obter detalhes da API, consulte [ListClusters](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSFargateProfile

O código de exemplo a seguir mostra como usar `Get-EKSFargateProfile`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um perfil do AWS Fargate.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
ClusterName      : TEST
CreatedAt        : 12/26/2019 12:34:47 PM
FargateProfileArn : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors        : {Amazon.EKS.Model.FargateProfileSelector}
Status           : ACTIVE
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- Para obter detalhes da API, consulte [DescribeFargateProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSFargateProfileList

O código de exemplo a seguir mostra como usar `Get-EKSFargateProfileList`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet lista os perfis do AWS Fargate associados ao cluster especificado no seu Conta da AWS na região especificada.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Saída:

```
EKSFargate  
EKSFargateProfile
```

- Para obter detalhes da API, consulte [ListFargateProfiles](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSNodegroup

O código de exemplo a seguir mostra como usar `Get-EKSNodegroup`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um grupo de nós do Amazon EKS.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
AmiType       : AL2_x86_64  
ClusterName   : PROD  
CreatedAt     : 12/25/2019 10:16:45 AM  
DiskSize      : 40  
Health        : Amazon.EKS.Model.NodegroupHealth  
InstanceTypes : {t3.large}
```

```
Labels      : {}
ModifiedAt  : 12/25/2019 10:16:45 AM
NodegroupArn : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName : ProdEKSNodeGroup
NodeRole     : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess :
Resources    :
ScalingConfig : Amazon.EKS.Model.NodegroupScalingConfig
Status       : CREATING
Subnets     : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags         : {}
Version      : 1.14
```

- Para obter detalhes da API, consulte [DescribeNodegroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSNodegroupList

O código de exemplo a seguir mostra como usar `Get-EKSNodegroupList`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet lista os grupos de nós do Amazon EKS associados ao cluster especificado em você Conta da AWS na região especificada.

```
Get-EKSNodegroupList -ClusterName PROD
```

Saída:

```
ProdEKSNodeGroup
```

- Para obter detalhes da API, consulte [ListNodegroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSResourceTag

O código de exemplo a seguir mostra como usar `Get-EKSResourceTag`.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet lista as tags de um recurso do Amazon EKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Saída:

```
Key Value
---
Name EKSPRODCLUSTER
```

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSUpdate

O código de exemplo a seguir mostra como usar Get-EKSUpdate.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet retorna informações descritivas sobre uma atualização em seu cluster Amazon EKS ou grupo de nós gerenciados associado.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Para obter detalhes da API, consulte [DescribeUpdate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-EKSUpdateList

O código de exemplo a seguir mostra como usar `Get-EKSUpdateList`.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet lista as atualizações associadas a um cluster ou grupo de nós gerenciados do Amazon EKS em seu Conta da AWS, na região especificada.

```
Get-EKSUpdateList -Name "PROD"
```

Saída:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Para obter detalhes da API, consulte [ListUpdates](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EKSCluster

O código de exemplo a seguir mostra como usar `New-EKSCluster`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um novo cluster chamado 'prod'.

```
New-EKSCluster -Name prod -ResourcesVpcConfig  
@{SubnetIds=@("subnet-0a1b2c3d", "subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}  
-RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Saída:

```
Arn : arn:aws:eks:us-west-2:012345678901:cluster/prod  
CertificateAuthority : Amazon.EKS.Model.Certificate  
ClientRequestToken :  
CreatedAt : 12/10/2018 9:25:31 PM  
Endpoint :  
Name : prod  
PlatformVersion : eks.3  
ResourcesVpcConfig : Amazon.EKS.Model.VpcConfigResponse  
RoleArn : arn:aws:iam::012345678901:role/eks-service-role
```

```
Status          : CREATING
Version         : 1.10
```

- Para obter detalhes da API, consulte [CreateCluster](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EKSFargateProfile

O código de exemplo a seguir mostra como usar New-EKSFargateProfile.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet cria um perfil AWS Fargate para seu cluster Amazon EKS. Você deve ter pelo menos um perfil do Fargate em um cluster para poder programar pods na infraestrutura do Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
@{Namespace="default"}
```

### Saída:

```
ClusterName      : TEST
CreatedAt        : 12/26/2019 12:38:21 PM
FargateProfileArn : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors        : {Amazon.EKS.Model.FargateProfileSelector}
Status           : CREATING
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- Para obter detalhes da API, consulte [CreateFargateProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-EKSNodeGroup

O código de exemplo a seguir mostra como usar New-EKSNodeGroup.



## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet cria um grupo gerenciado de nós de trabalho para um cluster Amazon EKS. Você pode criar um grupo de nós para o cluster somente se for igual à versão atual de Kubernetes para o cluster. Todos os grupos de nós são criados com a versão mais recente da AMI para a respectiva versão secundária do Kubernetes do cluster.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
-DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
-NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
"subnet-0d1a9fff35efa7691", "subnet-0a3f4928edbc224d4"
```

### Saída:

```
AmiType       : AL2_x86_64
ClusterName   : PROD
CreatedAt     : 12/25/2019 10:16:45 AM
DiskSize      : 40
Health        : Amazon.EKS.Model.NodegroupHealth
InstanceTypes : {t3.large}
Labels        : {}
ModifiedAt    : 12/25/2019 10:16:45 AM
NodegroupArn  : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName : ProdEKSNodeGroup
NodeRole      : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess  :
Resources     :
ScalingConfig : Amazon.EKS.Model.NodegroupScalingConfig
Status        : CREATING
Subnets      : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags          : {}
Version       : 1.14
```

- Para obter detalhes da API, consulte [CreateNodegroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EKSCluster

O código de exemplo a seguir mostra como usar Remove-EKSCluster.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet exclui o plano de controle de cluster do Amazon EKS.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Saída:

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

Arn                : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken  :
CreatedAt           : 12/25/2019 9:33:25 AM
Endpoint            : https://02E6D31E3E4F8C15D7BE7F58D527776A.y14.us-west-2.eks.amazonaws.com
Identity            : Amazon.EKS.Model.Identity
Logging             : Amazon.EKS.Model.Logging
Name                : DEV-KUBE-CL
PlatformVersion     : eks.7
ResourcesVpcConfig : Amazon.EKS.Model.VpcConfigResponse
RoleArn             : arn:aws:iam::012345678912:role/eks-iam-role
Status              : DELETING
Tags                : {}
Version             : 1.14
```

- Para obter detalhes da API, consulte [DeleteCluster](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EKSFargateProfile

O código de exemplo a seguir mostra como usar `Remove-EKSFargateProfile`.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet exclui um perfil do AWS Fargate. Quando você exclui um perfil do Fargate, todos os pods em execução no Fargate que foram criados com o perfil são excluídos.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

### Saída:

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

ClusterName      : TEST
CreatedAt        : 12/26/2019 12:34:47 PM
FargateProfileArn : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors        : {Amazon.EKS.Model.FargateProfileSelector}
Status           : DELETING
Subnets         : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags             : {}
```

- Para obter detalhes da API, consulte [DeleteFargateProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EKSNodegroup

O código de exemplo a seguir mostra como usar Remove-EKSNodegroup.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet exclui um grupo de nós do Amazon EKS para um cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

### Saída:

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

```

Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
"ProdEKSNodeGroup".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

AmiType      : AL2_x86_64
ClusterName  : PROD
CreatedAt    : 12/25/2019 10:16:45 AM
DiskSize     : 40
Health       : Amazon.EKS.Model.NodegroupHealth
InstanceTypes : {t3.large}
Labels       : {}
ModifiedAt   : 12/25/2019 11:01:16 AM
NodegroupArn : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName : ProdEKSNodeGroup
NodeRole      : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess  :
Resources     : Amazon.EKS.Model.NodegroupResources
ScalingConfig : Amazon.EKS.Model.NodegroupScalingConfig
Status        : DELETING
Subnets      : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags          : {}
Version       : 1.14

```

- Para obter detalhes da API, consulte [DeleteNodegroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-EKSResourceTag

O código de exemplo a seguir mostra como usar Remove-EKSResourceTag.

### Ferramentas para PowerShell

Exemplo 1: Esse cmdlet exclui tags especificadas de um recurso EKS.

```

Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
-TagKey "Name"

```

Saída:

```

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
"arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

```

- Para obter detalhes da API, consulte [UntagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-EKSClusterConfig

O código de exemplo a seguir mostra como usar Update-EKSClusterConfig.

### Ferramentas para PowerShell

Exemplo 1: atualiza uma configuração de cluster do Amazon EKS. Seu cluster continua funcionando durante a atualização.

```

Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
@{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}

```

### Saída:

```

CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate

```

- Para obter detalhes da API, consulte [UpdateClusterConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-EKSClusterVersion

O código de exemplo a seguir mostra como usar Update-EKSClusterVersion.

## Ferramentas para PowerShell

Exemplo 1: Esse cmdlet atualiza um cluster Amazon EKS para a versão especificada do Kubernetes. Seu cluster continua funcionando durante a atualização.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

### Saída:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Para obter detalhes da API, consulte [UpdateClusterVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Elastic Load Balancing - Exemplos da versão 1 usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Elastic Load Balancing - Versão 1.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Add-ELBLoadBalancerToSubnet

O código de exemplo a seguir mostra como usar `Add-ELBLoadBalancerToSubnet`.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configurado para o balanceador de carga especificado. A saída inclui a lista completa de sub-redes.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

#### Saída:

```
subnet-12345678
subnet-87654321
```

- Para obter detalhes da API, consulte [AttachLoadBalancerToSubnets](#) em Referência de AWS Tools for PowerShell cmdlet.

### Add-ELBResourceTag

O código de exemplo a seguir mostra como usar `Add-ELBResourceTag`.

#### Ferramentas para PowerShell

Exemplo 1: Esse exemplo adiciona as tags especificadas ao balanceador de carga especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
@{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Exemplo 2: Com a PowerShell versão 2, você deve usar `New-Object` para criar uma tag para o parâmetro `Tag`.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
```

```
$tag.Value = "lima"  
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Para obter detalhes da API, consulte [AddTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-ELBAvailabilityZoneForLoadBalancer

O código de exemplo a seguir mostra como usar `Disable-ELBAvailabilityZoneForLoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a zona de disponibilidade especificada do balanceador de carga especificado. A saída inclui as zonas de disponibilidade restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -  
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2b
```

- Para obter detalhes da API, consulte [DisableAvailabilityZonesForLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Dismount-ELBLoadBalancerFromSubnet

O código de exemplo a seguir mostra como usar `Dismount-ELBLoadBalancerFromSubnet`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a sub-rede especificada do conjunto de sub-redes configurado para o balanceador de carga especificado. A saída inclui as sub-redes restantes.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet  
subnet-12345678
```

Saída:



```
subnet-87654321
```

- Para obter detalhes da API, consulte [DetachLoadBalancerFromSubnets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELBLoadBalancerAttribute

O código de exemplo a seguir mostra como usar `Edit-ELBLoadBalancerAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo habilita o balanceamento de carga entre zonas para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
CrossZoneLoadBalancing_Enabled $true
```

Exemplo 2: Este exemplo desativa a drenagem da conexão para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -  
ConnectionDraining_Enabled $false
```

Exemplo 3: Este exemplo habilita o registro de acesso para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `  
>> -AccessLog_Enabled $true `  
>> -AccessLog_S3BucketName my-logs-bucket `  
>> -AccessLog_S3BucketPrefix my-app/prod `  
>> -AccessLog_EmitInterval 60
```

- Para obter detalhes da API, consulte [ModifyLoadBalancerAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-ELBAvailabilityZoneForLoadBalancer

O código de exemplo a seguir mostra como usar `Enable-ELBAvailabilityZoneForLoadBalancer`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona a zona de disponibilidade especificada ao balanceador de carga especificado. O resultado inclui a lista completa de zonas de disponibilidade.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2a
us-west-2b
```

- Para obter detalhes da API, consulte [EnableAvailabilityZonesForLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBInstanceHealth

O código de exemplo a seguir mostra como usar Get-ELBInstanceHealth.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo descreve o estado das instâncias registradas com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Saída:

Description State	InstanceId	ReasonCode
----- -----	-----	-----
N/A InService	i-87654321	N/A
Instance has failed at lea... OutOfService	i-12345678	Instance

Exemplo 2: Esse exemplo descreve o estado da instância especificada registrada com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Exemplo 3: Esse exemplo exibe a descrição completa do estado da instância especificada.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance  
i-12345678).Description
```

Saída:

```
Instance has failed at least the UnhealthyThreshold number of health checks  
consecutively.
```

- Para obter detalhes da API, consulte [DescribeInstanceHealth](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBLoadBalancer

O código de exemplo a seguir mostra como usar Get-ELBLoadBalancer.

### Ferramentas para PowerShell

Exemplo 1: este exemplo lista os nomes dos seus balanceadores de carga.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Saída:

```
LoadBalancerName  
-----  
my-load-balancer  
my-other-load-balancer  
my-internal-load-balancer
```

Exemplo 2: Este exemplo descreve o balanceador de carga especificado.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
AvailabilityZones      : {us-west-2a, us-west-2b}
```

```
BackendServerDescriptions :
  {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime                : 4/11/2015 12:12:45 PM
DNSName                    : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
HealthCheck                : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                  : {i-207d9717, i-afefb49b}
ListenerDescriptions      : {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                   : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups             : {sg-a61988c3}
SourceSecurityGroup        : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                      : vpc-a01106c2
```

Exemplo 3: Este exemplo descreve todos os seus balanceadores de carga na AWS região atual.

```
Get-ELBLoadBalancer
```

Exemplo 4: Este exemplo descreve todos os seus balanceadores de carga em todos os disponíveis Regiões da AWS.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Para obter detalhes da API, consulte [DescribeLoadBalancers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBLoadBalancerAttribute

O código de exemplo a seguir mostra como usar Get-ELBLoadBalancerAttribute.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve os atributos do balanceador de carga especificado.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Saída:

```

AccessLog           : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes : {}
ConnectionDraining  : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings  : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing

```

- Para obter detalhes da API, consulte [DescribeLoadBalancerAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBLoadBalancerPolicy

O código de exemplo a seguir mostra como usar `Get-ELBLoadBalancerPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as políticas associadas ao balanceador de carga especificado.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Saída:

PolicyAttributeDescriptions	PolicyName
PolicyTypeName	
-----	-----
-----	
{ProxyProtocol}	my-ProxyProtocol-policy
ProxyProtocolPolicyType	
{CookieName}	my-app-cookie-policy
AppCookieStickinessPolicyType	

Exemplo 2: Este exemplo descreve os atributos da política especificada.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-ProxyProtocol-policy).PolicyAttributeDescriptions
```

Saída:

AttributeName	AttributeValue
---------------	----------------

```
-----
ProxyProtocol      true
```

Exemplo 3: Este exemplo descreve as políticas predefinidas, incluindo os exemplos de políticas. Os nomes das políticas de amostra têm o prefixo ELBSample-.

```
Get-ELBLoadBalancerPolicy
```

Saída:

```
PolicyAttributeDescriptions      PolicyName
PolicyTypeName
-----
-----
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
SSLNegotiationPolicyType
```

- Para obter detalhes da API, consulte [DescribeLoadBalancerPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBLoadBalancerPolicyType

O código de exemplo a seguir mostra como usar Get-ELBLoadBalancerPolicyType.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém os tipos de políticas compatíveis com o Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Saída:

```

Description                                PolicyAttributeTypeDescriptions
-----
-----
Stickiness policy with session lifet... {CookieExpirationPeriod}
  LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
  BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
  SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
  PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
  AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
  ProxyProtocolPolicyType

```

Exemplo 2: Este exemplo descreve o tipo de política especificado.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Saída:

```

Description                                PolicyAttributeTypeDescriptions
-----
-----
Policy that controls whether to incl... {ProxyProtocol}
  ProxyProtocolPolicyType

```

Exemplo 3: Este exemplo exibe a descrição completa do tipo de política especificado.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Saída:

Policy that controls whether to include the IP address and port of the originating request for TCP messages.

This policy operates on TCP/SSL listeners only

- Para obter detalhes da API, consulte [DescribeLoadBalancerPolicyTypes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELBResourceTag

O código de exemplo a seguir mostra como usar Get-ELBResourceTag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as tags dos balanceadores de carga especificados.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-balancer")
```

Saída:

LoadBalancerName	Tags
-----	----
my-load-balancer	{project, department}
my-internal-load-balancer	{project, department}

Exemplo 2: Este exemplo descreve as tags do balanceador de carga especificado.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Saída:

Key	Value
---	-----
project	lima
department	digital-media

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.



## Join-ELBSecurityGroupToLoadBalancer

O código de exemplo a seguir mostra como usar `Join-ELBSecurityGroupToLoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo substitui o grupo de segurança atual do balanceador de carga especificado pelo grupo de segurança especificado.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup sg-87654321
```

Saída:

```
sg-87654321
```

Exemplo 2: Para manter o grupo de segurança atual e especificar um grupo de segurança adicional, especifique os grupos de segurança existentes e os novos.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -  
SecurityGroup @("sg-12345678", "sg-87654321")
```

Saída:

```
sg-12345678  
sg-87654321
```

- Para obter detalhes da API, consulte [ApplySecurityGroupsToLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELBAppCookieStickinessPolicy

O código de exemplo a seguir mostra como usar `New-ELBAppCookieStickinessPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma política de aderência que segue a vida útil da sessão fixa do cookie especificado gerado pelo aplicativo.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-app-cookie-policy -CookieName my-app-cookie
```

- Para obter detalhes da API, consulte [CreateAppCookieStickinessPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELBLBCookieStickinessPolicy

O código de exemplo a seguir mostra como usar `New-ELBLBCookieStickinessPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma política de permanência com a vida útil da sessão fixa controlada pelo período de expiração especificado (em segundos).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy -CookieExpirationPeriod 60
```

Exemplo 2: Este exemplo cria uma política de aderência com tempos de vida de sessão fixos controlados pela vida útil do navegador (agente de usuário).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- Para obter detalhes da API, consulte [CreateLbCookieStickinessPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELBLoadBalancer

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um balanceador de carga com um ouvinte HTTP em uma VPC.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
```

```
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Exemplo 2: Este exemplo cria um balanceador de carga com um ouvinte HTTP no EC2-Classic.

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a -Listener $httpListener
```

Saída:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Exemplo 3: Este exemplo cria um balanceador de carga com um ouvinte HTTPS.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
-Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Para obter detalhes da API, consulte [CreateLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELBLoadBalancerListener

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancerListener`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona um ouvinte HTTPS ao balanceador de carga especificado.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
```

```
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
$httpsListener
```

- Para obter detalhes da API, consulte [CreateLoadBalancerListeners](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELBLoadBalancerPolicy

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancerPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova política de protocolo proxy para um balanceador de carga especificado.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -Property
@{
    AttributeName="ProxyProtocol"
    AttributeValue="True"
}
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
$attribute
```

- Para obter detalhes da API, consulte [CreateLoadBalancerPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-ELBInstanceWithLoadBalancer

O código de exemplo a seguir mostra como usar `Register-ELBInstanceWithLoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo registra a instância EC2 especificada com o balanceador de carga especificado.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance  
i-12345678
```

**Saída:**

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

- Para obter detalhes da API, consulte [RegisterInstancesWithLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-ELBInstanceFromLoadBalancer**

O código de exemplo a seguir mostra como usar `Remove-ELBInstanceFromLoadBalancer`.

**Ferramentas para PowerShell**

Exemplo 1: Esse exemplo remove a instância EC2 especificada do balanceador de carga especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance  
i-12345678
```

**Saída:**

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-ELBInstanceFromLoadBalancer  
(DeregisterInstancesFromLoadBalancer)" on Target  
"Amazon.ElasticLoadBalancing.Model.Instance".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):  
  
InstanceId  
-----  
i-87654321
```

- Para obter detalhes da API, consulte [DeregisterInstancesFromLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELBLoadBalancer

O código de exemplo a seguir mostra como usar Remove-ELBLoadBalancer.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o balanceador de carga especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

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

- Para obter detalhes da API, consulte [DeleteLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELBLoadBalancerListener

O código de exemplo a seguir mostra como usar Remove-ELBLoadBalancerListener.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o ouvinte na porta 80 do balanceador de carga especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort 80
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener (DeleteLoadBalancerListeners)"
on Target "80".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- Para obter detalhes da API, consulte [DeleteLoadBalancerListeners](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-ELBLoadBalancerPolicy**

O código de exemplo a seguir mostra como usar `Remove-ELBLoadBalancerPolicy`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo exclui a política especificada do balanceador de carga especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro `Force`.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

**Saída:**

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

- Para obter detalhes da API, consulte [DeleteLoadBalancerPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-ELBResourceTag**

O código de exemplo a seguir mostra como usar `Remove-ELBResourceTag`.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a tag especificada do balanceador de carga especificado. Você será solicitado a confirmar antes que a operação continue, a menos que você também especifique o parâmetro Force. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Saída:

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

Exemplo 2: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte [RemoveTag](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELBHealthCheck

O código de exemplo a seguir mostra como usar Set-ELBHealthCheck.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo define as configurações de verificação de integridade para o balanceador de carga especificado.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
```



```
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Saída:

```
HealthyThreshold : 2
Interval         : 30
Target           : HTTP:80/ping
Timeout          : 3
UnhealthyThreshold : 2
```

- Para obter detalhes da API, consulte [ConfigureHealthCheck](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELBLoadBalancerListenerSSLCertificate

O código de exemplo a seguir mostra como usar Set-ELBLoadBalancerListenerSSLCertificate.

Ferramentas para PowerShell

Exemplo 1: Esse exemplo substitui o certificado que encerra as conexões SSL para o ouvinte especificado.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Para obter detalhes da API, consulte [SetLoadBalancerListenerSslCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELBLoadBalancerPolicyForBackendServer

O código de exemplo a seguir mostra como usar Set-ELBLoadBalancerPolicyForBackendServer.

Ferramentas para PowerShell

Exemplo 1: Este exemplo substitui as políticas da porta especificada pela política especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Exemplo 2: Este exemplo remove todas as políticas associadas à porta especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -  
InstancePort 80
```

- Para obter detalhes da API, consulte [SetLoadBalancerPoliciesForBackendServer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELBLoadBalancerPolicyOfListener

O código de exemplo a seguir mostra como usar Set-ELBLoadBalancerPolicyOfListener.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo substitui as políticas do ouvinte especificado pela política especificada.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Exemplo 2: Este exemplo remove todas as políticas associadas ao ouvinte especificado.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -  
LoadBalancerPort 443
```

- Para obter detalhes da API, consulte [SetLoadBalancerPoliciesOfListener](#) em Referência de AWS Tools for PowerShell cmdlet.

## Elastic Load Balancing - Exemplos da versão 2 usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### Add-ELB2ListenerCertificate

O código de exemplo a seguir mostra como usar Add-ELB2ListenerCertificate.

Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona um certificado adicional ao Listener especificado.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618' -Certificate @{CertificateArn = 'arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97' }
```

Saída:

```
CertificateArn
IsDefault
-----
-----
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
False
```

- Para obter detalhes da API, consulte [AddListenerCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

### Add-ELB2Tag

O código de exemplo a seguir mostra como usar Add-ELB2Tag.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona um novo Tag ao **AWS.Tools.ElasticLoadBalancingV2** recurso especificado.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key = 'productVersion'; Value = '1.0.0'}
```

- Para obter detalhes da API, consulte [AddTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELB2Listener

O código de exemplo a seguir mostra como usar Edit-ELB2Listener.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica a ação padrão do ouvinte especificado para resposta fixa.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
        "ContentType" = "text/plain"
        "MessageBody" = "Hello World"
        "StatusCode" = "200"
    }
    "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port 8080 -DefaultAction $newDefaultAction
```

Saída:

```
Certificates      : {}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
```

```
Port           : 8080
Protocol      : HTTP
SslPolicy     :
```

- Para obter detalhes da API, consulte [ModifyListener](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELB2LoadBalancerAttribute

O código de exemplo a seguir mostra como usar `Edit-ELB2LoadBalancerAttribute`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo modifica os atributos do balanceador de carga especificado.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key = 'deletion_protection.enabled'; Value = 'true'}
```

Saída:

```
Key                                     Value
---                                     -
deletion_protection.enabled            true
access_logs.s3.enabled                 false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds           60
routing.http2.enabled                  true
routing.http.drop_invalid_header_fields.enabled false
```

- Para obter detalhes da API, consulte [ModifyLoadBalancerAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELB2Rule

O código de exemplo a seguir mostra como usar `Edit-ELB2Rule`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica as configurações especificadas da regra do Listener.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
        "Values" = "/login1","/login2","/login3"
    }
    "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

### Saída:

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 10
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte [ModifyRule](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELB2TargetGroup

O código de exemplo a seguir mostra como usar Edit-ELB2TargetGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica as propriedades do grupo-alvo especificado.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

### Saída:

```
HealthCheckEnabled      : True
HealthCheckIntervalSeconds : 60
HealthCheckPath         : /index.html
HealthCheckPort         : 8080
```

```

HealthCheckProtocol      : HTTP
HealthCheckTimeoutSeconds : 5
HealthyThresholdCount    : 5
LoadBalancerArns        : {}
Matcher                  : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                     : 80
Protocol                 : HTTP
TargetGroupArn           : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName          : test-tg
TargetType                : instance
UnhealthyThresholdCount  : 2
VpcId                    : vpc-2cfd7000

```

- Para obter detalhes da API, consulte [ModifyTargetGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-ELB2TargetGroupAttribute

O código de exemplo a seguir mostra como usar Edit-ELB2TargetGroupAttribute.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica o atributo `deregistration_delay` do grupo-alvo especificado.

```

Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
'deregistration_delay.timeout_seconds'; Value = 600}

```

Saída:

```

Key                               Value
---                               -
stickiness.enabled                 false
deregistration_delay.timeout_seconds 600
stickiness.type                    lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin

```

- Para obter detalhes da API, consulte [ModifyTargetGroupAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2AccountLimit

O código de exemplo a seguir mostra como usar `Get-ELB2AccountLimit`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando lista os limites da conta ELB2 para uma determinada região.

```
Get-ELB2AccountLimit
```

Saída:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Para obter detalhes da API, consulte [DescribeAccountLimits](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2Listener

O código de exemplo a seguir mostra como usar `Get-ELB2Listener`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve os ouvintes do ALB/NLB especificado.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```



**Saída:**

```

Certificates      : {}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port             : 80
Protocol         : HTTP
SslPolicy        :

Certificates      : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port             : 443
Protocol         : HTTPS
SslPolicy        : ELBSecurityPolicy-2016-08

```

- Para obter detalhes da API, consulte [DescribeListeners](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-ELB2ListenerCertificate**

O código de exemplo a seguir mostra como usar Get-ELB2ListenerCertificate.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo descreve o certificado para o ouvinte especificado.

```

Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b '

```

**Saída:**

```

CertificateArn
IsDefault
-----
-----

```

```
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
True
```

- Para obter detalhes da API, consulte [DescribeListenerCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2LoadBalancer

O código de exemplo a seguir mostra como usar Get-ELB2LoadBalancer.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo exibe todos os balanceadores de carga de uma determinada região.

```
Get-ELB2LoadBalancer
```

#### Saída:

```
AvailabilityZones      : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Para obter detalhes da API, consulte [DescribeLoadBalancers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2LoadBalancerAttribute

O código de exemplo a seguir mostra como usar Get-ELB2LoadBalancerAttribute.

## Ferramentas para PowerShell

Exemplo 1: Esse comando descreve os atributos de um determinado balanceador de carga.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Saída:

Key	Value
---	-----
access_logs.s3.enabled	false
load_balancing.cross_zone.enabled	true
access_logs.s3.prefix	
deletion_protection.enabled	false
access_logs.s3.bucket	

- Para obter detalhes da API, consulte [DescribeLoadBalancerAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2Rule

O código de exemplo a seguir mostra como usar Get-ELB2Rule.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve as regras do ouvinte para o ARN do ouvinte especificado.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 1
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79
```

```

Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 2
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {}
IsDefault    : True
Priority      : default
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf

```

- Para obter detalhes da API, consulte [DescribeRules](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2SSLPolicy

O código de exemplo a seguir mostra como usar Get-ELB2SSLPolicy.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todas as políticas de ouvinte disponíveis para a ElasticLoadBalancing V2.

```
Get-ELB2SSLPolicy
```

Saída:

```

Ciphers
-----
Name
----
SslProtocols
-----
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08      {TLSv1,
  TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01  {TLSv1.1,
  TLSv1.2}

```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06 {TLSv1,
TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05 {TLSv1,
TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04 {TLSv1,
TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08 {TLSv1.1,
TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08 {TLSv1.2}
```

- Para obter detalhes da API, consulte [DescribeSslPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2Tag

O código de exemplo a seguir mostra como usar Get-ELB2Tag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as tags do recurso especificado.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
ResourceArn
           Tags
-----
-----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2TargetGroup

O código de exemplo a seguir mostra como usar Get-ELB2TargetGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve o grupo-alvo especificado.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckEnabled      : True
HealthCheckIntervalSeconds : 30
HealthCheckPath         : /
HealthCheckPort         : traffic-port
HealthCheckProtocol     : HTTP
HealthCheckTimeoutSeconds : 5
HealthyThresholdCount   : 5
LoadBalancerArns       : {arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                 : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                    : 80
Protocol                : HTTP
TargetGroupArn          : arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName         : test-tg
TargetType              : instance
UnhealthyThresholdCount : 2
VpcId                   : vpc-2cfd7000
```

- Para obter detalhes da API, consulte [DescribeTargetGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2TargetGroupAttribute

O código de exemplo a seguir mostra como usar Get-ELB2TargetGroupAttribute.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo descreve os atributos do grupo-alvo especificado.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

Key	Value
---	-----
stickiness.enabled	false
deregistration_delay.timeout_seconds	300
stickiness.type	lb_cookie
stickiness.lb_cookie.duration_seconds	86400
slow_start.duration_seconds	0
load_balancing.algorithm.type	round_robin

- Para obter detalhes da API, consulte [DescribeTargetGroupAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ELB2TargetHealth

O código de exemplo a seguir mostra como usar Get-ELB2TargetHealth.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna o status de saúde dos alvos presentes no grupo-alvo especificado.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

HealthCheckPort	Target	TargetHealth
-----	-----	-----
80	Amazon.ElasticLoadBalancingV2.Model.TargetDescription	
	Amazon.ElasticLoadBalancingV2.Model.TargetHealth	

- Para obter detalhes da API, consulte [DescribeTargetHealth](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELB2Listener

O código de exemplo a seguir mostra como usar New-ELB2Listener.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo ouvinte do ALB com a ação padrão 'Encaminhar' para enviar tráfego para o grupo-alvo especificado.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    ForwardConfig = @{
        TargetGroups = @(
            @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
        )
        TargetGroupStickinessConfig = @{
            DurationSeconds = 900
            Enabled = $true
        }
    }
    Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -Protocol
"HTTP" -DefaultAction $defaultAction
```

### Saída:

```
Certificates      : {}
DefaultActions   : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn  : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port             : 8001
Protocol         : HTTP
SslPolicy        :
```



- Para obter detalhes da API, consulte [CreateListener](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELB2LoadBalancer

O código de exemplo a seguir mostra como usar New-ELB2LoadBalancer.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo balanceador de carga de aplicativos voltado para a Internet com duas sub-redes.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6', 'subnet-fc02eea0'
```

Saída:

```
AvailabilityZones      : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXD0TRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- Para obter detalhes da API, consulte [CreateLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELB2Rule

O código de exemplo a seguir mostra como usar New-ELB2Rule.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova regra de ouvinte com ação de resposta fixa com base no valor do cabeçalho do cliente para o ouvinte especificado.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
        "ContentType" = "text/plain"
        "MessageBody" = "Hello World"
        "StatusCode" = "200"
    }
    "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "httpHeaderConfig" = @{
        "HttpHeaderName" = "customHeader"
        "Values" = "header2","header1"
    }
    "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -Action
$newRuleAction -Condition $newRuleCondition -Priority 10
```

Saída:

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault    : False
Priority      : 10
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte [CreateRule](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ELB2TargetGroup

O código de exemplo a seguir mostra como usar New-ELB2TargetGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo grupo-alvo com os parâmetros fornecidos.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Saída:

```
HealthCheckEnabled      : True
HealthCheckIntervalSeconds : 30
HealthCheckPath         : /index.html
HealthCheckPort         : 80
HealthCheckProtocol     : HTTP
HealthCheckTimeoutSeconds : 5
HealthyThresholdCount   : 2
LoadBalancerArns       : {}
Matcher                 : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                    : 80
Protocol                : HTTP
TargetGroupArn          : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName        : NewTargetGroup
TargetType              : instance
UnhealthyThresholdCount : 5
VpcId                   : vpc-2cfd7000
```

- Para obter detalhes da API, consulte [CreateTargetGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-ELB2Target

O código de exemplo a seguir mostra como usar Register-ELB2Target.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo registra a instância 'i-0672a4c4cdeae3111' com o grupo de destino especificado.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id = 'i-0672a4c4cdeae3111'}
```

- Para obter detalhes da API, consulte [RegisterTargets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2Listener

O código de exemplo a seguir mostra como usar Remove-ELB2Listener.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o ouvinte especificado.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

Exemplo 2: Este exemplo remove o ouvinte especificado do balanceador de carga.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
```

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

- Para obter detalhes da API, consulte [DeleteListener](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2ListenerCertificate

O código de exemplo a seguir mostra como usar `Remove-ELB2ListenerCertificate`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o certificado especificado do grupo-alvo especificado.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
(RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): y
```

- Para obter detalhes da API, consulte [RemoveListenerCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2LoadBalancer

O código de exemplo a seguir mostra como usar `Remove-ELB2LoadBalancer`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o balanceador de carga especificado.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- Para obter detalhes da API, consulte [DeleteLoadBalancer](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2Rule

O código de exemplo a seguir mostra como usar Remove-ELB2Rule.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a regra especificada do Listener

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- Para obter detalhes da API, consulte [DeleteRule](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2Tag

O código de exemplo a seguir mostra como usar Remove-ELB2Tag.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a tag da chave especificada.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey 'productVersion'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
"arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): y
```

- Para obter detalhes da API, consulte [RemoveTag](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-ELB2TargetGroup

O código de exemplo a seguir mostra como usar Remove-ELB2TargetGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o grupo-alvo especificado.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Saída:

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

```
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
```

- Para obter detalhes da API, consulte [DeleteTargetGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELB2IpAddressType

O código de exemplo a seguir mostra como usar Set-ELB2IpAddressType.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo altera o tipo de endereço IP do balanceador de carga de 'IPv4' para "DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
dualstack
```

Saída:

```
Value
-----
dualstack
```

- Para obter detalhes da API, consulte [SetIpAddressType](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELB2RulePriority

O código de exemplo a seguir mostra como usar Set-ELB2RulePriority.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera a prioridade da regra de ouvinte especificada.



```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =  
'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-  
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Saída:

```
Actions      : {Amazon.ElasticLoadBalancingV2.Model.Action}  
Conditions   : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}  
IsDefault    : False  
Priority      : 11  
RuleArn      : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/  
test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Para obter detalhes da API, consulte [SetRulePriorities](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELB2SecurityGroup

O código de exemplo a seguir mostra como usar Set-ELB2SecurityGroup.

Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona o grupo de segurança 'sg-07c3414abb8811cbd' ao balanceador de carga especificado.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-  
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup  
'sg-07c3414abb8811cbd'
```

Saída:

```
sg-07c3414abb8811cbd
```

- Para obter detalhes da API, consulte [SetSecurityGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-ELB2Subnet

O código de exemplo a seguir mostra como usar Set-ELB2Subnet.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo modifica as sub-redes do balanceador de carga especificado.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
'subnet-7d8a0a51', 'subnet-c37a67a6'
```

Saída:

```
LoadBalancerAddresses SubnetId      ZoneName
-----
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Para obter detalhes da API, consulte [SetSubnets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-ELB2Target

O código de exemplo a seguir mostra como usar Unregister-ELB2Target.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela o registro da instância 'i-0672a4c4cdeae3111' do grupo de destino especificado.

```
$targetDescription = New-Object
Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Para obter detalhes da API, consulte [DeregisterTargets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon FSx usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon FSx.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-FSXResourceTag**

O código de exemplo a seguir mostra como usar `Add-FSXResourceTag`.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona tags ao recurso fornecido.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

Saída:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Para obter detalhes da API, consulte [TagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-FSXBackup

O código de exemplo a seguir mostra como usar Get-FSXBackup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca backups criados desde ontem para o ID do sistema de arquivos fornecido.

```
Get-FSXBackup -Filter @{"Name="file-system-id";Values=$fsx.FileSystemId} | Where-Object CreationTime -gt (Get-Date).AddDays(-1)
```

### Saída:

```
BackupId       : backup-01dac234e56782bcc
CreationTime   : 6/14/2019 3:35:14 AM
FailureDetails :
FileSystem     : Amazon.FSx.Model.FileSystem
KmsKeyId       : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f1-
e1234c5af123
Lifecycle      : AVAILABLE
ProgressPercent : 100
ResourceARN    : arn:aws:fsx:eu-west-1:123456789012:backup/backup-01dac234e56782bcc
Tags           : {}
Type           : AUTOMATIC
```

- Para obter detalhes da API, consulte [DescribeBackups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-FSXFileSystem

O código de exemplo a seguir mostra como usar Get-FSXFileSystem.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna a descrição de determinado FileSystemID.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

### Saída:

```
CreationTime   : 1/17/2019 9:55:30 AM
```

```

DNSName           : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails    :
FileSystemId      : fs-01cd23bc4bdf5678a
FileSystemType    : WINDOWS
KmsKeyId         : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-8bde-
a9f0-e1234c5af678
Lifecycle        : AVAILABLE
LustreConfiguration :
NetworkInterfaceIds : {eni-07d1dda1322b7e209}
OwnerId          : 123456789012
ResourceARN      : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a
StorageCapacity  : 300
SubnetIds        : {subnet-7d123456}
Tags             : {FSx-Service}
VpcId           : vpc-41cf2b3f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration

```

- Para obter detalhes da API, consulte [DescribeFileSystems](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-FSXResourceTagList

O código de exemplo a seguir mostra como usar Get-FSXResourceTagList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as tags do recurso arn fornecido.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Saída:

```

Key           Value
---           -
FSx-Service   Windows
Users         Dev

```

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-FSXBackup

O código de exemplo a seguir mostra como usar New-FSXBackup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um backup do sistema de arquivos fornecido.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

### Saída:

```
BackupId       : backup-0b1fac2345623456ba
CreationTime   : 6/14/2019 5:37:17 PM
FailureDetails :
FileSystem     : Amazon.FSx.Model.FileSystem
KmsKeyId       : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f3-
e1234c5af678
Lifecycle     : CREATING
ProgressPercent : 0
ResourceARN    : arn:aws:fsx:eu-west-1:123456789012:backup/
backup-0b1fac2345623456ba
Tags          : {}
Type          : USER_INITIATED
```

- Para obter detalhes da API, consulte [CreateBackup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-FSXFileSystem

O código de exemplo a seguir mostra como usar New-FSXFileSystem.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo sistema de arquivos Windows de 300 GB, permitindo acesso a partir da sub-rede especificada, que suporta taxa de transferência de até 8 megabytes por segundo. O novo sistema de arquivos é automaticamente associado ao Microsoft Active Directory especificado.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

### Saída:

```
CreationTime      : 12/10/2018 6:06:59 PM
DNSName           : fs-abcdef01234567890.example.com
FailureDetails    :
FileSystemId      : fs-abcdef01234567890
FileSystemType    : WINDOWS
KmsKeyId          : arn:aws:kms:us-west-2:123456789012:key/a1234567-252c-45e9-
afaa-123456789abc
Lifecycle         : CREATING
LustreConfiguration :
NetworkInterfaceIds : {}
OwnerId          : 123456789012
ResourceARN      : arn:aws:fsx:us-west-2:123456789012:file-system/fs-
abcdef01234567890
StorageCapacity  : 300
SubnetIds        : {subnet-1a2b3c4d5e6f}
Tags             : {}
VpcId           : vpc-1a2b3c4d5e6f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes da API, consulte [CreateFileSystem](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-FSXFileSystemFromBackup

O código de exemplo a seguir mostra como usar `New-FSXFileSystemFromBackup`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um novo sistema de arquivos Amazon FSx a partir de um backup existente do Amazon FSx for Windows File Server.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
@{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

**Saída:**

```
CreationTime      : 8/8/2019 12:59:58 PM
DNSName           : fs-012ff34e56789120.ktmsad.local
FailureDetails    :
FileSystemId      : fs-012ff34e56789120
FileSystemType    : WINDOWS
KmsKeyId          : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-1bde-
a2f3-e4567c8a9321
Lifecycle         : CREATING
LustreConfiguration :
NetworkInterfaceIds : {}
OwnerId          : 933303704102
ResourceARN       : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-012ff34e56789120
StorageCapacity   : 300
SubnetIds         : {subnet-fa1ae23c}
Tags              : {tag:Name}
VpcId             : vpc-12cf3b4f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes da API, consulte [CreateFileSystemFromBackup](#) em Referência de AWS Tools for PowerShell cmdlet.

**Remove-FSXBackup**

O código de exemplo a seguir mostra como usar Remove-FSXBackup.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo remove o ID de backup fornecido.

```
Remove-FSXBackup -BackupId $backupID
```

**Saída:**

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXBackup (DeleteBackup)" on target
"backup-0bbca1e2345678e12".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```



```
BackupId                Lifecycle
-----                -
backup-0bbca1e2345678e12 DELETED
```

- Para obter detalhes da API, consulte [DeleteBackup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-FSXFileSystem

O código de exemplo a seguir mostra como usar Remove-FSXFileSystem.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o ID do sistema de arquivos FSX fornecido.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXFileSystem (DeleteFileSystem)" on target
"fs-012ff34e567890120".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

FileSystemId                Lifecycle WindowsResponse
-----                -
fs-012ff34e567890120 DELETING Amazon.FSx.Model.DeleteFileSystemWindowsResponse
```

- Para obter detalhes da API, consulte [DeleteFileSystem](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-FSXResourceTag

O código de exemplo a seguir mostra como usar Remove-FSXResourceTag.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a tag de recurso do ARN do sistema de arquivos FSX fornecido.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXResourceTag (UntagResource)" on target
"arn:aws:fsx:eu-west-1:933303704102:file-system/fs-07cd45bc6bdf2674a".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [UntagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-FSXFileSystem

O código de exemplo a seguir mostra como usar Update-FSXFileSystem.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza os dias de retenção automática de backup do sistema de arquivos FSX. UpdateFileSystemWindowsConfiguration

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

Saída:

```
CreationTime      : 1/17/2019 9:55:30 AM
DNSName           : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails    :
FileSystemId      : fs-01cd23bc4bdf5678a
FileSystemType    : WINDOWS
```

```
KmsKeyId           : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-
a1f2-e1234c5af678
Lifecycle          : AVAILABLE
LustreConfiguration :
NetworkInterfaceIds : {eni-01cd23bc4bdf5678a}
OwnerId            : 933303704102
ResourceARN        : arn:aws:fsx:eu-west-1:933303704102:file-system/
fs-07cd45bc6bdf2674a
StorageCapacity    : 300
SubnetIds           : {subnet-1d234567}
Tags                : {FSx-Service}
VpcId               : vpc-23cf4b5f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes da API, consulte [UpdateFileSystem](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Glue exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Glue.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **New-GLUEJob**

O código de exemplo a seguir mostra como usar New-GLUEJob.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria uma nova tarefa no AWS Glue. O valor do nome do comando é sempre **glueet1**. O AWS Glue é compatível com a execução de scripts de tarefas escritos em Python ou Scala. Neste exemplo, o script de trabalho (MyTestGlueJob.py) é escrito em Python. Os parâmetros do Python são especificados na **\$DefArgs** variável e, em seguida, passados para o PowerShell comando no **DefaultArguments** parâmetro, que aceita uma tabela de hash. Os parâmetros na **\$JobParams** variável vêm da CreateJob API, documentados no tópico Jobs (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html>) da referência da API AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueet1'
$Command.ScriptLocation = 's3://aws-glue-scripts-000000000000-us-west-2/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
    '--TempDir' = 's3://aws-glue-temporary-000000000000-us-west-2/admin'
    '--job-bookmark-option' = 'job-bookmark-disable'
    '--job-language' = 'python'
}
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity" = "5"
    "Command" = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments" = $DefArgs
    "Description" = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries" = "1"
    "Name" = "MyOregonTestGlueJob"
    "Role" = "Amazon-GlueServiceRoleForSSM"
```

```
"Timeout"           = "20"  
}
```

```
New-GlueJob @JobParams
```

- Para obter detalhes da API, consulte [CreateJob](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Health exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Health.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### Get-HLTHEvent

O código de exemplo a seguir mostra como usar Get-HLTHEvent.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna eventos do AWS Personal Health Dashboard. O usuário adiciona o parâmetro -Region para ver os eventos disponíveis para o serviço na região Leste dos EUA (Norte da Virgínia), mas o parâmetro -Filter\_Region filtra os eventos registrados nas regiões UE (Londres) e Oeste dos EUA (Oregon) (eu-west-2 e us-west-2). O StartTime parâmetro -Filter\_ filtra por um intervalo de vezes em que os eventos podem começar, enquanto o EndTime parâmetro -Filter\_ filtra por um intervalo de vezes em que os eventos podem terminar. O resultado

é um evento de manutenção programada para o RDS que começa dentro do intervalo -Filter\_ especificado e termina dentro do StartTime intervalo programado -Filter\_. EndTime

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Saída:

```
Arn          : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone :
EndTime        : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode   : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime : 2/28/2019 2:26:07 PM
Region         : us-west-2
Service        : RDS
StartTime      : 3/14/2019 2:00:00 PM
StatusCode     : open
```

- Para obter detalhes da API, consulte [DescribeEvents](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de IAM usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Add-IAMClientIDToOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `Add-IAMClientIDToOpenIDConnectProvider`.

#### Ferramentas para PowerShell

Exemplo 1: este comando adiciona o ID do cliente (ou público) **my-application-ID** ao provedor OIDC existente denominado **server.example.com**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
-OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Para obter detalhes da API, consulte [AddClientIdToOpenIdConnectProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

### Add-IAMRoleTag

O código de exemplo a seguir mostra como usar `Add-IAMRoleTag`.

#### Ferramentas para PowerShell

Exemplo 1: este exemplo adiciona uma tag ao perfil no Identity Management Service

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte [TagRole](#) em Referência de AWS Tools for PowerShell cmdlet.

### Add-IAMRoleToInstanceProfile

O código de exemplo a seguir mostra como usar `Add-IAMRoleToInstanceProfile`.

#### Ferramentas para PowerShell

Exemplo 1: este comando adiciona o perfil denominado **S3Access** a um perfil de instância existente denominado **webserver**. Para criar o perfil de instância, use o comando **New-**

**IAMInstanceProfile**. Depois de criar o perfil de instância e associá-lo a um perfil usando esse comando, você pode anexá-lo a uma instância do EC2. Para isso, use o cmdlet **New-EC2Instance** com o parâmetro **InstanceProfile\_Arn** ou **InstanceProfile-Name** para executar a nova instância.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Para obter detalhes da API, consulte [AddRoleToInstanceProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-IAMUserTag

O código de exemplo a seguir mostra como usar Add-IAMUserTag.

### Ferramentas para PowerShell

Exemplo 1: este exemplo adiciona uma tag ao usuário no Identity Management Service

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte [TagUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Add-IAMUserToGroup

O código de exemplo a seguir mostra como usar Add-IAMUserToGroup.

### Ferramentas para PowerShell

Exemplo 1: este comando adiciona o usuário chamado **Bob** ao grupo denominado **Admins**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Para obter detalhes da API, consulte [AddUserToGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Disable-IAMMFADevice

O código de exemplo a seguir mostra como usar Disable-IAMMFADevice.



## Ferramentas para PowerShell

Exemplo 1: este comando desabilita o dispositivo de MFA de hardware associado ao usuário **Bob** que tem o número de série **123456789012**.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Exemplo 2: este comando desativa o dispositivo de MFA virtual associado ao usuário **David** que tem o ARN **arn:aws:iam::210987654321:mfa/David**. Observe que o dispositivo de MFA virtual não é excluído da conta. O dispositivo virtual ainda está presente e aparece na saída do comando **Get-IAMVirtualMFADevice**. Antes de criar um dispositivo de MFA virtual para o mesmo usuário, você deve excluir o antigo usando o comando **Remove-IAMVirtualMFADevice**.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/David"
```

- Para obter detalhes da API, consulte [DeactivateMfaDevice](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-IAMPassword

O código de exemplo a seguir mostra como usar `Edit-IAMPassword`.

## Ferramentas para PowerShell

Exemplo 1: este comando altera a senha do usuário que está executando o comando. Esse comando pode ser chamado somente por usuários do IAM. Se esse comando for chamado quando você estiver conectado com as credenciais da AWS conta (raiz), o comando retornará um erro. **InvalidUserType**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Para obter detalhes da API, consulte [ChangePassword](#) em Referência de AWS Tools for PowerShell cmdlet.

## Enable-IAMMFADevice

O código de exemplo a seguir mostra como usar `Enable-IAMMFADevice`.

## Ferramentas para PowerShell

Exemplo 1: este comando habilita o dispositivo de MFA de hardware com o número de série **987654321098** e associa o dispositivo ao usuário **Bob**. Ele inclui os dois primeiros códigos em sequência do dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -  
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Exemplo 2: este exemplo cria e habilita um dispositivo de MFA virtual. O primeiro comando cria o dispositivo virtual e retorna a representação de objeto do dispositivo na variável **\$MFADevice**. Você pode usar as propriedades **.Base32StringSeed** ou **QRCodePng** para configurar a aplicação de software do usuário. O comando final atribui o dispositivo ao usuário **David**, identificando o dispositivo pelo número de série. O comando também sincroniza o dispositivo com AWS a inclusão dos dois primeiros códigos em sequência do dispositivo de MFA virtual.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"  
# see example for New-IAMVirtualMFADevice to see how to configure the software  
program with PNG or base32 seed code  
Enable-IAMMFADevice -UserName "David" -SerialNumber $MFADevice.SerialNumber  
-SerialNumber $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2  
"13572468"
```

- Para obter detalhes da API, consulte [EnableMfaDevice](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAccessKey

O código de exemplo a seguir mostra como usar `Get-IAMAccessKey`.

## Ferramentas para PowerShell

Exemplo 1: este comando lista as chaves de acesso do usuário do IAM chamado **Bob**. Observe que não é possível listar as chaves de acesso secretas dos usuários do IAM. Se as chaves de acesso secretas forem perdidas, você deverá criar novas chaves de acesso com o cmdlet **New-IAMAccessKey**.

```
Get-IAMAccessKey -UserName "Bob"
```

**Saída:**

AccessKeyId	CreateDate	Status	UserName
-----	-----	-----	-----
AKIAIOSFODNN7EXAMPLE	12/3/2014 10:53:41 AM	Active	Bob
AKIAI44QH8DHBEXAMPLE	6/6/2013 8:42:26 PM	Inactive	Bob

- Para obter detalhes da API, consulte [ListAccessKeys](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-IAMAccessKeyLastUsed**

O código de exemplo a seguir mostra como usar `Get-IAMAccessKeyLastUsed`.

**Ferramentas para PowerShell**

Exemplo 1: retorna o nome de usuário proprietário e as informações do último uso da chave de acesso fornecida.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Para obter detalhes da API, consulte [GetAccessKeyLastUsed](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-IAMAccountAlias**

O código de exemplo a seguir mostra como usar `Get-IAMAccountAlias`.

**Ferramentas para PowerShell**

Exemplo 1: este comando retorna o alias da conta da Conta da AWS.

```
Get-IAMAccountAlias
```

**Saída:**

```
ExampleCo
```

- Para obter detalhes da API, consulte [ListAccountAliases](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAccountAuthorizationDetail

O código de exemplo a seguir mostra como usar `Get-IAMAccountAuthorizationDetail`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém detalhes de autorização sobre as identidades na AWS conta e exibe a lista de elementos do objeto retornado, incluindo usuários, grupos e funções. Por exemplo, a propriedade **UserDetailList** exibe detalhes sobre os usuários. Informações semelhantes estão disponíveis nas propriedades **RoleDetailList** e **GroupDetailList**.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

#### Saída:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

#### Saída:

```
Arn          : arn:aws:iam::123456789012:user/Administrator
CreateDate   : 10/16/2014 9:03:09 AM
GroupList    : {Administrators}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE1
UserName     : Administrator
UserPolicyList : {}

Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/6/2015 12:54:42 PM
GroupList    : {Developers}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE2
UserName     : bab
UserPolicyList : {}
```

```
Arn          : arn:aws:iam::123456789012:user/BackupToS3
CreateDate   : 1/27/2015 10:15:08 AM
GroupList    : {Backup}
Path         : /
UserId       : AIDACKCEVSQ6CEXAMPLE3
UserName     : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Para obter detalhes da API, consulte [GetAccountAuthorizationDetails](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAccountPasswordPolicy

O código de exemplo a seguir mostra como usar `Get-IAMAccountPasswordPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre a política de senha da conta atual. Se nenhuma política de senha estiver definida na conta, o comando retorna um erro **NoSuchEntity**.

```
Get-IAMAccountPasswordPolicy
```

### Saída:

```
AllowUsersToChangePassword : True
ExpirePasswords             : True
HardExpiry                  : False
MaxPasswordAge              : 90
MinimumPasswordLength       : 8
PasswordReusePrevention     : 20
RequireLowercaseCharacters  : True
RequireNumbers               : True
RequireSymbols               : False
RequireUppercaseCharacters  : True
```

- Para obter detalhes da API, consulte [GetAccountPasswordPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAccountSummary

O código de exemplo a seguir mostra como usar `Get-IAMAccountSummary`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna informações sobre o uso atual da entidade do IAM e das cotas atuais da entidade do IAM na Conta da AWS.

```
Get-IAMAccountSummary
```

Saída:

Key	Value
Users	7
GroupPolicySizeQuota	5120
PolicyVersionsInUseQuota	10000
ServerCertificatesQuota	20
AccountSigningCertificatesPresent	0
AccountAccessKeysPresent	0
Groups	3
UsersQuota	5000
RolePolicySizeQuota	10240
UserPolicySizeQuota	2048
GroupsPerUserQuota	10
AssumeRolePolicySizeQuota	2048
AttachedPoliciesPerGroupQuota	2
Roles	9
VersionsPerPolicyQuota	5
GroupsQuota	100
PolicySizeQuota	5120
Policies	5
RolesQuota	250
ServerCertificates	0
AttachedPoliciesPerRoleQuota	2
MFADevicesInUse	2
PoliciesQuota	1000
AccountMFAEnabled	1
Providers	2
InstanceProfilesQuota	100
MFADevices	4
AccessKeysPerUserQuota	2
AttachedPoliciesPerUserQuota	2
SigningCertificatesPerUserQuota	2
PolicyVersionsInUse	4
InstanceProfiles	1

...

- Para obter detalhes da API, consulte [GetAccountSummary](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAttachedGroupPolicyList

O código de exemplo a seguir mostra como usar `Get-IAMAttachedGroupPolicyList`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas que estão anexadas ao grupo do IAM nomeado **Admins** na AWS conta. Para ver a lista de políticas em linha incorporadas no grupo, use o comando **Get-IAMGroupPolicyList**.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Saída:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess	AdministratorAccess

- Para obter detalhes da API, consulte [ListAttachedGroupPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAttachedRolePolicyList

O código de exemplo a seguir mostra como usar `Get-IAMAttachedRolePolicyList`.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna os nomes e os ARNs das políticas gerenciadas anexadas ao perfil do IAM denominado **SecurityAuditRole** na conta da AWS . Para ver a lista de políticas em linha incorporadas no perfil, use o comando **Get-IAMRolePolicyList**.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Saída:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/SecurityAudit	SecurityAudit

- Para obter detalhes da API, consulte [ListAttachedRolePolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMAttachedUserPolicyList

O código de exemplo a seguir mostra como usar `Get-IAMAttachedUserPolicyList`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas para o usuário do IAM nomeado **Bob** na AWS conta. Para ver a lista de políticas em linha incorporadas no usuário do IAM, use o comando **Get-IAMUserPolicyList**.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Saída:

PolicyArn	PolicyName
-----	-----
arn:aws:iam::aws:policy/TesterPolicy	TesterPolicy

- Para obter detalhes da API, consulte [ListAttachedUserPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMContextKeysForCustomPolicy

O código de exemplo a seguir mostra como usar `Get-IAMContextKeysForCustomPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo busca todas as chaves de contexto presentes no JSON da política fornecida. A fim de produzir várias políticas, você pode fornecer uma lista de valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
```



```
west-2:123456789012:table/", "Condition": {"DateGreaterThan":
{"aws:CurrentTime": "2015-08-16T12:00:00Z"}}}]}'
$policy2 = '{"Version": "2012-10-17", "Statement":
{"Effect": "Allow", "Action": "dynamodb:*", "Resource": "arn:aws:dynamodb:us-
west-2:123456789012:table/"}}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Para obter detalhes da API, consulte [GetContextKeysForCustomPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMContextKeysForPrincipalPolicy

O código de exemplo a seguir mostra como usar `Get-IAMContextKeysForPrincipalPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo busca todas as chaves de contexto presentes no JSON da política fornecida e as políticas anexadas à entidade do IAM (usuário, perfil etc.). Para `-PolicyInputList` você pode fornecer uma lista de vários valores como valores separados por vírgula.

```
$policy1 = '{"Version": "2012-10-17", "Statement":
{"Effect": "Allow", "Action": "dynamodb:*", "Resource": "arn:aws:dynamodb:us-
west-2:123456789012:table/", "Condition": {"DateGreaterThan":
{"aws:CurrentTime": "2015-08-16T12:00:00Z"}}}]}'
$policy2 = '{"Version": "2012-10-17", "Statement":
{"Effect": "Allow", "Action": "dynamodb:*", "Resource": "arn:aws:dynamodb:us-
west-2:123456789012:table/"}}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Para obter detalhes da API, consulte [GetContextKeysForPrincipalPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMCredentialReport

O código de exemplo a seguir mostra como usar `Get-IAMCredentialReport`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo abre o relatório retornado e o envia ao pipeline como uma matriz de linhas de texto. A primeira linha é o cabeçalho com nomes de colunas separados por

vírgula. Cada linha sucessiva é a linha de detalhes de um usuário, com cada campo separado por vírgulas. Antes de visualizar o relatório, você deve gerá-lo com o cmdlet **Request-IAMCredentialReport**. Para recuperar o relatório como uma única string, use **-Raw** em vez de **-AsTextArray**. O alias **-SplitLines** também é aceito no switch **-AsTextArray**. Para obter a lista completa de colunas na saída, consulte a referência de API do serviço. Observe que, se não usar **-AsTextArray** ou **-SplitLines**, você deve extrair o texto da propriedade **.Content** usando a classe **StreamReader** .NET.

```
Request-IAMCredentialReport
```

Saída:

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

```
Get-IAMCredentialReport -AsTextArray
```

Saída:

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,password
root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04-20T16:06:00+00:00,A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+00:00,A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/A,A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Para obter detalhes da API, consulte [GetCredentialReport](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMEntitiesForPolicy

O código de exemplo a seguir mostra como usar `Get-IAMEntitiesForPolicy`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna uma lista de grupos, perfis e usuários do IAM que têm a política **arn:aws:iam::123456789012:policy/TestPolicy** anexada.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Saída:

```
IsTruncated   : False
Marker        :
PolicyGroups  : {}
PolicyRoles   : {testRole}
PolicyUsers   : {Bob, Theresa}
```

- Para obter detalhes da API, consulte [ListEntitiesForPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMGroup

O código de exemplo a seguir mostra como usar Get-IAMGroup.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre o grupo do IAM **Testers**, incluindo uma compilação de todos os usuários do IAM que pertencem ao grupo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Saída:

Group	IsTruncated	Marker
Users		
-----	-----	-----
-----		
Amazon.IdentityManagement.Model.Group	False	
{Theresa, David}		

```
$results.Group
```

**Saída:**

```
Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId  : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName : Testers
Path     : /
```

```
$results.Users
```

**Saída:**

```
Arn      : arn:aws:iam::123456789012:user/Theresa
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path     : /
UserId   : 40SVDDJJTF4XEEXAMPLE2
UserName : Theresa

Arn      : arn:aws:iam::123456789012:user/David
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path     : /
UserId   : Y4FKWQCXTA52QEXAMPLE3
UserName : David
```

- Para obter detalhes da API, consulte [GetGroupem](#) Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMGroupForUser

O código de exemplo a seguir mostra como usar `Get-IAMGroupForUser`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna a lista de grupos do IAM aos quais o usuário do IAM **David** pertence.

```
Get-IAMGroupForUser -UserName David
```

**Saída:**

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEXAMPLE1
GroupName : Administrators
Path      : /

Arn      : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId   : RHNZZGQJ7QHMAEXAMPLE2
GroupName : Testers
Path      : /

Arn      : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId   : ZU2E0WMK6WBZ0EXAMPLE3
GroupName : Developers
Path      : /
```

- Para obter detalhes da API, consulte [ListGroupsForUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMGroupList

O código de exemplo a seguir mostra como usar Get-IAMGroupList.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna uma coleção de todos os grupos do IAM definidos no atual Conta da AWS.

```
Get-IAMGroupList
```

Saída:

```
Arn      : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId   : 6WCH4TRY3KIHIEXAMPLE1
GroupName : Administrators
Path      : /

Arn      : arn:aws:iam::123456789012:group/Developers
```

```

CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2E0WMK6WBZ0EXAMPLE2
GroupName  : Developers
Path      : /

Arn       : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path      : /

```

- Para obter detalhes da API, consulte [ListGroupsem](#) Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMGroupPolicy

O código de exemplo a seguir mostra como usar Get-IAMGroupPolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre a política em linha incorporada denominada **PowerUserAccess-Testers** do grupo **Testers**. A propriedade **PolicyDocument** é codificada em URL. Ela é decodificada neste exemplo com o método .NET **UrlDecode**.

```

$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results

```

Saída:

```

GroupName      PolicyDocument                                     PolicyName
-----
Testers        %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",

```

```
        "NotAction": "iam:*",
        "Resource": "*"
    }
]
}
```

- Para obter detalhes da API, consulte [GetGroupPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMGroupPolicyList

O código de exemplo a seguir mostra como usar `Get-IAMGroupPolicyList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna uma lista das políticas em linha incorporadas no grupo **Testers**. Para obter as políticas gerenciadas anexadas ao grupo, use o comando **Get-IAMAttachedGroupPolicyList**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Saída:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Para obter detalhes da API, consulte [ListGroupPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMInstanceProfile

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfile`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna detalhes do nome **ec2instancerole** do perfil da instância definido na AWS conta atual.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

**Saída:**

```
Arn          : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate   : 2/17/2015 2:49:04 PM
InstanceId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles        : {ec2instancerole}
```

- Para obter detalhes da API, consulte [GetInstanceProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-IAMInstanceProfileForRole**

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfileForRole`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo retorna detalhes do perfil de instância associado ao perfil **ec2instancerole**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

**Saída:**

```
Arn          : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
CreateDate   : 2/17/2015 2:49:04 PM
InstanceId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles        : {ec2instancerole}
```

- Para obter detalhes da API, consulte [ListInstanceProfilesForRole](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-IAMInstanceProfileList**

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfileList`.



## Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna uma coleção dos perfis de instância definidos na versão atual Conta da AWS.

```
Get-IAMInstanceProfileList
```

Saída:

```
Arn           : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate    : 2/17/2015 2:49:04 PM
InstanceId    : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path         : /
Roles        : {ec2instancerole}
```

- Para obter detalhes da API, consulte [ListInstanceProfiles](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMLoginProfile

O código de exemplo a seguir mostra como usar Get-IAMLoginProfile.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna a data de criação da senha e se uma redefinição de senha é necessária para o usuário do IAM **David**.

```
Get-IAMLoginProfile -UserName David
```

Saída:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
12/10/2014 3:39:44 PM	False	David

- Para obter detalhes da API, consulte [GetLoginProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMMFADevice

O código de exemplo a seguir mostra como usar `Get-IAMMFADevice`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre o dispositivo de MFA atribuído ao usuário do IAM **David**. Neste exemplo, você percebe que é um dispositivo virtual porque o **SerialNumber** é um ARN em vez do número de série real de um dispositivo físico.

```
Get-IAMMFADevice -UserName David
```

Saída:

EnableDate	SerialNumber	UserName
-----	-----	-----
4/8/2015 9:41:10 AM	arn:aws:iam::123456789012:mfa/David	David

- Para obter detalhes da API, consulte [ListMfaDevices](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `Get-IAMOpenIDConnectProvider`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre o provedor OpenID Connect cujo ARN é **arn:aws:iam::123456789012:oidc-provider/accounts.google.com**. A propriedade **ClientIDList** é uma compilação que contém todos os IDs de cliente definidos para esse provedor.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Saída:

ClientIDList Url	CreateDate	ThumbprintList
-----	-----	-----
---		

```
{MyOIDCApp}                2/3/2015 3:00:30 PM
 {12345abcdefghijklmnopqrst98765uvwxy}  oidc.example.com
```

- Para obter detalhes da API, consulte [GetOpenIdConnectProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMOpenIDConnectProviderList

O código de exemplo a seguir mostra como usar `Get-IAMOpenIDConnectProviderList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna uma lista de ARNS de todos os provedores OpenID Connect definidos na Conta da AWS atual.

```
Get-IAMOpenIDConnectProviderList
```

Saída:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Para obter detalhes da API, consulte [ListOpenIdConnectProviders](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMPolicy

O código de exemplo a seguir mostra como usar `Get-IAMPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna detalhes sobre a política gerenciada cujo ARN é **arn:aws:iam::123456789012:policy/MySamplePolicy**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
Arn          : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount : 0
CreateDate   : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : Z27SI6FQMGNQ2EXAMPLE1
PolicyName  : MySamplePolicy
UpdateDate  : 2/6/2015 10:40:08 AM
```

- Para obter detalhes da API, consulte [GetPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMPolicyList

O código de exemplo a seguir mostra como usar Get-IAMPolicyList.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo retorna uma coleção das três primeiras políticas gerenciadas disponíveis na AWS conta atual. Como não **-scope** está especificado, ele usa como padrão **all** e inclui políticas AWS gerenciadas e gerenciadas pelo cliente.

```
Get-IAMPolicyList -MaxItem 3
```

Saída:

```
Arn          : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount : 0
CreateDate   : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : Z27SI6FQMGNQ2EXAMPLE1
PolicyName  : AWSDirectConnectReadOnlyAccess
UpdateDate  : 2/6/2015 10:40:08 AM

Arn          : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount : 0
```

```

CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable    : True
Path            : /
PolicyId        : NJKMU274MET4EEXAMPLE2
PolicyName      : AmazonGlacierReadOnlyAccess
UpdateDate      : 2/6/2015 10:40:27 AM

Arn             : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount : 0
CreateDate      : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description     :
IsAttachable    : True
Path           : /
PolicyId       : 5ULJS02FYVPYGEXAMPLE3
PolicyName     : AWSMarketplaceFullAccess
UpdateDate    : 2/11/2015 9:21:45 AM

```

Exemplo 2: Este exemplo retorna uma coleção das duas primeiras políticas gerenciadas pelo cliente disponíveis na AWS conta corrente. Ele usa **-Scope local** para limitar a saída somente às políticas gerenciadas pelo cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Saída:

```

Arn             : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount : 0
CreateDate      : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description     :
IsAttachable    : True
Path           : /
PolicyId       : SQVCBLC4VAOUCEXAMPLE4
PolicyName     : MyLocalPolicy
UpdateDate    : 2/12/2015 9:39:53 AM

Arn             : arn:aws:iam::123456789012:policy/policyforec2instanceroles
AttachmentCount : 1
CreateDate      : 2/17/2015 2:51:38 PM

```

```

DefaultVersionId : v11
Description      :
IsAttachable    : True
Path            : /
PolicyId        : X5JPBLJH2Z2S0EXAMPLE5
PolicyName      : policyforec2instancerole
UpdateDate     : 2/18/2015 8:52:31 AM

```

- Para obter detalhes da API, consulte [ListPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMPolicyVersion

O código de exemplo a seguir mostra como usar `Get-IAMPolicyVersion`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna o documento da política na versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyManagedPolicy**. O documento da política na propriedade **Document** é codificado em URL, sendo decodificado neste exemplo com o método .NET **UrlDecode**.

```

$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results

```

Saída:

```

CreateDate          Document
IsDefaultVersion   VersionId
-----
-----
-----
2/12/2015 9:39:53 AM %7B%0A%20%20%22Version%22%3A%20%222012-10...   True
                    v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version": "2012-10-17",
  "Statement":

```

```
{
  "Effect": "Allow",
  "Action": "*",
  "Resource": "*"
}
```

- Para obter detalhes da API, consulte [GetPolicyVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMPolicyVersionList

O código de exemplo a seguir mostra como usar `Get-IAMPolicyVersionList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna a lista de versões disponíveis da política cujo ARN é **arn:aws:iam::123456789012:policy/MyManagedPolicy**. Para obter o documento de política de uma versão específica, use o comando **Get-IAMPolicyVersion** e especifique o **VersionId** do que você deseja.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Saída:

CreateDate VersionId	Document	IsDefaultVersion
----- -----	-----	-----
2/12/2015 9:39:53 AM v2		True
2/12/2015 9:39:09 AM v1		False

- Para obter detalhes da API, consulte [ListPolicyVersions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMRole

O código de exemplo a seguir mostra como usar `Get-IAMRole`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna os detalhes do **lambda\_exec\_role**. Ele inclui o documento da política de confiança que especifica quem pode assumir esse perfil. O documento da política é codificado em URL e pode ser decodificado usando o método .NET **UrlDecode**. Neste exemplo, todos os espaços em branco da política original foram removidos antes de ela ser carregada na política. Para ver os documentos da política de permissões que determinam o que alguém que assume o perfil pode fazer, use **Get-IAMRolePolicy** para políticas em linha e **Get-IAMPolicyVersion** para políticas gerenciadas anexadas.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Saída:

```
Arn                : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22Service%22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%7D%5D%7D
CreateDate         : 4/2/2015 9:16:11 AM
Path               : /
RoleId             : 2YBIKAIBHNKB4EXAMPLE1
RoleName           : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Saída:

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Para obter detalhes da API, consulte [GetRole](#) em Referência de AWS Tools for PowerShell cmdlet.



## Get-IAMRoleList

O código de exemplo a seguir mostra como usar `Get-IAMRoleList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera uma lista de todos os perfis do IAM na Conta da AWS.

```
Get-IAMRoleList
```

Exemplo 2: Esse exemplo de trecho de código recupera uma lista de funções do IAM na AWS conta, as exibe três por vez e espera que você pressione Enter entre cada grupo. Ele passa o valor de **Marker** da chamada anterior para especificar onde o próximo grupo deve começar.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- Para obter detalhes da API, consulte [ListRoles](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMRolePolicy

O código de exemplo a seguir mostra como usar `Get-IAMRolePolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna o documento da política de permissões da política denominada **oneClick\_lambda\_exec\_role\_policy** incorporada no perfil do IAM **lambda\_exec\_role**. O documento resultante da política é codificado em URL. Ela é decodificada neste exemplo com o método .NET **UrlDecode**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
oneClick_lambda_exec_role_policy
$results
```

**Saída:**

PolicyDocument	PolicyName
<pre>           UserName -----           ----- %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%... oneClick_lambda_exec_role_policy      lambda_exec_role </pre>	

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

**Saída:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte [GetRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMRolePolicyList

O código de exemplo a seguir mostra como usar `Get-IAMRolePolicyList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo retorna a lista de nomes de políticas em linha incorporadas no perfil do IAM `lamda_exec_role`. Para ver os detalhes de uma política em linha, use o comando `Get-IAMRolePolicy`.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Saída:

```
oneClick_lambda_exec_role_policy
```

- Para obter detalhes da API, consulte [ListRolePolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMRoleTagList

O código de exemplo a seguir mostra como usar `Get-IAMRoleTagList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo busca a tag associada ao perfil.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Para obter detalhes da API, consulte [ListRoleTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMSAMLProvider

O código de exemplo a seguir mostra como usar `Get-IAMSAMLProvider`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera os detalhes sobre o provedor SAML 2.0 cujo ARM é `arn:aws:iam::123456789012:saml-provider/SAMLADFS`. A resposta inclui o documento de

metadados que você obteve do provedor de identidade para criar a entidade do provedor AWS SAML, bem como as datas de criação e expiração.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

Saída:

```

CreateDate                SAMLMetadataDocument
-----
ValidUntil
-----
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
12/23/2114 12:16:54 PM

```

- Para obter detalhes da API, consulte [GetSamlProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMSAMLProviderList

O código de exemplo a seguir mostra como usar `Get-IAMSAMLProviderList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera a lista de provedores SAML 2.0 criados na Conta da AWS atual. Ele retorna o ARN, a data de criação e a data de expiração de cada provedor SAML.

```
Get-IAMSAMLProviderList
```

Saída:

```

Arn                        CreateDate
-----
ValidUntil
-----
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
12/23/2114 12:16:54 PM

```

- Para obter detalhes da API, consulte [ListSAMLProviders](#) na Referência de Cmdlet do AWS Tools for PowerShell .

## Get-IAMServerCertificate

O código de exemplo a seguir mostra como usar `Get-IAMServerCertificate`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera detalhes sobre o certificado do servidor denominado **MyServerCertificate**. Você pode encontrar os detalhes do certificado nas propriedades **CertificateBody** e **ServerCertificateMetadata**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

### Saída:

```
CertificateBody          : -----BEGIN CERTIFICATE-----

MIICiTCCAfICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6

b24xFDASBgNVBA5TC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAd

BkgkqhkiG9w0BCQEWEG5vb25lQGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

MTIwNDI1MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD

VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC0lBTSBDb25z

b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWVxHmAdBkgkqhkiG9w0BCQEWEG5vb25lQGft

YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn

+a4GmWIWJ

21uUSfwfEvySWtC2XADZ4nB+BLygVIk60CpiwsZ3G93vUEI03IyNoH/

f0wYK8m9T

rDHudUZg3qX4waLG5M43q7Wgc/

MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE

Ibb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4

nUhVVxYUntneD9+h8Mg9q6q

+auNkyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb

FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb

NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

```

-----END CERTIFICATE-----
CertificateChain      :
ServerCertificateMetadata :
  Amazon.IdentityManagement.Model.ServerCertificateMetadata

```

```
$result.ServerCertificateMetadata
```

Saída:

```

Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path              : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM

```

- Para obter detalhes da API, consulte [GetServerCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMServerCertificateList

O código de exemplo a seguir mostra como usar Get-IAMServerCertificateList.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera a lista de certificados de servidor enviados à Conta da AWS atual.

```
Get-IAMServerCertificateList
```

Saída:

```

Arn                : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration         : 1/14/2018 9:52:36 AM
Path              : /Org1/Org2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate        : 4/21/2015 11:14:16 AM

```

- Para obter detalhes da API, consulte [ListServerCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMServiceLastAccessedDetail

O código de exemplo a seguir mostra como usar `Get-IAMServiceLastAccessedDetail`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo fornece detalhes do último serviço acessado pela entidade do IAM (usuário, grupo, perfil ou política) associada na chamada de solicitação.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Saída:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Para obter detalhes da API, consulte [GetServiceLastAccessedDetails](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMServiceLastAccessedDetailWithEntity

O código de exemplo a seguir mostra como usar `Get-IAMServiceLastAccessedDetailWithEntity`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo fornece o carimbo de data e hora do último acesso do serviço na solicitação pela respectiva entidade do IAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f -ServiceNamespace ec2  
$results
```

Saída:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
```

```
Error           :
IsTruncated     : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate  : 12/29/19 11:19:31 AM
JobStatus       : COMPLETED
Marker          :
```

```
$results.EntityDetailsList
```

Saída:

```
EntityInfo                               LastAuthenticated
-----
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Saída:

```
Arn : arn:aws:iam::123456789012:user/TestUser
Id  : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Para obter detalhes da API, consulte [GetServiceLastAccessedDetailsWithEntities](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMSigningCertificate

O código de exemplo a seguir mostra como usar `Get-IAMSigningCertificate`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera detalhes sobre o certificado de assinatura associado ao usuário chamado **Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Saída:



```

CertificateBody : -----BEGIN CERTIFICATE-----
                MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
                VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
                b24xFDASBgNVBAwTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHzAd
                BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI0MjA0NTIxWhcN
                MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
                VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC01BTSBDb25z
                b2x1MRIwEAYDVQQDEw1UZXRhbnQ2LsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFT
                YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
                rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
                Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
                FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStB
                NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status         : Active
UploadDate    : 4/20/2015 1:26:01 PM
UserName      : Bob

```

- Para obter detalhes da API, consulte [ListSigningCertificates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMUser

O código de exemplo a seguir mostra como usar Get-IAMUser.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera detalhes sobre o usuário chamado **David**.

```
Get-IAMUser -UserName David
```

Saída:

```

Arn           : arn:aws:iam::123456789012:user/David
CreateDate    : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path          : /
UserId       : Y4FKWQCXTA52QEXAMPLE1

```

```
UserName      : David
```

Exemplo 2: este exemplo recupera detalhes sobre o usuário do IAM atualmente conectado.

```
Get-IAMUser
```

Saída:

```
Arn           : arn:aws:iam::123456789012:user/Bob
CreateDate    : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path          : /
UserId        : 7K3GJEANSKZF2EXAMPLE2
UserName      : Bob
```

- Para obter detalhes da API, consulte [GetUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMUserList

O código de exemplo a seguir mostra como usar `Get-IAMUserList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera uma coleção de usuários no atual Conta da AWS.

```
Get-IAMUserList
```

Saída:

```
Arn           : arn:aws:iam::123456789012:user/Administrator
CreateDate    : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path          : /
UserId        : 7K3GJEANSKZF2EXAMPLE1
UserName      : Administrator

Arn           : arn:aws:iam::123456789012:user/Bob
CreateDate    : 4/6/2015 12:54:42 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path          : /
```

```

    UserId      : L3EWNONDOM3YUEXAMPLE2
    UserName    : bab

    Arn         : arn:aws:iam::123456789012:user/David
    CreateDate  : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path        : /
    UserId      : Y4FKWQCXTA52QEXAMPLE3
    UserName    : David

```

- Para obter detalhes da API, consulte [ListUsers](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMUserPolicy

O código de exemplo a seguir mostra como usar Get-IAMUserPolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera os detalhes da política em linha denominada **Davids\_IAM\_Admin\_Policy** incorporada no usuário do IAM chamado **David**. O documento de política é codificado em URL.

```

$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results

```

Saída:

```

PolicyDocument                                     PolicyName
-----
-----
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...  Davids_IAM_Admin_Policy
David
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",

```

```
    "Action": [
      "iam:*"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

- Para obter detalhes da API, consulte [GetUserPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMUserPolicyList

O código de exemplo a seguir mostra como usar `Get-IAMUserPolicyList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera a lista de nomes das políticas em linha incorporadas no usuário do IAM chamado **David**.

```
Get-IAMUserPolicyList -UserName David
```

Saída:

```
 Davids_IAM_Admin_Policy
```

- Para obter detalhes da API, consulte [ListUserPolicies](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMUserTagList

O código de exemplo a seguir mostra como usar `Get-IAMUserTagList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo busca a tag associada ao usuário.

```
Get-IAMUserTagList -UserName joe
```

- Para obter detalhes da API, consulte [ListUserTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-IAMVirtualMFADevice

O código de exemplo a seguir mostra como usar `Get-IAMVirtualMFADevice`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera uma coleção dos dispositivos virtuais de MFA atribuídos aos usuários na AWS conta. A propriedade do **User** de cada um é um objeto com detalhes do usuário do IAM ao qual o dispositivo está atribuído.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Saída:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Para obter detalhes da API, consulte [ListVirtualMfaDevices](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMAccessKey

O código de exemplo a seguir mostra como usar `New-IAMAccessKey`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma chave de acesso e um par de chaves de acesso secreto e os atribui ao usuário **David**. Certifique-se de salvar os valores **AccessKeyId** e **SecretAccessKey** em um arquivo, pois este é o único momento em que você pode obter a **SecretAccessKey**. Não

será possível recuperá-la depois. Caso perca a chave secreta, você deve criar um par de chaves de acesso.

```
New-IAMAccessKey -UserName David
```

Saída:

```
AccessKeyId      : AKIAIOSFODNN7EXAMPLE
CreateDate       : 4/13/2015 1:00:42 PM
SecretAccessKey  : wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Status           : Active
UserName         : David
```

- Para obter detalhes da API, consulte [CreateAccessKey](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMAccountAlias

O código de exemplo a seguir mostra como usar `New-IAMAccountAlias`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo altera o alias da AWS conta para **mycompanyaws**. O endereço da página de login do usuário muda para `https://mycompanyaws.signin.aws.amazon.com/console`. O URL original usando o número de ID da conta em vez do alias (`https://<accountidnumber>.signin.aws.amazon.com/console`) continua funcionando. No entanto, todos os URLs baseados em alias definidos anteriormente param de funcionar.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte [CreateAccountAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMGroup

O código de exemplo a seguir mostra como usar `New-IAMGroup`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um grupo do IAM denominado **Developers**.

```
New-IAMGroup -GroupName Developers
```

#### Saída:

```
Arn          : arn:aws:iam::123456789012:group/Developers
CreateDate   : 4/14/2015 11:21:31 AM
GroupId      : QNEJ5PM4NFSQCEXAMPLE1
GroupName    : Developers
Path         : /
```

- Para obter detalhes da API, consulte [CreateGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMInstanceProfile

O código de exemplo a seguir mostra como usar `New-IAMInstanceProfile`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um perfil de instância do IAM denominado **ProfileForDevEC2Instance**. Você deve executar o comando **Add-IAMRoleToInstanceProfile** separadamente para associar o perfil de instância a um perfil do IAM existente que fornece permissões à instância. Por fim, anexe o perfil de instância a uma instância do EC2 ao executá-la. Para isso, use o cmdlet **New-EC2Instance** com o parâmetro **InstanceProfile\_Arn** ou **InstanceProfile\_Name**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

#### Saída:

```
Arn          : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate    : 4/14/2015 11:31:39 AM
InstanceProfileId : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path         : /
Roles        : {}
```

- Para obter detalhes da API, consulte [CreateInstanceProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMLoginProfile

O código de exemplo a seguir mostra como usar `New-IAMLoginProfile`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma senha (temporária) para o usuário do IAM chamado Bob e define a sinalização que exige que o usuário altere a senha na próxima vez que **Bob** fizer login.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Saída:

CreateDate	PasswordResetRequired	UserName
-----	-----	-----
4/14/2015 12:26:30 PM	True	Bob

- Para obter detalhes da API, consulte [CreateLoginProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `New-IAMOpenIDConnectProvider`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um provedor OIDC do IAM associado ao serviço do provedor compatível com OIDC encontrado no URL **https://example.oidcprovider.com** e no ID do cliente **my-testapp-1**. O provedor OIDC fornece a impressão digital. Para autenticar a impressão digital, siga as etapas em <http://docs.aws.amazon.com/IAM/latest/UserGuide/-thumbprint.html>. `identity-providers-oidc-obtain`

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Saída:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```



- Para obter detalhes da API, consulte [CreateOpenIdConnectProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMPolicy

O código de exemplo a seguir mostra como usar `New-IAMPolicy`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova política do IAM na AWS conta atual chamada **MySamplePolicy**. O arquivo **MySamplePolicy.json** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw MySamplePolicy.json)
```

### Saída:

```
Arn           : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount : 0
CreateDate    : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description   :
IsAttachable  : True
Path         : /
PolicyId     : LD4KP6HVFE7WGEXAMPLE1
PolicyName   : MySamplePolicy
UpdateDate   : 4/14/2015 2:45:59 PM
```

- Para obter detalhes da API, consulte [CreatePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMPolicyVersion

O código de exemplo a seguir mostra como usar `New-IAMPolicyVersion`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma versão "v2" da política do IAM cujo ARN é **arn:aws:iam::123456789012:policy/MyPolicy** e a torna a versão padrão. O arquivo

**NewPolicyVersion.json** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Saída:

CreateDate VersionId	Document	IsDefaultVersion
----- -----	-----	-----
4/15/2015 10:54:54 AM v2		True

- Para obter detalhes da API, consulte [CreatePolicyVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMRole

O código de exemplo a seguir mostra como usar **New-IAMRole**.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um perfil denominado **MyNewRole** e anexa a ele a política encontrada no arquivo **NewRoleTrustPolicy.json**. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON. O documento de política exibido na saída é codificado em URL. Ele é decodificado nesse exemplo com o método **.NET UriDecode**.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Saída:

```
Arn : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D%0A%20%20%22Statement%22%3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
```

```
%0D%0A%20%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20%20%20
%22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
%3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
%20%20%20%20%7D%0D%0A%20
%20%5D%0D%0A%7D
CreateDate          : 4/15/2015 11:04:23 AM
Path                : /
RoleId              : V5PAJI2KPN4EAEXAMPLE1
RoleName            : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Para obter detalhes da API, consulte [CreateRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMSAMLProvider

O código de exemplo a seguir mostra como usar `New-IAMSAMLProvider`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma entidade provedora SAML no IAM. Ele é denominado **MySAMLProvider** e descrito pelo documento de metadados SAML encontrado no arquivo **SAMLMetaData.xml**, que foi baixado separadamente do site do provedor de serviços SAML.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

Saída:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Para obter detalhes da API, consulte [CreateSAMLProvider](#) na Referência de Cmdlet do AWS Tools for PowerShell .

## New-IAMServiceLinkedRole

O código de exemplo a seguir mostra como usar New-IAMServiceLinkedRole.

Ferramentas para PowerShell

Exemplo 1: este exemplo cria um perfil vinculado ao serviço para o serviço de ajuste de escala automático.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Para obter detalhes da API, consulte [CreateServiceLinkedRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-IAMUser

O código de exemplo a seguir mostra como usar New-IAMUser.

Ferramentas para PowerShell

Exemplo 1: este exemplo cria um usuário do IAM chamado **Bob**. Se Bob precisar entrar no AWS console, você deverá executar o comando separadamente **New-IAMLoginProfile** para criar um perfil de login com uma senha. Se Bob precisar executar AWS PowerShell ou executar comandos CLI multiplataforma ou AWS fazer chamadas de API, você deverá executar **New-IAMAccessKey** o comando separadamente para criar chaves de acesso.

```
New-IAMUser -UserName Bob
```

**Saída:**

```

Arn          : arn:aws:iam::123456789012:user/Bob
CreateDate   : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path         : /
UserId       : AIDAJWGEFDMEMEXAMPLE1
UserName     : Bob

```

- Para obter detalhes da API, consulte [CreateUser](#) em Referência de AWS Tools for PowerShell cmdlet.

**New-IAMVirtualMFADevice**

O código de exemplo a seguir mostra como usar `New-IAMVirtualMFADevice`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **Base32StringSeed** de que o programa de software de MFA virtual precisa para criar uma conta (como alternativa ao código QR). Depois de configurar o programa com o valor, obtenha dois códigos de autenticação sequencial do programa. Por fim, use o último comando para vincular o dispositivo de MFA virtual ao usuário do IAM **Bob** e sincronizar a conta com os dois códigos de autenticação.

```

$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE

```

**Saída:**

```

-- Pause here to enter base-32 string seed code into virtual MFA program to register
account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012

```

Exemplo 2: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **QRCodePNG** e o gravam em um arquivo. Essa imagem pode ser digitalizada pelo programa de software de MFA virtual para criar uma conta (como alternativa à inserção manual do valor StringSeed Base32). Depois de criar a conta no programa de MFA virtual, obtenha dois códigos de autenticação sequencial e insira-os nos últimos comandos para vincular o dispositivo MFA virtual ao usuário do IAM **Bob** e sincronizar a conta.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path QRCode.png
```

Saída:

```
-- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Para obter detalhes da API, consulte [CreateVirtualMfaDevice](#) em Referência de AWS Tools for PowerShell cmdlet.

## Publish-IAMServerCertificate

O código de exemplo a seguir mostra como usar Publish-IAMServerCertificate.

Ferramentas para PowerShell

Exemplo 1: este exemplo faz upload de um novo certificado de servidor na conta do IAM. Os arquivos contendo o corpo do certificado, a chave privada e (opcionalmente) a cadeia de certificação devem ser codificados em PEM. Observe que os parâmetros exigem o conteúdo real dos arquivos em vez dos nomes deles. Você deve usar o parâmetro switch **-Raw** para processar com êxito o conteúdo do arquivo.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
(Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Saída:

```
Arn                : arn:aws:iam::123456789012:server-certificate/MyTestCert
```

```

Expiration      : 1/14/2018 9:52:36 AM
Path           : /
ServerCertificateId : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate     : 4/21/2015 11:14:16 AM

```

- Para obter detalhes da API, consulte [UploadServerCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Publish-IAMSigningCertificate

O código de exemplo a seguir mostra como usar Publish-IAMSigningCertificate.

### Ferramentas para PowerShell

Exemplo 1: este exemplo faz upload de um novo certificado de assinatura X.509 e o associa ao usuário do IAM chamado **Bob**. O arquivo que contém o corpo do certificado é codificado em PEM. O parâmetro **CertificateBody** exige o conteúdo real do arquivo de certificado em vez do nome do arquivo. Você deve usar o parâmetro switch **-Raw** para processar o arquivo com êxito.

```

Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
SampleSigningCert.pem)

```

### Saída:

```

CertificateBody : -----BEGIN CERTIFICATE-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI1MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQEWEG5vb25lQGFT
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvjx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

```

```
CertificateId : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status       : Active
UploadDate   : 4/20/2015 1:26:01 PM
UserName     : Bob
```

- Para obter detalhes da API, consulte [UploadSigningCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-IAMGroupPolicy

O código de exemplo a seguir mostra como usar Register-IAMGroupPolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo anexa a política gerenciada pelo cliente denominada **TesterPolicy** ao grupo do IAM **Testers**. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão padrão dessa política.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: Este exemplo anexa a política AWS gerenciada nomeada **AdministratorAccess** ao grupo **Admins** do IAM. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Para obter detalhes da API, consulte [AttachGroupPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-IAMRolePolicy

O código de exemplo a seguir mostra como usar Register-IAMRolePolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo anexa a política AWS gerenciada nomeada **SecurityAudit** à função **CoSecurityAuditors** do IAM. Os usuários que assumem esse perfil são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.



```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn  
arn:aws:iam::aws:policy/SecurityAudit
```

- Para obter detalhes da API, consulte [AttachRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-IAMUserPolicy

O código de exemplo a seguir mostra como usar Register-IAMUserPolicy.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo anexa a política AWS gerenciada nomeada **AmazonCognitoPowerUser** ao usuário **Bob** do IAM. O usuário é imediatamente afetado pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/  
AmazonCognitoPowerUser
```

- Para obter detalhes da API, consulte [AttachUserPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMAccessKey

O código de exemplo a seguir mostra como usar Remove-IAMAccessKey.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o par de chaves de AWS acesso com o ID **AKIAIOSFODNN7EXAMPLE** da chave do usuário chamado **Bob**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Para obter detalhes da API, consulte [DeleteAccessKey](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMAccountAlias

O código de exemplo a seguir mostra como usar Remove-IAMAccountAlias.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o alias da conta do seu Conta da AWS. A página de login do usuário com o alias em <https://mycompanyaws.signin.aws.amazon.com/console> no longer works. Em vez disso, você deve usar o URL original com seu número de Conta da AWS identificação em <https://.signin.aws.amazon.com/console>. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte [DeleteAccountAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMAccountPasswordPolicy

O código de exemplo a seguir mostra como usar `Remove-IAMAccountPasswordPolicy`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a política de senha do Conta da AWS e redefine todos os valores para seus padrões originais. Se uma política de senha não existir no momento, a seguinte mensagem de erro será exibida: A política de conta com nome PasswordPolicy não pode ser encontrada.

```
Remove-IAMAccountPasswordPolicy
```

- Para obter detalhes da API, consulte [DeleteAccountPasswordPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMClientIDFromOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `Remove-IAMClientIDFromOpenIDConnectProvider`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo remove o ID do cliente **My-TestApp-3** da lista de IDs de cliente associados ao provedor OIDC do IAM cujo ARN é **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3  
-OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/  
example.oidcprovider.com
```

- Para obter detalhes da API, consulte [RemoveClientIDFromOpenIdConnectProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMGroup

O código de exemplo a seguir mostra como usar `Remove-IAMGroup`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o grupo do IAM denominado **MyTestGroup**. O primeiro comando remove todos os usuários do IAM que são membros do grupo, e o segundo exclui o grupo do IAM. Ambos os comandos funcionam sem nenhuma solicitação de confirmação.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName  
MyTestGroup -Force  
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Para obter detalhes da API, consulte [DeleteGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMGroupPolicy

O código de exemplo a seguir mostra como usar `Remove-IAMGroupPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo remove a política em linha denominada **TesterPolicy** do grupo do IAM **Testers**. Os usuários desse grupo perdem imediatamente as permissões definidas nessa política.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Para obter detalhes da API, consulte [DeleteGroupPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMInstanceProfile

O código de exemplo a seguir mostra como usar `Remove-IAMInstanceProfile`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o perfil de instância do EC2 denominado **MyAppInstanceProfile**. O primeiro comando desassocia todos os perfis do perfil de instância e, em seguida, o segundo comando exclui o perfil de instância.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile  
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Para obter detalhes da API, consulte [DeleteInstanceProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMLoginProfile

O código de exemplo a seguir mostra como usar `Remove-IAMLoginProfile`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o perfil de login do usuário do IAM denominado **Bob**. Isso impede que o usuário faça login no AWS console. Isso não impede que o usuário execute nenhuma AWS chamada de CLI ou API usando chaves de AWS acesso que ainda possam estar anexadas à conta do usuário. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Para obter detalhes da API, consulte [DeleteLoginProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `Remove-IAMOpenIDConnectProvider`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o provedor OIDC do IAM que se conecta ao provedor **example.oidcprovider.com**. Certifique-se de atualizar ou excluir quaisquer perfis que façam referência a esse provedor no elemento **Principal** da política de confiança do perfil.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn  
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes da API, consulte [DeleteOpenIdConnectProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMPolicy

O código de exemplo a seguir mostra como usar `Remove-IAMPolicy`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo exclui a política cujo ARN é **arn:aws:iam::123456789012:policy/MySamplePolicy**. Antes de excluir a política, exclua primeiro todas as versões, exceto a padrão, executando **Remove-IAMPolicyVersion**. Você também deve desassociar a política de qualquer usuário, grupo ou perfil do IAM.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política, desassociando-a de todas as entidades do IAM anexadas e, por fim, excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como a versão padrão em uma compilação e depois exclui cada política na compilação. A terceira linha recupera todos os usuários, grupos e perfis do IAM aos quais a política está anexada. As linhas de quatro a seis desassociam a política de cada entidade anexada. A última linha usa esse comando para remover a política gerenciada e a versão padrão restante. O exemplo inclui o parâmetro switch **-Force** em qualquer linha que precise dele para suprimir solicitações de confirmação.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy  
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |  
Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
```

```
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Para obter detalhes da API, consulte [DeletePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMPolicyVersion

O código de exemplo a seguir mostra como usar `Remove-IAMPolicyVersion`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui a versão identificada como **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MySamplePolicy**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política e depois excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como padrão em uma compilação e, em seguida, usa esse comando para excluir cada política na compilação. A última linha remove a política em si, bem como a versão padrão restante. Observe que, para excluir com êxito uma política gerenciada, você também deve desassociar a política de qualquer usuário, grupo ou perfis usando os comandos **Unregister-IAMUserPolicy**, **Unregister-IAMGroupPolicy** e **Unregister-IAMRolePolicy**. Veja o exemplo do cmdlet **Remove-IAMPolicy**.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
  Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Para obter detalhes da API, consulte [DeletePolicyVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMRole

O código de exemplo a seguir mostra como usar `Remove-IAMRole`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o perfil denominado **MyNewRole** da conta atual do IAM. Antes de excluir o perfil, use primeiro o comando **Unregister-IAMRolePolicy** para desassociar todas as políticas gerenciadas. As políticas em linha são excluídas com o perfil.

```
Remove-IAMRole -RoleName MyNewRole
```

Exemplo 2: este exemplo desassocia todas as políticas gerenciadas do perfil denominado **MyNewRole** e depois o exclui. A primeira linha recupera todas as políticas gerenciadas anexadas ao perfil como uma compilação e, em seguida, desassocia cada política da compilação do perfil. A segunda linha exclui o perfil em si. As políticas em linha são excluídas com o perfil.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -  
RoleName MyNewRole  
Remove-IAMRole -RoleName MyNewRole
```

- Para obter detalhes da API, consulte [DeleteRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMRoleFromInstanceProfile

O código de exemplo a seguir mostra como usar `Remove-IAMRoleFromInstanceProfile`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o perfil denominado **MyNewRole** do perfil de instância do EC2 denominado **MyNewRole**. Um perfil de instância criado no console do IAM sempre tem o mesmo nome do perfil, como neste exemplo. Se você os criar na API ou na CLI, eles poderão ter nomes diferentes.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole  
-Force
```

- Para obter detalhes da API, consulte [RemoveRoleFromInstanceProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMRolePermissionsBoundary

O código de exemplo a seguir mostra como usar `Remove-IAMRolePermissionsBoundary`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um perfil do IAM.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Para obter detalhes da API, consulte [DeleteRolePermissionsBoundary](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMRolePolicy

O código de exemplo a seguir mostra como usar `Remove-IAMRolePolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui a política em linha **S3AccessPolicy** incorporada no perfil do IAM **S3BackupRole**.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Para obter detalhes da API, consulte [DeleteRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMRoleTag

O código de exemplo a seguir mostra como usar `Remove-IAMRoleTag`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a tag da função chamada "MyRoleName" com a chave de tag como "abac". Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```



- Para obter detalhes da API, consulte [UntagRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMSAMLProvider

O código de exemplo a seguir mostra como usar Remove-IAMSAMLProvider.

Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o provedor SAML 2.0 do IAM cujo ARN é **arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFSPProvider
```

- Para obter detalhes da API, consulte [DeleteSAMLProvider](#) na Referência de Cmdlet do AWS Tools for PowerShell .

## Remove-IAMServerCertificate

O código de exemplo a seguir mostra como usar Remove-IAMServerCertificate.

Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o certificado do servidor denominado **MyServerCert**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Para obter detalhes da API, consulte [DeleteServerCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMServiceLinkedRole

O código de exemplo a seguir mostra como usar Remove-IAMServiceLinkedRole.

Ferramentas para PowerShell

Exemplo 1: este exemplo excluiu o perfil vinculado ao serviço. Observe que, se o serviço ainda estiver usando esse perfil, esse comando resultará em uma falha.

```
Remove-IAMServiceLinkedRole -RoleName  
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Para obter detalhes da API, consulte [DeleteServiceLinkedRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMSigningCertificate

O código de exemplo a seguir mostra como usar Remove-IAMSigningCertificate.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o certificado de assinatura com o ID **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** do usuário do IAM chamado **Bob**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Para obter detalhes da API, consulte [DeleteSigningCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMUser

O código de exemplo a seguir mostra como usar Remove-IAMUser.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o usuário do IAM chamado **Bob**.

```
Remove-IAMUser -UserName Bob
```

Exemplo 2: este exemplo exclui a usuária do IAM chamada **Theresa** com todos os elementos que devem ser excluídos primeiro.

```
$name = "Theresa"  
  
# find any groups and remove user from them  
$groups = Get-IAMGroupForUser -UserName $name  
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -  
UserName $name -Force }
```

```
# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
$name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
-UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
$cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
supress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
$mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Para obter detalhes da API, consulte [DeleteUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMUserFromGroup

O código de exemplo a seguir mostra como usar Remove-IAMUserFromGroup.

## Ferramentas para PowerShell

Exemplo 1: este exemplo remove o usuário do IAM **Bob** do grupo **Testers**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Exemplo 2: este exemplo encontra todos os grupos dos quais a usuária do IAM **Theresa** é membro e, em seguida, remove a **Theresa** desses grupos.

```
$groups = Get-IAMGroupForUser -UserName Theresa  
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -  
  UserName Theresa -Force }
```

Exemplo 3: este exemplo mostra uma forma alternativa de remover o usuário do IAM **Bob** do grupo **Testers**.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName  
  Testers -Force
```

- Para obter detalhes da API, consulte [RemoveUserFromGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

### Remove-IAMUserPermissionsBoundary

O código de exemplo a seguir mostra como usar `Remove-IAMUserPermissionsBoundary`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um usuário do IAM.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Para obter detalhes da API, consulte [DeleteUserPermissionsBoundary](#) em Referência de AWS Tools for PowerShell cmdlet.

### Remove-IAMUserPolicy

O código de exemplo a seguir mostra como usar `Remove-IAMUserPolicy`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo exclui a política em linha denominada **AccessToEC2Policy** incorporada no usuário do IAM chamado **Bob**.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Exemplo 2: este exemplo encontra todas as políticas em linha incorporadas na usuária do IAM chamada **Theresa** e depois as exclui.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa  
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName  
Theresa -Force}
```

- Para obter detalhes da API, consulte [DeleteUserPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMUserTag

O código de exemplo a seguir mostra como usar `Remove-IAMUserTag`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo remove a tag do usuário denominado “joe” com a chave de tag como “abac” e “xyzw”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Para obter detalhes da API, consulte [UntagUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-IAMVirtualMFADevice

O código de exemplo a seguir mostra como usar `Remove-IAMVirtualMFADevice`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o dispositivo de MFA virtual do IAM cujo ARN é **arn:aws:iam::123456789012:mfa/bob**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Exemplo 2: este exemplo verifica se a usuária do IAM Theresa tem um dispositivo de MFA atribuído. Se for encontrado, o dispositivo é desabilitado para a usuária do IAM. Se o dispositivo for virtual, ele também é excluído.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
    $mfa.SerialNumber }
}
```

- Para obter detalhes da API, consulte [DeleteVirtualMfaDevice](#) em Referência de AWS Tools for PowerShell cmdlet.

## Request-IAMCredentialReport

O código de exemplo a seguir mostra como usar Request-IAMCredentialReport.

### Ferramentas para PowerShell

Exemplo 1: este exemplo solicita a geração de um novo relatório, que pode ser feito a cada quatro horas. Se o último relatório ainda for recente, o campo Estado será **COMPLETE**. Use **Get-IAMCredentialReport** para visualizar o relatório completo.

```
Request-IAMCredentialReport
```

Saída:

Description	State
-----	-----
No report exists. Starting a new report generation task	STARTED

- Para obter detalhes da API, consulte [GenerateCredentialReport](#) em Referência de AWS Tools for PowerShell cmdlet.

## Request- IAMServiceLastAccessedDetail

O código de exemplo a seguir mostra como usar Request- IAMServiceLastAccessedDetail.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo é um cmdlet equivalente da GenerateServiceLastAccessedDetails API. Isso fornece um ID de trabalho que pode ser usado em Get-IAM e Get-IAM ServiceLastAccessedDetail ServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Para obter detalhes da API, consulte [GenerateServiceLastAccessedDetails](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set- IAMDefaultPolicyVersion

O código de exemplo a seguir mostra como usar Set- IAMDefaultPolicyVersion.

### Ferramentas para PowerShell

Exemplo 1: este exemplo define a versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyPolicy** como versão ativa padrão.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -VersionId v2
```

- Para obter detalhes da API, consulte [SetDefaultPolicyVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set- IAMRolePermissionsBoundary

O código de exemplo a seguir mostra como usar Set- IAMRolePermissionsBoundary.

### Ferramentas para PowerShell

Exemplo 1: este exemplo mostra como definir o limite de permissões de um perfil do IAM. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte [PutRolePermissionsBoundary](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-IAMUserPermissionsBoundary

O código de exemplo a seguir mostra como usar Set-IAMUserPermissionsBoundary.

### Ferramentas para PowerShell

Exemplo 1: este exemplo mostra como definir o limite de permissões do usuário. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary  
arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte [PutUserPermissionsBoundary](#) em Referência de AWS Tools for PowerShell cmdlet.

## Sync-IAMMFADevice

O código de exemplo a seguir mostra como usar Sync-IAMMFADevice.

### Ferramentas para PowerShell

Exemplo 1: este exemplo sincroniza o dispositivo de MFA associado ao usuário do IAM **Bob** e cujo ARN é **arn:aws:iam::123456789012:mfa/bob** com um programa autenticador que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -  
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Exemplo 2: este exemplo sincroniza o dispositivo de MFA do IAM associado à usuária do IAM **Theresa** com um dispositivo físico que tem o número de série **ABCD12345678** e que forneceu os dois códigos de autenticação.



```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -  
AuthenticationCode2 987654 -UserName Theresa
```

- Para obter detalhes da API, consulte [ResyncMfaDevice](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-IAMGroupPolicy

O código de exemplo a seguir mostra como usar `Unregister-IAMGroupPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **arn:aws:iam::123456789012:policy/TesterAccessPolicy** do grupo denominado **Testers**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn  
arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao grupo denominado **Testers** e as desassocia do grupo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -  
Groupname Testers
```

- Para obter detalhes da API, consulte [DetachGroupPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-IAMRolePolicy

O código de exemplo a seguir mostra como usar `Unregister-IAMRolePolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** do perfil denominado **FedTesterRole**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn  
arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao perfil denominado **FedTesterRole** e as desassocia dele.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -  
Rolenamename FedTesterRole
```

- Para obter detalhes da API, consulte [DetachRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-IAMUserPolicy

O código de exemplo a seguir mostra como usar `Unregister-IAMUserPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo desassocia a política gerenciada cujo ARN é **arn:aws:iam::123456789012:policy/TesterPolicy** do usuário do IAM chamado **Bob**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/  
TesterPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao usuário do IAM chamado **Theresa** e as desassocia dele.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username  
Theresa
```

- Para obter detalhes da API, consulte [DetachUserPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMAccessKey

O código de exemplo a seguir mostra como usar `Update-IAMAccessKey`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo altera para **Inactive** o status da chave de acesso **AKIAIOSFODNN7EXAMPLE** do usuário do IAM denominado **Bob**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Para obter detalhes da API, consulte [UpdateAccessKey](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMAccountPasswordPolicy

O código de exemplo a seguir mostra como usar `Update-IAMAccountPasswordPolicy`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a política de senha da conta com as configurações especificadas. Observe que quaisquer parâmetros que não estejam incluídos no comando não são modificados. Em vez disso, eles são redefinidos para os valores padrão.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -RequireUppercaseCharacters $true
```

- Para obter detalhes da API, consulte [UpdateAccountPasswordPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMAssumeRolePolicy

O código de exemplo a seguir mostra como usar `Update-IAMAssumeRolePolicy`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza o perfil do IAM denominado **ClientRole** com uma nova política de confiança, cujo conteúdo vem do arquivo **ClientRolePolicy.json**. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw ClientRolePolicy.json)
```

- Para obter detalhes da API, consulte [UpdateAssumeRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMGroup

O código de exemplo a seguir mostra como usar Update-IAMGroup.

### Ferramentas para PowerShell

Exemplo 1: este exemplo renomeia o grupo do IAM **Testers** para **AppTesters**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Exemplo 2: este exemplo altera o caminho do grupo do IAM **AppTesters** para **/Org1/Org2/**. Isso altera o ARN do grupo para **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte [UpdateGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMLoginProfile

O código de exemplo a seguir mostra como usar Update-IAMLoginProfile.

### Ferramentas para PowerShell

Exemplo 1: este exemplo define uma nova senha temporária para o usuário **Bob** do IAM e exige que a pessoa altere a senha na próxima vez que fizer login.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired $true
```

- Para obter detalhes da API, consulte [UpdateLoginProfile](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMOpenIDConnectProviderThumbprint

O código de exemplo a seguir mostra como usar Update-IAMOpenIDConnectProviderThumbprint.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a lista de impressões digitais do certificado do provedor OIDC cujo ARN é **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**, a fim de usar uma nova impressão digital. O provedor OIDC compartilha o novo valor quando o certificado associado ao provedor é alterado.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
7359755EXAMPLEEabc3060bce3EXAMPLEEc4542a3
```

- Para obter detalhes da API, consulte [UpdateOpenIdConnectProviderThumbprint](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMRole

O código de exemplo a seguir mostra como usar Update-IAMRole.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a descrição do perfil e o valor máximo da duração da sessão (em segundos) para o qual a sessão de um perfil pode ser solicitada.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Para obter detalhes da API, consulte [UpdateRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMRoleDescription

O código de exemplo a seguir mostra como usar Update-IAMRoleDescription.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a descrição de um perfil do IAM na sua conta.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Para obter detalhes da API, consulte [UpdateRoleDescription](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMSAMLProvider

O código de exemplo a seguir mostra como usar Update-IAMSAMLProvider.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza o provedor SAML no IAM cujo ARN é **arn:aws:iam::123456789012:saml-provider/SAMLADFS** com um novo documento de metadados SAML do arquivo **SAMLMetaData.xml**. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Para obter detalhes da API, consulte [UpdateSamlProvider](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMServerCertificate

O código de exemplo a seguir mostra como usar Update-IAMServerCertificate.

### Ferramentas para PowerShell

Exemplo 1: este exemplo renomeia o certificado denominado **MyServerCertificate** para **MyRenamedServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewServerCertificateName MyRenamedServerCertificate
```

Exemplo 2: este exemplo move o certificado denominado **MyServerCertificate** para o caminho **/Org1/Org2/**. Isso altera o ARN do recurso para **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /  
Org1/Org2/
```

- Para obter detalhes da API, consulte [UpdateServerCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMSigningCertificate

O código de exemplo a seguir mostra como usar Update-IAMSigningCertificate.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza o certificado associado ao usuário do IAM chamado **Bob** e cujo ID do certificado é **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** para marcá-lo como inativo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -  
UserName Bob -Status Inactive
```

- Para obter detalhes da API, consulte [UpdateSigningCertificate](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-IAMUser

O código de exemplo a seguir mostra como usar Update-IAMUser.

### Ferramentas para PowerShell

Exemplo 1: este exemplo renomeia o usuário do IAM **Bob** para **Robert**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Exemplo 2: este exemplo altera o caminho do usuário do IAM **Bob** para **/Org1/Org2/**, o que efetivamente altera o ARN do usuário para **arn:aws:iam::123456789012:user/Org1/Org2/bob**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte [UpdateUser](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-IAMGroupPolicy

O código de exemplo a seguir mostra como usar Write-IAMGroupPolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma política em linha denominada **AppTesterPolicy** e a incorpora no grupo do IAM **AppTesters**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem no arquivo **apptesterpolicy.json**. Observe que você deve usar o parâmetro **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -  
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Para obter detalhes da API, consulte [PutGroupPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-IAMRolePolicy

O código de exemplo a seguir mostra como usar Write-IAMRolePolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma política em linha denominada **FedTesterRolePolicy** e a incorpora no perfil do IAM **FedTesterRole**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **FedTesterPolicy.json**. Observe que você deve usar o parâmetro **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -  
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Para obter detalhes da API, consulte [PutRolePolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-IAMUserPolicy

O código de exemplo a seguir mostra como usar Write-IAMUserPolicy.



## Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma política em linha denominada **EC2AccessPolicy** e a incorpora no usuário do IAM **Bob**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **EC2AccessPolicy.json**. Observe que você deve usar o parâmetro **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-Content -Raw EC2AccessPolicy.json)
```

- Para obter detalhes da API, consulte [PutUserPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Kinesis usando o Tools for PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-KINRecord**

O código de exemplo a seguir mostra como usar Get-KINRecord.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo mostra como retornar e extrair dados de uma série de um ou mais registros. O iterador fornecido para `Get-InRecord` determina a posição inicial dos registros a serem retornados, os quais, neste exemplo, são capturados em uma variável, `$records`. Cada registro individual pode então ser acessado indexando a coleção `$records`. Supondo que os dados no registro sejam texto codificado em UTF-8, o comando final mostra como você pode extrair os dados do `MemoryStream` objeto e retorná-los como texto para o console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNAP"
```

Saída:

```
MillisBehindLatest NextShardIterator           Records
-----
0                AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Saída:

```
ApproximateArrivalTimestamp Data                PartitionKey SequenceNumber
-----
3/7/2016 5:14:33 PM          System.IO.MemoryStream Key1
4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Saída:

```
test data from string
```

- Para obter detalhes da API, consulte [GetRecord](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-KINShardIterator

O código de exemplo a seguir mostra como usar `Get-KINShardIterator`.

## Ferramentas para PowerShell

Exemplo 1: retorna um iterador de fragmento para o fragmento e a posição inicial especificados. Detalhes dos identificadores de fragmentos e dos números de sequência podem ser obtidos na saída do cmdlet `Get-KinStream`, fazendo referência à coleção `Shards` do objeto de fluxo retornado. O iterador retornado pode ser usado com o cmdlet `Get-KinRecord` para extrair registros de dados no fragmento.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -  
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Saída:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Para obter detalhes da API, consulte [GetShardIterator](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-KINStream

O código de exemplo a seguir mostra como usar `Get-KINStream`.

## Ferramentas para PowerShell

Exemplo 1: retorna detalhes do fluxo especificado.

```
Get-KINStream -StreamName "mystream"
```

Saída:

```
HasMoreShards      : False  
RetentionPeriodHours : 24  
Shards             : {}  
StreamARN          : arn:aws:kinesis:us-west-2:123456789012:stream/mystream  
StreamName         : mystream  
StreamStatus       : ACTIVE
```

- Para obter detalhes da API, consulte [DescribeStream](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-KINStream

O código de exemplo a seguir mostra como usar New-KINStream.

### Ferramentas para PowerShell

Exemplo 1: cria um novo fluxo. Por padrão, esse cmdlet não retorna nenhuma saída, então a PassThru opção - é adicionada para retornar o valor fornecido ao StreamName parâmetro - para uso posterior.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Para obter detalhes da API, consulte [CreateStream](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-KINStream

O código de exemplo a seguir mostra como usar Remove-KINStream.

### Ferramentas para PowerShell

Exemplo 1: exclui o fluxo especificado. Você será solicitado a confirmar antes que o comando seja executado. Para suprimir a solicitação de confirmação, use a opção -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Para obter detalhes da API, consulte [DeleteStream](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-KINRecord

O código de exemplo a seguir mostra como usar Write-KINRecord.

### Ferramentas para PowerShell

Exemplo 1: grava um registro contendo a string fornecida ao parâmetro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey "Key1"
```

Exemplo 2: grava um registro contendo os dados contidos no arquivo especificado. O arquivo é tratado como uma sequência de bytes, portanto, se ele contiver texto, ele deverá ser gravado com qualquer codificação necessária antes de ser usado com esse cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey  
"Key2"
```

- Para obter detalhes da API, consulte [PutRecord](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de Lambda usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Lambda.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-LMResourceTag**

O código de exemplo a seguir mostra como usar Add-LMResourceTag.

### Ferramentas para PowerShell

Exemplo 1: adiciona as três tags (Washington, Oregon e Califórnia) e seus valores associados à função especificada identificada por seu ARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Para obter detalhes da API, consulte [TagResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMAccountSetting

O código de exemplo a seguir mostra como usar Get-LMAccountSetting.

### Ferramentas para PowerShell

Exemplo 1: este exemplo é apresentado para comparar o limite da conta e o uso da conta

```
Get-LMAccountSetting | Select-Object
@{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
@{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Saída:

```
TotalCodeSizeLimit TotalCodeSizeUsed
-----
80530636800          15078795
```

- Para obter detalhes da API, consulte [GetAccountSettings](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMAlias

O código de exemplo a seguir mostra como usar Get-LMAlias.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera os pesos da configuração de roteamento para o alias de uma função do Lambda específica.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
RoutingConfig
```

**Saída:**

```
AdditionalVersionWeights
-----
{[1, 0.6]}
```

- Para obter detalhes da API, consulte [GetAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-LMFunctionConcurrency**

O código de exemplo a seguir mostra como usar `Get-LMFunctionConcurrency`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo obtém a simultaneidade reservada para a função do Lambda

```
Get-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -Select *
```

**Saída:**

```
ReservedConcurrentExecutions
-----
100
```

- Para obter detalhes da API, consulte [GetFunctionConcurrency](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-LMFunctionConfiguration**

O código de exemplo a seguir mostra como usar `Get-LMFunctionConfiguration`.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo retorna a configuração específica de versão de uma função do Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MyLambdaFunction123" -Qualifier
"PowerShellAlias"
```

**Saída:**

```
CodeSha256           : uW0W0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize             : 1426
DeadLetterConfig     : Amazon.Lambda.Model.DeadLetterConfig
Description          : Verson 3 to test Aliases
Environment          : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn          : arn:aws:lambda:us-
east-1:123456789012:function:MyLambdaFunction123
                    : PowershellAlias
FunctionName         : MyLambdaFunction123
Handler              : lambda_function.launch_instance
KMSKeyArn            :
LastModified         : 2019-12-25T09:52:59.872+0000
LastUpdateStatus     : Successful
LastUpdateStatusReason :
LastUpdateStatusReasonCode :
Layers               : {}
MasterArn            :
MemorySize           : 128
RevisionId           : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                 : arn:aws:iam::123456789012:role/service-role/lambda
Runtime              : python3.8
State                : Active
StateReason          :
StateReasonCode      :
Timeout              : 600
TracingConfig        : Amazon.Lambda.Model.TracingConfigResponse
Version              : 4
VpcConfig            : Amazon.Lambda.Model.VpcConfigDetail
```

- Para obter detalhes da API, consulte [GetFunctionConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-LMFunctionList**

O código de exemplo a seguir mostra como usar Get-LMFunctionList.

**Ferramentas para PowerShell**

Exemplo 1: este exemplo mostra todas as funções do Lambda com tamanho de código classificado



```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
RunTime, Timeout, CodeSize
```

Saída:

FunctionName CodeSize ----- -----	Runtime	Timeout
test 243	python2.7	3
MylambdaFunction123 659	python3.8	600
myfuncpython1 675	python3.8	303

- Para obter detalhes da API, consulte [ListFunctions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMPolicy

O código de exemplo a seguir mostra como usar Get-LMPolicy.

### Ferramentas para PowerShell

Exemplo 1: este exemplo mostra a política de função da função do Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Saída:

```
{"Version":"2012-10-17","Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:
east-1:123456789102:function:test"]}]}
```

- Para obter detalhes da API, consulte [GetPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMProvisionedConcurrencyConfig

O código de exemplo a seguir mostra como usar `Get-LMProvisionedConcurrencyConfig`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo obtém a configuração de simultaneidade provisionada para o alias especificado da função do Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
Qualifier "NewAlias1"
```

### Saída:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                    : IN_PROGRESS
StatusReason                              :
```

- Para obter detalhes da API, consulte [GetProvisionedConcurrencyConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMProvisionedConcurrencyConfigList

O código de exemplo a seguir mostra como usar `Get-LMProvisionedConcurrencyConfigList`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera a lista de configurações de simultaneidade provisionada para uma função do Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MyLambdaFunction123"
```

- Para obter detalhes da API, consulte [ListProvisionedConcurrencyConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMResourceTag

O código de exemplo a seguir mostra como usar `Get-LMResourceTag`.

## Ferramentas para PowerShell

Exemplo 1: recupera as tags e seus valores atualmente definidos na função especificada.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Saída:

```
Key          Value
---          -
California Sacramento
Oregon       Salem
Washington Olympia
```

- Para obter detalhes da API, consulte [ListTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-LMVersionsByFunction

O código de exemplo a seguir mostra como usar Get-LMVersionsByFunction.

## Ferramentas para PowerShell

Exemplo 1: este exemplo retorna a lista de configurações específicas de versão para cada versão da função do Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Saída:

```
FunctionName      Runtime  MemorySize Timeout CodeSize LastModified
-----
RoleName
-----
-----
MylambdaFunction123 python3.8      128    600    659
2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8      128     5    1426
2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8      128     5    1426
2019-12-25T09:39:36.779+0000 lambda
```

```
MyLambdaFunction123 python3.8          128      600      1426
2019-12-25T09:52:59.872+0000 lambda
```

- Para obter detalhes da API, consulte [ListVersionsByFunction](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-LMAlias

O código de exemplo a seguir mostra como usar New-LMAlias.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria um novo alias do Lambda para versão e configuração de roteamento especificadas a fim de indicar o percentual de solicitações de invocação que ele receberá.

```
New-LMAlias -FunctionName "MyLambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6"} -Description "Alias for
version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Para obter detalhes da API, consulte [CreateAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## Publish-LMFunction

O código de exemplo a seguir mostra como usar Publish-LMFunction.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova função C# (dotnetcore1.0 runtime) nomeada no MyFunction AWS Lambda, fornecendo os binários compilados para a função a partir de um arquivo zip no sistema de arquivos local (caminhos relativos ou absolutos podem ser usados). As funções Lambda do C# especificam o manipulador da função usando a designação: `:Namespace.AssemblyName ClassName::MethodName`. É necessário substituir adequadamente o nome da montagem (sem o sufixo `.dll`), o namespace, o nome da classe e o nome do método da especificação do manipulador. A nova função terá as variáveis de ambiente `envvar1` e `envvar2` configuradas com base nos valores fornecidos.

```
Publish-LMFunction -Description "My C# Lambda Function" `
-FunctionName MyFunction `
```

```
-ZipFilename .\MyFunctionBinaries.zip `
-Handler "AssemblyName::Namespace.ClassName::MethodName" `
-Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
-Runtime dotnetcore1.0 `
-Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

### Saída:

```
CodeSha256      : /NgBMd...gq71I=
CodeSize       : 214784
DeadLetterConfig :
Description    : My C# Lambda Function
Environment    : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn    : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName   : MyFunction
Handler        : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn     :
LastModified   : 2016-12-29T23:50:14.207+0000
MemorySize    : 128
Role           : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime        : dotnetcore1.0
Timeout        : 3
Version        : $LATEST
VpcConfig     :
```

Exemplo 2: este exemplo é semelhante ao anterior, com a exceção de que os binários da função são carregados primeiramente em um bucket do Amazon S3 (que deve estar na mesma região da função do Lambda desejada) e o objeto resultante do S3 será referenciado ao criar a função.

```
Write-S3Object -BucketName mybucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
-FunctionName MyFunction `
-BucketName mybucket `
-Key MyFunctionBinaries.zip `
-Handler "AssemblyName::Namespace.ClassName::MethodName" `
-Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
-Runtime dotnetcore1.0 `
-Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Para obter detalhes da API, consulte [CreateFunction](#) em Referência de AWS Tools for PowerShell cmdlet.

## Publish-LMVersion

O código de exemplo a seguir mostra como usar Publish-LMVersion.

### Ferramentas para PowerShell

Exemplo 1: este exemplo cria uma versão para o snapshot existente do código da função do Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing Existing Snapshot of function code as a new version through Powershell"
```

- Para obter detalhes da API, consulte [PublishVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMAlias

O código de exemplo a seguir mostra como usar Remove-LMAlias.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui o alias da função do Lambda mencionado no comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Para obter detalhes da API, consulte [DeleteAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMFunction

O código de exemplo a seguir mostra como usar Remove-LMFunction.

### Ferramentas para PowerShell

Exemplo 1: este exemplo exclui uma versão específica de uma função do Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Para obter detalhes da API, consulte [DeleteFunction](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMFunctionConcurrency

O código de exemplo a seguir mostra como usar Remove-LMFunctionConcurrency.

### Ferramentas para PowerShell

Exemplo 1: este exemplo remove a simultaneidade de função da função do Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MyLambdaFunction123"
```

- Para obter detalhes da API, consulte [DeleteFunctionConcurrency](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMPermission

O código de exemplo a seguir mostra como usar Remove-LMPermission.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a política de função para o especificado StatementId de uma função Lambda.

```
$policy = Get-LMPolicy -FunctionName "MyLambdaFunction123" -Select Policy |  
ConvertFrom-Json | Select-Object -ExpandProperty Statement  
Remove-LMPermission -FunctionName "MyLambdaFunction123" -StatementId $policy[0].Sid
```

- Para obter detalhes da API, consulte [RemovePermission](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMProvisionedConcurrencyConfig

O código de exemplo a seguir mostra como usar Remove-LMProvisionedConcurrencyConfig.

### Ferramentas para PowerShell

Exemplo 1: este exemplo remove a configuração de simultaneidade provisionada para um alias específico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -Qualifier  
"NewAlias1"
```

- Para obter detalhes da API, consulte [DeleteProvisionedConcurrencyConfig](#)em Referência de AWS Tools for PowerShell cmdlet.

## Remove-LMResourceTag

O código de exemplo a seguir mostra como usar Remove-LMResourceTag.

### Ferramentas para PowerShell

Exemplo 1: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Uma única chamada será feita para o serviço a fim de remover as tags.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Exemplo 2: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Isso acontece porque a chamada para o serviço é feita pela tag fornecida.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Para obter detalhes da API, consulte [UntagResource](#)em Referência de AWS Tools for PowerShell cmdlet.

## Update-LMAlias

O código de exemplo a seguir mostra como usar Update-LMAlias.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a configuração de um alias de função do Lambda existente. Ele atualiza o RoutingConfiguration valor para transferir 60% (0,6) do tráfego para a versão 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6}
```



- Para obter detalhes da API, consulte [UpdateAlias](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-LMFunctionCode

O código de exemplo a seguir mostra como usar Update-LMFunctionCode.

### Ferramentas para PowerShell

Exemplo 1: atualiza a função chamada 'MyFunction' com o novo conteúdo contido no arquivo zip especificado. Para uma função do Lambda em C# .NET Core, o arquivo zip deve conter a montagem compilada.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Exemplo 2: este exemplo é semelhante ao anterior, mas usa um objeto do Amazon S3 contendo o código atualizado para atualizar a função.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName mybucket -Key  
UpdatedCode.zip
```

- Para obter detalhes da API, consulte [UpdateFunctionCode](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-LMFunctionConfiguration

O código de exemplo a seguir mostra como usar Update-LMFunctionConfiguration.

### Ferramentas para PowerShell

Exemplo 1: este exemplo atualiza a configuração da função do Lambda existente

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler  
"lambda_function.launch_instance" -Timeout 600 -Environment_Variable  
{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/  
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:  
123456789101:MyfirstTopic
```

- Para obter detalhes da API, consulte [UpdateFunctionConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-LMFunctionConcurrency

O código de exemplo a seguir mostra como usar Write-LMFunctionConcurrency.

### Ferramentas para PowerShell

Exemplo 1: este exemplo aplica as configurações de simultaneidade para a função de maneira geral.

```
Write-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -
ReservedConcurrentExecution 100
```

- Para obter detalhes da API, consulte [PutFunctionConcurrency](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-LMProvisionedConcurrencyConfig

O código de exemplo a seguir mostra como usar Write-LMProvisionedConcurrencyConfig.

### Ferramentas para PowerShell

Exemplo 1: este exemplo adiciona uma configuração de simultaneidade provisionada ao alias de uma função

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Para obter detalhes da API, consulte [PutProvisionedConcurrencyConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de Amazon ML usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon ML.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### **Get-MLBatchPrediction**

O código de exemplo a seguir mostra como usar `Get-MLBatchPrediction`.

Ferramentas para PowerShell

Exemplo 1: retorna os metadados detalhados para uma previsão em lote com ID de identificação.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Para obter detalhes da API, consulte [GetBatchPrediction](#) em Referência de AWS Tools for PowerShell cmdlet.

### **Get-MLBatchPredictionList**

O código de exemplo a seguir mostra como usar `Get-MLBatchPredictionList`.

Ferramentas para PowerShell

Exemplo 1: retorna uma lista de todos `BatchPredictions` os registros de dados associados que correspondem ao critério de pesquisa fornecido na solicitação.

```
Get-MLBatchPredictionList
```

Exemplo 2: Retorna uma lista de todos `BatchPredictions` com o status `CONCLUÍDO`.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes da API, consulte [DescribeBatchPredictions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-MLDataSource

O código de exemplo a seguir mostra como usar Get-MLDataSource.

### Ferramentas para PowerShell

Exemplo 1: retorna os metadados, o status e as informações do arquivo de dados de a DataSource com o ID de identificação

```
Get-MLDataSource -DataSourceId ID
```

- Para obter detalhes da API, consulte [GetDataSource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-MLDataSourceList

O código de exemplo a seguir mostra como usar Get-MLDataSourceList.

### Ferramentas para PowerShell

Exemplo 1: Retorna uma lista de todos DataSources e seus registros de dados associados.

```
Get-MLDataSourceList
```

Exemplo 2: Retorna uma lista de todos DataSources com o status CONCLUÍDO.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes da API, consulte [DescribeDataSources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-MLEvaluation

O código de exemplo a seguir mostra como usar Get-MLEvaluation.

## Ferramentas para PowerShell

Exemplo 1: retorna metadados e status para uma avaliação com ID de identificação.

```
Get-MLEvaluation -EvaluationId ID
```

- Para obter detalhes da API, consulte [GetEvaluation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-MLEvaluationList

O código de exemplo a seguir mostra como usar Get-MLEvaluationList.

## Ferramentas para PowerShell

Exemplo 1: Retorna uma lista de todos os recursos de avaliação

```
Get-MLEvaluationList
```

Exemplo 2: Retorna uma lista de todas as avaliações com o status CONCLUÍDO.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes da API, consulte [DescribeEvaluations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-MLModel

O código de exemplo a seguir mostra como usar Get-MLModel.

## Ferramentas para PowerShell

Exemplo 1: Retorna os metadados detalhados, o status, o esquema e as informações do arquivo de dados de um MLModel com ID de identificação.

```
Get-MLModel -ModelId ID
```

- Para obter detalhes da API, consulte [AWS Tools for PowerShell GetMLModel](#) em Cmdlet Reference.

## Get-MLModelList

O código de exemplo a seguir mostra como usar `Get-MLModelList`.

### Ferramentas para PowerShell

Exemplo 1: Retorna uma lista de todos os modelos e seus registros de dados associados.

```
Get-MLModelList
```

Exemplo 2: Retorna uma lista de todos os modelos com o status CONCLUÍDO.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes da API, consulte [DescribeMLModels em Cmdlet Reference](#).AWS Tools for PowerShell

## Get-MLPrediction

O código de exemplo a seguir mostra como usar `Get-MLPrediction`.

### Ferramentas para PowerShell

Exemplo 1: Envie um registro para o URL do endpoint de previsão em tempo real do modelo com ID de identificação.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Para obter detalhes da API, consulte [Predict](#) in AWS Tools for PowerShell Cmdlet Reference.

## New-MLBatchPrediction

O código de exemplo a seguir mostra como usar `New-MLBatchPrediction`.

### Ferramentas para PowerShell

Exemplo 1: Crie uma nova solicitação de previsão de lote para o modelo com ID de identificação e coloque a saída no local especificado do S3.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Para obter detalhes da API, consulte [CreateBatchPrediction](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-MLDataSourceFromS3

O código de exemplo a seguir mostra como usar `New-MLDataSourceFromS3`.

### Ferramentas para PowerShell

Exemplo 1: Crie uma fonte de dados com dados de um local do S3, com um nome de `NAME` e um esquema de `SCHEMA`.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- Para obter detalhes da API, consulte [CreateDataSourceFromS3 em Referência](#) de AWS Tools for PowerShell cmdlet.

## New-MLEvaluation

O código de exemplo a seguir mostra como usar `New-MLEvaluation`.

### Ferramentas para PowerShell

Exemplo 1: criar uma avaliação para um determinado ID de fonte de dados e ID de modelo

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Para obter detalhes da API, consulte [CreateEvaluation](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-MLModel

O código de exemplo a seguir mostra como usar `New-MLModel`.

### Ferramentas para PowerShell

Exemplo 1: Crie um novo modelo com dados de treinamento.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Para obter detalhes da API, consulte [CreateMLModel em Cmdlet](#) Reference AWS Tools for PowerShell .

## New-MLRealtimeEndpoint

O código de exemplo a seguir mostra como usar New-MLRealtimeEndpoint.

### Ferramentas para PowerShell

Exemplo 1: Crie um novo endpoint de previsão em tempo real para o ID do modelo fornecido.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Para obter detalhes da API, consulte [CreateRealtimeEndpoint](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de Macie usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com Macie.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### Get-MAC2FindingList

O código de exemplo a seguir mostra como usar Get-MAC2FindingList.



## Ferramentas para PowerShell

Exemplo 1: Retorna uma lista de descobertas contendo uma detecção FindingIds de dados confidenciais com o tipo “CREDIT\_CARD\_NUMBER” ou “US\_SOCIAL\_SECURITY\_NUMBER”

```
$criterionAddProperties = New-Object
    Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
    "CREDIT_CARD_NUMBER"
    "US_SOCIAL_SECURITY_NUMBER"
)

$FindingCriterion = @{
    'classificationDetails.result.sensitiveData.detections.type' =
    [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- Para obter detalhes da API, consulte [ListFindings](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS OpsWorks exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS OpsWorks.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### New-OPSDeployment

O código de exemplo a seguir mostra como usar New-OPSDeployment.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando cria uma nova implantação de aplicativo em todas as instâncias baseadas em Linux em uma camada no Stacks. AWS OpsWorks Mesmo se você especificar um ID de camada, também deverá especificar um ID de pilha. O comando permite que a implantação reinicie as instâncias, se necessário.

```
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z" -LayerId
"511b99c5-ec78-4caa-8a9d-1440116ffd1b" -AppId "0f7a109c-bf68-4336-8cb9-
d37fe0b8c61d" -Command_Name deploy -Command_Arg @{Name="allow_reboot";Value="true"}
```

Exemplo 2: Esse comando implanta a **appsetup** receita do **phpapp** livro de receitas e a **secbaseline** receita do livro de receitas. **testcookbook** O destino de implantação é uma instância, mas o ID da pilha e o ID da camada também são necessários. O **allow\_reboot** atributo do parâmetro Command\_Arg está definido como **true**, o que permite que a implantação reinicie as instâncias, se necessário.

```
$commandArgs = '{ "Name":"execute_recipes", "Args"{ "recipes":
["phpapp::appsetup","testcookbook::secbaseline"] } }'
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z"
-LayerId "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -InstanceId
"d89a6118-0007-4ccf-a51e-59f844127021" -Command_Name $commandArgs -Command_Arg
@{Name="allow_reboot";Value="true"
```

- Para obter detalhes da API, consulte [CreateDeployment](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS Price List exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Price List.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Get-PLSAttributeValue

O código de exemplo a seguir mostra como usar Get-PLSAttributeValue.

### Ferramentas para PowerShell

Exemplo 1: retorna os valores do atributo 'volumeType' para o Amazon EC2 na região us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-east-1
```

### Saída:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Para obter detalhes da API, consulte [GetAttributeValues](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-PLSProduct

O código de exemplo a seguir mostra como usar Get-PLSProduct.

### Ferramentas para PowerShell

Exemplo 1: Retorna detalhes de todos os produtos para o Amazon EC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
{"product":{"productFamily":"Compute Instance","attributes":{"enhancedNetworkingSupported":"Yes","memory":"30.5 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS Region","storage":"EBS only","instanceFamily":"Memory optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up to 10 Gigabit","servicename":"Amazon Elastic Compute Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX, Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...}}
```

Exemplo 2: Retorna dados do Amazon EC2 na região us-east-1 filtrados por tipos de volume de “Uso geral” que são suportados por SSD.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter @{"Type":"TERM_MATCH";Field="volumeType";Value="General Purpose"},@{"Type":"TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-east-1
```

Saída:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-backed","maxThroughputvolume":"160 MB/sec","volumeType":"General Purpose","maxIopsvolume":"10000"},...}}
```

- Para obter detalhes da API, consulte [GetProducts](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-PLSService

O código de exemplo a seguir mostra como usar Get-PLSService.

### Ferramentas para PowerShell

Exemplo 1: retorna os metadados de todos os códigos de serviço disponíveis na região us-east-1.

```
Get-PLSService -Region us-east-1
```

Saída:

```
AttributeNames                                     ServiceCode
-----
{productFamily, servicecode, groupDescription, termType...} AWSBudgets
{productFamily, servicecode, termType, usagetype...}   AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}   AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}   AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}   AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}   AWSConfig
...
```

Exemplo 2: retorna os metadados do serviço Amazon EC2 na região us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
AttributeNames                                     ServiceCode
-----
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Para obter detalhes da API, consulte [DescribeServices](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos de Resource Groups usando Tools for PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Resource Groups.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-RGResourceTag

O código de exemplo a seguir mostra como usar Add-RGResourceTag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona a chave de tag 'Instances' com o valor 'workboxes' ao determinado grupo de recursos arn

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes
```

### Saída:

```
Arn                                     Tags
---                                     ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
workboxes]}
```

- Para obter detalhes da API, consulte [Tag](#) in AWS Tools for PowerShell Cmdlet Reference.

### Find-RGResource

O código de exemplo a seguir mostra como usar Find-RGResource.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um tipo de recurso ResourceQuery for Instance com filtros de tag e encontra recursos.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
    ResourceTypeFilters = @('AWS::EC2::Instance')
    TagFilters = @( @{
        Key = 'auto'
        Values = @('no')
    })
}

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
ResourceIdentifiers
```

Saída:

ResourceArn	ResourceType
-----	-----
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b	AWS::EC2::Instance

- Para obter detalhes da API, consulte [SearchResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGGroup

O código de exemplo a seguir mostra como usar Get-RGGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera o grupo de recursos de acordo com o nome do grupo

```
Get-RGGroup -GroupName auto-no
```

Saída:

Description	GroupArn	Name
-----	-----	----

```
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Para obter detalhes da API, consulte [GetGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGGroupList

O código de exemplo a seguir mostra como usar Get-RGGroupList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista o grupo de recursos já criado.

```
Get-RGGroupList
```

Saída:

GroupArn	GroupName
-----	-----
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no	auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes	auto-yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600	build600

- Para obter detalhes da API, consulte [ListGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGGroupQuery

O código de exemplo a seguir mostra como usar Get-RGGroupQuery.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca a consulta de recursos para o determinado grupo de recursos

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Saída:

Query	Type
-------	------



```

-----
-----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0

```

- Para obter detalhes da API, consulte [GetGroupQuery](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGGroupResourceList

O código de exemplo a seguir mostra como usar Get-RGGroupResourceList.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os recursos do grupo com base no filtro por tipo de recurso

```

Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
-GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers

```

Saída:

ResourceArn	ResourceType
-----	-----
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1	AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc	AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa	AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012	AWS::EC2::Instance

- Para obter detalhes da API, consulte [ListGroupResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGResourceTag

O código de exemplo a seguir mostra como usar Get-RGResourceTag.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as tags para o determinado grupo de recursos arn

```

Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes

```

**Saída:**

```
Key          Value
---          -
Instances    workboxes
```

- Para obter detalhes da API, consulte [GetTags](#) em Referência de AWS Tools for PowerShell cmdlet.

**New-RGGroup**

O código de exemplo a seguir mostra como usar New-RGGroup.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo cria um novo grupo de AWS recursos de Resource Groups baseado em tags chamado TestPowerShellGroup. O grupo inclui instâncias do Amazon EC2 na região atual que são marcadas com a chave de tag "Nome" e o valor da tag "test2". O comando retorna a consulta, o tipo de grupo e os resultados da operação.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
 [{"Key":"Name","Values":["test2"]}]} '
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

**Saída:**

```
Query
-----
Type
-----

{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":
 ["test2"]}]} TAG_FILTERS_1_0

LoggedAt      : 11/20/2018 2:40:59 PM
Group         : Amazon.ResourceGroups.Model.Group
ResourceQuery : Amazon.ResourceGroups.Model.ResourceQuery
```

```
Tags           : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength   : 338
HttpStatusCode  : OK
```

- Para obter detalhes da API, consulte [CreateGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-RGGroup

O código de exemplo a seguir mostra como usar Remove-RGGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove o grupo de recursos nomeado

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y

Description GroupArn
Name
-----
----
                arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-elbv2
non-tag-cfn-elbv2
```

- Para obter detalhes da API, consulte [DeleteGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-RGResourceTag

O código de exemplo a seguir mostra como usar Remove-RGResourceTag.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a tag mencionada do grupo de recursos

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes -Key Instances
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y

Arn                                                    Keys
---                                                    ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Para obter detalhes da API, consulte [Untag](#) in AWS Tools for PowerShell Cmdlet Reference.

## Update-RGGroup

O código de exemplo a seguir mostra como usar Update-RGGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo atualiza a descrição do grupo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Saída:

```
Description          GroupArn
Name
-----
----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes auto-yes
```

- Para obter detalhes da API, consulte [UpdateGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-RGGroupQuery

O código de exemplo a seguir mostra como usar Update-RGGroupQuery.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um objeto de consulta e atualiza a consulta para o grupo.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
    ResourceTypeFilters = @('AWS::EC2::Instance')
    TagFilters = @( @{
        Key='Environment'
        Values='Build600.11'
    })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Saída:

```
GroupName ResourceQuery
-----
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Para obter detalhes da API, consulte [UpdateGroupQuery](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos da API de marcação de Resource Groups usando o Tools for PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS Tools for PowerShell with Resource Groups Tagging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-RGTResourceTag

O código de exemplo a seguir mostra como usar Add-RGTResourceTag.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona as chaves de tag “stage” e “version” com os valores “beta” e “preprod\_test” a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Uma única chamada é feita para o serviço para aplicar as tags.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
"version"="preprod_test" }
```

Exemplo 2: Este exemplo adiciona as tags e os valores especificados a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- Para obter detalhes da API, consulte [TagResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGTResource

O código de exemplo a seguir mostra como usar `Get-RGTResource`.

### Ferramentas para PowerShell

Exemplo 1: retorna todos os recursos marcados em uma região e as chaves de tag associadas ao recurso. Se nenhum parâmetro `-Region` for fornecido ao cmdlet, ele tentará inferir a região a partir do shell ou dos metadados da instância EC2.

```
Get-RGTResource
```

Saída:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Exemplo 2: retorna todos os recursos marcados do tipo especificado em uma região. A string para cada nome de serviço e tipo de recurso é a mesma incorporada no Amazon Resource Name (ARN) de um recurso.

```
Get-RGTResource -ResourceType "s3"
```

Saída:

ResourceARN	Tags
-----	----
arn:aws:s3:::mybucket	{stage, version,
othertag}	

Exemplo 3: retorna todos os recursos marcados do tipo especificado em uma região. Observe que quando os tipos de recursos são canalizados para o cmdlet, uma chamada para o serviço é feita para cada tipo de recurso fornecido.

```
"dynamodb","s3" | Get-RGTResource
```

Saída:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version, othertag}

Exemplo 4: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Saída:

ResourceARN	Tags
-----	----
arn:aws:s3:::mybucket	{stage, version, othertag}

Exemplo 5: retorna todos os recursos marcados que correspondem ao filtro e ao tipo de recurso especificados.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Saída:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

Exemplo 6: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Saída:

ResourceARN	Tags
-------------	------



```
-----  
arn:aws:dynamodb:us-west-2:123456789012:table/mytable  
-----  
{stage, version}
```

- Para obter detalhes da API, consulte [GetResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGTTagKey

O código de exemplo a seguir mostra como usar Get-RGTTagKey.

### Ferramentas para PowerShell

Exemplo 1: retorna todas as chaves de tag na região especificada. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região a partir da região padrão do shell ou dos metadados da instância EC2. Observe que as chaves da tag não são retornadas em nenhuma ordem específica.

```
Get-RGTTagKey -region us-west-2
```

Saída:

```
version  
stage
```

- Para obter detalhes da API, consulte [GetTagKeys](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-RGTTagValue

O código de exemplo a seguir mostra como usar Get-RGTTagValue.

### Ferramentas para PowerShell

Exemplo 1: retorna o valor da tag especificada em uma região. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região a partir da região padrão do shell ou dos metadados da instância EC2.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Saída:

beta

- Para obter detalhes da API, consulte [GetTagValues](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-RGTResourceTag

O código de exemplo a seguir mostra como usar Remove-RGTResourceTag.

### Ferramentas para PowerShell

Exemplo 1: remove as chaves de tag “estágio” e “versão” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de remover as tags. Antes que as tags sejam removidas, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::mybucket"  
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"  
  
Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Exemplo 2: remove as chaves de tag “estágio” e “versão” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet. Antes de cada chamada, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::mybucket"  
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"  
  
$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Para obter detalhes da API, consulte [UntagResources](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Route 53 usando Ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Route 53.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### **Edit-R53ResourceRecordSet**

O código de exemplo a seguir mostra como usar `Edit-R53ResourceRecordSet`.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um registro A para `www.example.com` e altera o registro A para `test.example.com` de `192.0.2.3` para `192.0.2.1`. Observe que os valores dos registros do tipo TXT de alterações devem estar entre aspas duplas. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o `Get-R53Change` cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
```

```
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
    and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
    ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Exemplo 2: Este exemplo mostra como criar conjuntos de registros de recursos de alias. 'Z22222222' é o ID da zona hospedada do Amazon Route 53 na qual você está criando o conjunto de registros do recurso alias. 'exemplo.com' é o ápice da zona para o qual você deseja criar um alias e 'www.exemplo.com' é um subdomínio para o qual você também deseja criar um alias. 'Z111111111111111' é um exemplo de ID de zona hospedada para o balanceador de carga e 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' é um exemplo de nome de domínio do balanceador de carga com o qual o Amazon Route 53 responde às consultas de exemplo.com e www.example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
```

```

$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z11111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
    ChangeBatch_Comment="This change batch creates two alias resource record sets, one
for the zone apex, example.com, and one for www.example.com, that both point to
example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

Exemplo 3: Este exemplo cria dois registros A para `www.example.com`. Um quarto das vezes ( $1/(1+3)$ ), o Amazon Route 53 responde às consultas de `www.example.com` com os dois valores do primeiro conjunto de registros de recursos (192.0.2.9 e 192.0.2.10). Três quartos das vezes ( $3/(1+3)$ ) O Amazon Route 53 responde às consultas de `www.example.com` com os dois valores para o segundo conjunto de registros de recursos (192.0.2.11 e 192.0.2.12). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o `Get-R53Change` cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```

$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet

```

```

$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets, each
of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

Exemplo 4: Este exemplo mostra como criar conjuntos de registros de recursos de alias ponderados, supondo que example.com seja o domínio para o qual você deseja criar conjuntos de registros de recursos de alias ponderados. SetIdentifier diferencia os dois conjuntos de registros de recursos de alias ponderados um do outro. Esse elemento é necessário porque os elementos Nome e Tipo têm os mesmos valores para os dois conjuntos de registros de recursos. Z111111111111 e Z3333333333333333 são exemplos de IDs de zona hospedada para o balanceador de carga ELB especificado pelo valor de DNSName. example-load-balancer-22222222.us-east-1.elb.amazonaws.com e example-load-balancer-4444444444.us-east-1.elb.amazonaws.com são exemplos de domínios do Elastic Load Balancing dos quais o Amazon Route 53 responde a consultas de example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```

$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."

```

```

$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z555555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
record sets. Amazon Route 53 responds to queries for example.com with the first ELB
domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params

```

Exemplo 5: Este exemplo cria dois conjuntos de registros de recursos de alias de latência, um para um balanceador de carga ELB na região Oeste dos EUA (Oregon) (us-west-2) e outro para um balanceador de carga na região Ásia-Pacífico (Cingapura) (ap-southeast-1). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```

$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

```

```
$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Para obter detalhes da API, consulte [ChangeResourceRecordSets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53AccountLimit

O código de exemplo a seguir mostra como usar `Get-R53AccountLimit`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna o número máximo de zonas hospedadas que podem ser criadas usando a conta atual.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Saída:

```
15
```



- Para obter detalhes da API, consulte [GetAccountLimit](#)em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53CheckerIpRanges

O código de exemplo a seguir mostra como usar Get-R53CheckerIpRanges.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna os CIDRs para os verificadores de saúde do Route53

```
Get-R53CheckerIpRanges
```

Saída:

```
15.177.2.0/23
15.177.6.0/23
15.177.10.0/23
15.177.14.0/23
15.177.18.0/23
15.177.22.0/23
15.177.26.0/23
15.177.30.0/23
15.177.34.0/23
15.177.38.0/23
15.177.42.0/23
15.177.46.0/23
15.177.50.0/23
15.177.54.0/23
15.177.58.0/23
15.177.62.0/23
54.183.255.128/26
54.228.16.0/26
54.232.40.64/26
54.241.32.64/26
54.243.31.192/26
54.244.52.192/26
54.245.168.0/26
54.248.220.0/26
54.250.253.192/26
54.251.31.128/26
54.252.79.128/26
```

```
54.252.254.192/26
54.255.254.192/26
107.23.255.0/26
176.34.159.192/26
177.71.207.128/26
```

- Para obter detalhes da API, consulte [GetCheckerIpRanges](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53HostedZone

O código de exemplo a seguir mostra como usar Get-R53HostedZone.

### Ferramentas para PowerShell

Exemplo 1: Retorna detalhes da zona hospedada com a ID Z1D633PJN98FT9.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Para obter detalhes da API, consulte [GetHostedZone](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53HostedZoneCount

O código de exemplo a seguir mostra como usar Get-R53HostedZoneCount.

### Ferramentas para PowerShell

Exemplo 1: retorna o número total de zonas hospedadas públicas e privadas da atual Conta da AWS.

```
Get-R53HostedZoneCount
```

- Para obter detalhes da API, consulte [GetHostedZoneCount](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53HostedZoneLimit

O código de exemplo a seguir mostra como usar Get-R53HostedZoneLimit.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna o limite do número máximo de registros que podem ser criados na zona hospedada especificada.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Saída:

```
5
```

- Para obter detalhes da API, consulte [GetHostedZoneLimit](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53HostedZoneList

O código de exemplo a seguir mostra como usar Get-R53HostedZoneList.

## Ferramentas para PowerShell

Exemplo 1: gera todas as suas zonas hospedadas públicas e privadas.

```
Get-R53HostedZoneList
```

Exemplo 2: Exibe todas as zonas hospedadas associadas ao conjunto de delegações reutilizáveis que tem a ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Para obter detalhes da API, consulte [ListHostedZones](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53HostedZonesByName

O código de exemplo a seguir mostra como usar Get-R53HostedZonesByName.

## Ferramentas para PowerShell

Exemplo 1: retorna todas as suas zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio.

```
Get-R53HostedZonesByName
```

Exemplo 2: retorna suas zonas hospedadas públicas e privadas, em ordem ASCII por nome de domínio, começando pelo nome DNS especificado.

```
Get-R53HostedZonesByName -DnsName example2.com
```

Exemplo 3: Este exemplo mostra como enumerar manualmente as zonas hospedadas recuperando primeiro um único item e depois iterando dois por vez até que todas as zonas tenham sido retornadas, usando propriedades de marcador anexadas à resposta do serviço na pilha após cada chamada. **\$AWSHistory**

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- Para obter detalhes da API, consulte [ListHostedZonesByName](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53QueryLoggingConfigList

O código de exemplo a seguir mostra como usar `Get-R53QueryLoggingConfigList`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo retorna todas as configurações do registro de consultas DNS associadas à atual. Conta da AWS

```
Get-R53QueryLoggingConfigList
```

Saída:

Id	HostedZoneId	CloudWatchLogsLogGroupArn
----	--------------	---------------------------

```
--
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Para obter detalhes da API, consulte [ListQueryLoggingConfigs](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-R53ReusableDelegationSet

O código de exemplo a seguir mostra como usar `Get-R53ReusableDelegationSet`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera informações sobre o conjunto de delegações especificado, incluindo os quatro servidores de nomes atribuídos ao conjunto de delegações.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Saída:

```
Id                CallerReference NameServers
--                -
/delegationset/N23DS9X4AYEAAA testcaller      {ns-545.awsdns-04.net,
ns-1264.awsdns-30.org, ns-2004.awsdns-58.co.uk, ns-240.awsdns-30.com}
```

- Para obter detalhes da API, consulte [GetReusableDelegationSet](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-R53HostedZone

O código de exemplo a seguir mostra como usar `New-R53HostedZone`.

### Ferramentas para PowerShell

Exemplo 1: cria uma nova zona hospedada chamada 'example.com', associada a um conjunto de delegações reutilizável. Observe que você deve fornecer um valor para o `CallerReference`

parâmetro para que as solicitações precisem ser repetidas, se necessário, sem o risco de executar a operação duas vezes. Como a zona hospedada está sendo criada em uma VPC, ela é automaticamente privada e você não deve definir o parâmetro - `HostedZoneConfig_PrivateZone`.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Para obter detalhes da API, consulte [CreateHostedZone](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-R53QueryLoggingConfig

O código de exemplo a seguir mostra como usar `New-R53QueryLoggingConfig`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma nova configuração de registro de consultas DNS do Route53 para a zona hospedada especificada. O Amazon Route53 publicará registros de consulta de DNS no grupo de registros do Cloudwatch especificado.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

### Saída:

```
QueryLoggingConfig          Location
-----
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- Para obter detalhes da API, consulte [CreateQueryLoggingConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-R53ReusableDelegationSet

O código de exemplo a seguir mostra como usar `New-R53ReusableDelegationSet`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria um conjunto de delegação reutilizável de 4 servidores de nomes que podem ser reutilizados por várias zonas hospedadas.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Saída:

```
DelegationSet          Location
-----
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- Para obter detalhes da API, consulte [CreateReusableDelegationSet](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-R53VPCWithHostedZone

O código de exemplo a seguir mostra como usar `Register-R53VPCWithHostedZone`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo associa a VPC especificada à zona hospedada privada.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Saída:

```
Id          Status SubmittedAt          Comment
--          -
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- Para obter detalhes da API, consulte [AssociateVPC WithHostedZone em AWS Tools for PowerShell Cmdlet Reference](#).

## Remove-R53HostedZone

O código de exemplo a seguir mostra como usar Remove-R53HostedZone.

### Ferramentas para PowerShell

Exemplo 1: Exclui a zona hospedada com o ID especificado. Você será solicitado a confirmar antes que o comando continue, a menos que você adicione o parâmetro -Force switch.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Para obter detalhes da API, consulte [DeleteHostedZone](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-R53QueryLoggingConfig

O código de exemplo a seguir mostra como usar Remove-R53QueryLoggingConfig.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo remove a configuração especificada para o registro de consultas DNS.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Para obter detalhes da API, consulte [DeleteQueryLoggingConfig](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-R53ReusableDelegationSet

O código de exemplo a seguir mostra como usar Remove-R53ReusableDelegationSet.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o conjunto de delegações reutilizáveis especificado.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Para obter detalhes da API, consulte [DeleteReusableDelegationSet](#) em Referência de AWS Tools for PowerShell cmdlet.



## Unregister-R53VPCFromHostedZone

O código de exemplo a seguir mostra como usar `Unregister-R53VPCFromHostedZone`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo dissocia a VPC especificada da zona hospedada privada.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
-VPC_VPCRegion us-east-1
```

Saída:

Id	Status	SubmittedAt	Comment
--	-----	-----	-----
/change/C2XFCAAAA9HKZG	PENDING	01/28/2020 10:35:55	

- Para obter detalhes da API, consulte [DisassociateVPC em Cmdlet Reference FromHostedZone](#).AWS Tools for PowerShell

## Update-R53HostedZoneComment

O código de exemplo a seguir mostra como usar `Update-R53HostedZoneComment`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando atualiza o comentário para a zona hospedada especificada.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
zone"
```

Saída:

```
Id                : /hostedzone/Z385PDS6AAAAAR
Name              : example.com.
CallerReference   : C5B55555-7147-EF04-8341-69131E805C89
Config            : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService     :
```

- Para obter detalhes da API, consulte [UpdateHostedZoneComment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon S3 usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon S3.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### Copy-S3Object

O código de exemplo a seguir mostra como usar Copy-S3Object.

#### Ferramentas para PowerShell

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
```

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt  
-DestinationBucket backup-files
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip.

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:\downloads
```

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- Para obter detalhes da API, consulte [CopyObject](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3ACL

O código de exemplo a seguir mostra como usar Get-S3ACL.

### Ferramentas para PowerShell

Exemplo 1: O comando obtém os detalhes do proprietário do objeto do S3.

```
Get-S3ACL -BucketName 's3casetestbucket' -key 'initialize.ps1' -Select  
AccessControlList.Owner
```

Saída:

```
DisplayName Id  
----- --  
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obter detalhes da API, consulte [GetACL](#) em AWS Tools for PowerShell Cmdlet Reference.

## Get-S3Bucket

O código de exemplo a seguir mostra como usar `Get-S3Bucket`.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna todos os buckets do S3.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado “test-files”.

```
Get-S3Bucket -BucketName test-files
```

- Para obter detalhes da API, consulte [ListBuckets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketAccelerateConfiguration

O código de exemplo a seguir mostra como usar `Get-S3BucketAccelerateConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor `Habilitado`.

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

Saída:

```
Value  
-----  
Enabled
```

- Para obter detalhes da API, consulte [GetBucketAccelerateConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketAnalyticsConfiguration

O código de exemplo a seguir mostra como usar `Get-S3BucketAnalyticsConfiguration`.

## Ferramentas para PowerShell

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- Para obter detalhes da API, consulte [GetBucketAnalyticsConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketAnalyticsConfigurationList

O código de exemplo a seguir mostra como usar Get-S3BucketAnalyticsConfigurationList.

## Ferramentas para PowerShell

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- Para obter detalhes da API, consulte [ListBucketAnalyticsConfigurations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketEncryption

O código de exemplo a seguir mostra como usar Get-S3BucketEncryption.

## Ferramentas para PowerShell

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- Para obter detalhes da API, consulte [GetBucketEncryption](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketInventoryConfiguration

O código de exemplo a seguir mostra como usar `Get-S3BucketInventoryConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testinventory'
```

- Para obter detalhes da API, consulte [GetBucketInventoryConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketInventoryConfigurationList

O código de exemplo a seguir mostra como usar `Get-S3BucketInventoryConfigurationList`.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [ListBucketInventoryConfigurations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketLocation

O código de exemplo a seguir mostra como usar `Get-S3BucketLocation`.

### Ferramentas para PowerShell

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket “s3testbucket”.

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

**Saída:**

```
Value
-----
ap-south-1
```

- Para obter detalhes da API, consulte [GetBucketLocation](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-S3BucketLogging**

O código de exemplo a seguir mostra como usar `Get-S3BucketLogging`.

**Ferramentas para PowerShell**

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

**Saída:**

```
TargetBucketName  Grants TargetPrefix
-----
testbucket1       {}      testprefix
```

- Para obter detalhes da API, consulte [GetBucketLogging](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-S3BucketMetricsConfiguration**

O código de exemplo a seguir mostra como usar `Get-S3BucketMetricsConfiguration`.

**Ferramentas para PowerShell**

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId 'testfilter'
```

- Para obter detalhes da API, consulte [GetBucketMetricsConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketNotification

O código de exemplo a seguir mostra como usar `Get-S3BucketNotification`.

### Ferramentas para PowerShell

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão.

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty  
TopicConfigurations
```

Saída:

```
Id      Topic  
--      -  
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obter detalhes da API, consulte [GetBucketNotification](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketPolicy

O código de exemplo a seguir mostra como usar `Get-S3BucketPolicy`.

### Ferramentas para PowerShell

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [GetBucketPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketPolicyStatus

O código de exemplo a seguir mostra como usar `Get-S3BucketPolicyStatus`.



## Ferramentas para PowerShell

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- Para obter detalhes da API, consulte [GetBucketPolicyStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketReplication

O código de exemplo a seguir mostra como usar Get-S3BucketReplication.

## Ferramentas para PowerShell

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado “mybucket”.

```
Get-S3BucketReplication -BucketName mybucket
```

- Para obter detalhes da API, consulte [GetBucketReplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketRequestPayment

O código de exemplo a seguir mostra como usar Get-S3BucketRequestPayment.

## Ferramentas para PowerShell

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado “mybucket”. Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- Para obter detalhes da API, consulte [GetBucketRequestPayment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketTagging

O código de exemplo a seguir mostra como usar Get-S3BucketTagging.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- Para obter detalhes da API, consulte [GetBucketTagging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketVersioning

O código de exemplo a seguir mostra como usar Get-S3BucketVersioning.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [GetBucketVersioning](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3BucketWebsite

O código de exemplo a seguir mostra como usar Get-S3BucketWebsite.

### Ferramentas para PowerShell

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [GetBucketWebsite](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3CORSConfiguration

O código de exemplo a seguir mostra como usar `Get-S3CORSConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna um objeto que contém todas as regras de configuração do CORS correspondentes ao determinado S3 Bucket.

```
Get-S3CORSConfiguration -BucketName 's3testbucket' -Select Configuration.Rules
```

Saída:

```
AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example1.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example2.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {GET}
AllowedOrigins : {*}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {}
```

- Para obter detalhes da API, consulte [GetCorsConfiguration](#) em AWS Tools for PowerShell Cmdlet Reference.

## Get-S3LifecycleConfiguration

O código de exemplo a seguir mostra como usar `Get-S3LifecycleConfiguration`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo recupera a configuração do ciclo de vida do bucket.

```
Get-S3LifecycleConfiguration -BucketName test-bla
```

Saída:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Para obter detalhes da API, consulte [GetLifecycleConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3Object

O código de exemplo a seguir mostra como usar Get-S3Object.

## Ferramentas para PowerShell

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName test-files
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName test-files -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- Para obter detalhes da API, consulte [ListObjects](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3ObjectLockConfiguration

O código de exemplo a seguir mostra como usar `Get-S3ObjectLockConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 's3bucketteesting' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value  
-----  
Enabled
```

- Para obter detalhes da API, consulte [GetObjectLockConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3ObjectMetadata

O código de exemplo a seguir mostra como usar `Get-S3ObjectMetadata`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna os metadados do objeto com a chave 'ListTrusts.txt' no bucket S3 fornecido.

```
Get-S3ObjectMetadata -BucketName 's3testbucket' -Key 'ListTrusts.txt'
```

Saída:

```
Headers                : Amazon.S3.Model.HeadersCollection  
Metadata               : Amazon.S3.Model.MetadataCollection  
DeleteMarker           :  
AcceptRanges           : bytes  
ContentRange           :  
Expiration              :  
RestoreExpiration      :
```

```

RestoreInProgress           : False
LastModified                : 01/01/2020 08:02:05
ETag                       : "d000011112a222e333e3bb4ee5d43d21"
MissingMeta                 : 0
VersionId                  : null
Expires                    : 01/01/0001 00:00:00
WebsiteRedirectLocation    :
ServerSideEncryptionMethod : AES256
ServerSideEncryptionCustomerMethod :
ServerSideEncryptionKeyManagementServiceKeyId :
ReplicationStatus         :
PartsCount                 :
ObjectLockLegalHoldStatus :
ObjectLockMode             :
ObjectLockRetainUntilDate : 01/01/0001 00:00:00
StorageClass               :
RequestCharged             :

```

- Para obter detalhes da API, consulte [GetObjectMetadata](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3ObjectRetention

O código de exemplo a seguir mostra como usar `Get-S3ObjectRetention`.

### Ferramentas para PowerShell

Exemplo 1: o comando retorna o modo e a data até a qual o objeto ficará retido.

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- Para obter detalhes da API, consulte [GetObjectRetention](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3ObjectTagSet

O código de exemplo a seguir mostra como usar `Get-S3ObjectTagSet`.

### Ferramentas para PowerShell

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

Saída:

```
Key Value
---
test value
```

- Para obter detalhes da API, consulte [GetObjectTagging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3PreSignedURL

O código de exemplo a seguir mostra como usar Get-S3PreSignedURL.

Ferramentas para PowerShell

Exemplo 1: O comando retorna um URL pré-assinado para uma chave especificada e uma data de expiração.

```
Get-S3PreSignedURL -BucketName 's3testbucket' -Key 'testkey' -Expires '2023-11-16'
```

Exemplo 2: O comando retorna uma URL pré-assinada para um bucket de diretório com a chave especificada e uma data de expiração.

```
[Amazon.AWSCfgsS3]::UseSignatureVersion4 = $true
Get-S3PreSignedURL -BucketName sampledirectorybucket--use1-az5--x-s3 -Key
'testkey' -Expire '2023-11-17'
```

- Para obter detalhes da API, consulte [GetPreSignedURL na Referência do AWS Tools for PowerShell](#) Cmdlet.

## Get-S3PublicAccessBlock

O código de exemplo a seguir mostra como usar Get-S3PublicAccessBlock.

Ferramentas para PowerShell

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [GetPublicAccessBlock](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-S3Version

O código de exemplo a seguir mostra como usar `Get-S3Version`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna os metadados sobre todas as versões dos objetos em um determinado bucket do S3.

```
Get-S3Version -BucketName 's3testbucket'
```

Saída:

```
IsTruncated      : False
KeyMarker        :
VersionIdMarker  :
NextKeyMarker    :
NextVersionIdMarker :
Versions         : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
  ListTrusts.json}
Name             : s3testbucket
Prefix          :
MaxKeys         : 1000
CommonPrefixes  : {}
Delimiter       :
```

- Para obter detalhes da API, consulte [ListVersions](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-S3Bucket

O código de exemplo a seguir mostra como usar `New-S3Bucket`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando cria um novo bucket privado chamado “sample-bucket”.



```
New-S3Bucket -BucketName sample-bucket
```

Exemplo 2: Esse comando cria um novo bucket chamado “sample-bucket” com permissões de leitura e gravação.

```
New-S3Bucket -BucketName sample-bucket -PublicReadWrite
```

Exemplo 3: Esse comando cria um novo bucket chamado “sample-bucket” com permissões somente de leitura.

```
New-S3Bucket -BucketName sample-bucket -PublicReadOnly
```

Exemplo 4: Esse comando cria um novo bucket de diretório chamado “samplebucket--use1-az5--x-s3” com. PutBucketConfiguration

```
$bucketConfiguration = @{
    BucketInfo = @{
        DataRedundancy = 'SingleAvailabilityZone'
        Type = 'Directory'
    }
    Location = @{
        Name = 'use1-az5'
        Type = 'AvailabilityZone'
    }
}
New-S3Bucket -BucketName samplebucket--use1-az5--x-s3 -BucketConfiguration
$bucketConfiguration -Region us-east-1
```

- Para obter detalhes da API, consulte [PutBucket](#) em Referência de AWS Tools for PowerShell cmdlet.

## Read-S3Object

O código de exemplo a seguir mostra como usar Read-S3Object.

### Ferramentas para PowerShell

Exemplo 1: este comando recupera o item “sample.txt” do bucket “test-files” e o salva em um arquivo chamado “local-sample.txt” no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

Exemplo 2: este comando recupera o diretório virtual “DIR” do bucket “test-files” e o salva em uma pasta chamada “Local-DIR” no local atual. O arquivo “Local-DIR” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em “.json” de buckets com “config” no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Para obter detalhes da API, consulte [GetObject](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3Bucket

O código de exemplo a seguir mostra como usar Remove-S3Bucket.

### Ferramentas para PowerShell

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- Para obter detalhes da API, consulte [DeleteBucket](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketAnalyticsConfiguration

O código de exemplo a seguir mostra como usar Remove-S3BucketAnalyticsConfiguration.

## Ferramentas para PowerShell

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- Para obter detalhes da API, consulte [DeleteBucketAnalyticsConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketEncryption

O código de exemplo a seguir mostra como usar Remove-S3BucketEncryption.

## Ferramentas para PowerShell

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

Saída:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on  
target "s3casetestbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteBucketEncryption](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketInventoryConfiguration

O código de exemplo a seguir mostra como usar Remove-S3BucketInventoryConfiguration.

## Ferramentas para PowerShell

Exemplo 1: Esse comando remove o inventário chamado 'testInventoryName' correspondente ao determinado bucket do S3.

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testInventoryName'
```

### Saída:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "s3testbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteBucketInventoryConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketMetricsConfiguration

O código de exemplo a seguir mostra como usar `Remove-S3BucketMetricsConfiguration`.

### Ferramentas para PowerShell

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId  
'testmetrics'
```

- Para obter detalhes da API, consulte [DeleteBucketMetricsConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketPolicy

O código de exemplo a seguir mostra como usar `Remove-S3BucketPolicy`.

### Ferramentas para PowerShell

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [DeleteBucketPolicy](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketReplication

O código de exemplo a seguir mostra como usar `Remove-S3BucketReplication`.

### Ferramentas para PowerShell

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado “mybucket”. Observe que essa operação requer permissão para a `DeleteReplicationConfiguration` ação s3. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção `-Force`.

```
Remove-S3BucketReplication -BucketName mybucket
```

- Para obter detalhes da API, consulte [DeleteBucketReplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketTagging

O código de exemplo a seguir mostra como usar `Remove-S3BucketTagging`.

### Ferramentas para PowerShell

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteBucketTagging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3BucketWebsite

O código de exemplo a seguir mostra como usar Remove-S3BucketWebsite.

### Ferramentas para PowerShell

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

#### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteBucketWebsite](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3CORSConfiguration

O código de exemplo a seguir mostra como usar Remove-S3CORSConfiguration.

### Ferramentas para PowerShell

Exemplo 1: Esse comando remove a configuração do CORS para um determinado bucket do S3.

```
Remove-S3CORSConfiguration -BucketName 's3testbucket'
```

#### Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
target "s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteCorsConfiguration](#) em Cmdlet Reference.AWS Tools for PowerShell

## Remove-S3LifecycleConfiguration

O código de exemplo a seguir mostra como usar Remove-S3LifecycleConfiguration.

### Ferramentas para PowerShell

Exemplo 1: O comando remove todas as regras de ciclo de vida de um determinado bucket do S3.

```
Remove-S3LifecycleConfiguration -BucketName 's3testbucket'
```

- Para obter detalhes da API, consulte [DeleteLifecycleConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3MultipartUpload

O código de exemplo a seguir mostra como usar Remove-S3MultipartUpload.

### Ferramentas para PowerShell

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January 02, 2014"
```

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às 10:45:37.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02 10:45:37"
```

- Para obter detalhes da API, consulte [AbortMultipartUpload](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3Object

O código de exemplo a seguir mostra como usar `Remove-S3Object`.

### Ferramentas para PowerShell

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros para chaves que não puderam ser processadas pelo serviço, adicione o `ReportErrorsOnly` parâmetro - (esse parâmetro também pode ser especificado com o alias `-Quiet`).

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Exemplo 4: Este exemplo usa uma expressão embutida com o `KeyCollection` parâmetro - para obter as chaves dos objetos a serem excluídos. `Get-S3Object` retorna uma coleção de instâncias `Amazon.S3.Model.S3Object`, cada uma com um membro-chave do tipo string identificando o objeto.

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-files" -  
KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```



Exemplo 5: este exemplo obtém todos os objetos que têm um prefixo de chave “prefix/subprefix” no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias - S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada para o serviço.

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-S3Object -Force
```

Exemplo 6: Este exemplo envia uma coleção de ObjectVersion instâncias do Amazon.S3.Model.S3 que representam marcadores de exclusão para o cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada para o serviço.

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: Esse script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro -KeyAndVersionCollection .

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
  {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
  $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- Para obter detalhes da API, consulte [DeleteObjects](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3ObjectTagSet

O código de exemplo a seguir mostra como usar Remove-S3ObjectTagSet.

## Ferramentas para PowerShell

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave "testfile.txt" no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select '^Key'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
"testfile.txt".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
testfile.txt
```

- Para obter detalhes da API, consulte [DeleteObjectTagging](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-S3PublicAccessBlock

O código de exemplo a seguir mostra como usar Remove-S3PublicAccessBlock.

## Ferramentas para PowerShell

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select '^BucketName'
```

Saída:

```
s3testbucket
```

- Para obter detalhes da API, consulte [DeletePublicAccessBlock](#) em Referência de AWS Tools for PowerShell cmdlet.

## Set-S3BucketEncryption

O código de exemplo a seguir mostra como usar Set-S3BucketEncryption.

### Ferramentas para PowerShell

Exemplo 1: este comando habilita a criptografia padrão do lado do servidor AES256 com chaves gerenciadas do Amazon S3 (SSE-S3) no bucket em questão.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 's3testbucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obter detalhes da API, consulte [PutBucketEncryption](#) em Referência de AWS Tools for PowerShell cmdlet.

## Test-S3Bucket

O código de exemplo a seguir mostra como usar Test-S3Bucket.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna True se o bucket existir, caso contrário, False. O comando retorna True mesmo que o bucket não pertença ao usuário.

```
Test-S3Bucket -BucketName test-files
```

- Para obter detalhes da API, consulte [Test-S3Bucket](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketAccelerateConfiguration

O código de exemplo a seguir mostra como usar Write-S3BucketAccelerateConfiguration.

### Ferramentas para PowerShell

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
```

```
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- Para obter detalhes da API, consulte [PutBucketAccelerateConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketNotification

O código de exemplo a seguir mostra como usar Write-S3BucketNotification.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo configura a configuração do tópico do SNS para o evento do S3 ObjectRemovedDelete e ativa a notificação para o determinado bucket do s3

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName kt-tools -TopicConfiguration $topic
```

Exemplo 2: Este exemplo habilita notificações ObjectCreatedAll para o determinado bucket, enviando-o para a função Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}
```

```
Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
$lambdaConfig
```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único comando.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
$firstLambdaConfig,$secondLambdaConfig
```

- Para obter detalhes da API, consulte [PutBucketNotification](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketReplication

O código de exemplo a seguir mostra como usar Write-S3BucketReplication.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo define uma configuração de replicação com uma única regra que permite a replicação para o bucket 'exampletargetbucket' de qualquer novo objeto criado com o prefixo de nome de chave "" no bucket 'examplebucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemplo 2: Este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket 'exampletargetbucket' de qualquer novo objeto criado com o prefixo de nome de chave "" ou "". TaxDocs OtherDocs Os prefixos de chave não devem se sobrepôr.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
```

```

    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

Exemplo 3: Este exemplo atualiza a configuração de replicação no bucket especificado para desativar a regra que controla a replicação de objetos com o prefixo de nome de chave "TaxDocs" no bucket 'exampletargetbucket'.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

- Para obter detalhes da API, consulte [PutBucketReplication](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketRequestPayment

O código de exemplo a seguir mostra como usar Write-S3BucketRequestPayment.

### Ferramentas para PowerShell

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado "mybucket" para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para definir o pagamento da solicitação de volta ao padrão, use 'BucketOwner' para o parâmetro RequestPaymentConfiguration\_Payer.

```
Write-S3BucketRequestPayment -BucketName mybucket -RequestPaymentConfiguration_Payer
Requester
```

- Para obter detalhes da API, consulte [PutBucketRequestPayment](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketTagging

O código de exemplo a seguir mostra como usar Write-S3BucketTagging.

### Ferramentas para PowerShell

Exemplo 1: este comando aplica duas tags a um bucket denominado **cloudtrail-test-2018** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **Get-S3BucketTagging -BucketName bucket\_name**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **Add-RGTResourceTag** e **Remove-RGTResourceTag**. Como alternativa, use o Editor de tags no console AWS de gerenciamento para gerenciar as tags de bucket do S3.

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **cloudtrail-test-2018** para o cmdlet **Write-S3BucketTagging**. Ele aplica as tags Stage:Production e Department:Finance ao bucket. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging -TagSet
@( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Para obter detalhes da API, consulte [PutBucketTagging](#) em Referência de AWS Tools for PowerShell cmdlet.



## Write-S3BucketVersioning

O código de exemplo a seguir mostra como usar Write-S3BucketVersioning.

### Ferramentas para PowerShell

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status Enabled
```

- Para obter detalhes da API, consulte [PutBucketVersioning](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3BucketWebsite

O código de exemplo a seguir mostra como usar Write-S3BucketWebsite.

### Ferramentas para PowerShell

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 's3testbucket' -  
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obter detalhes da API, consulte [PutBucketWebsite](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3LifecycleConfiguration

O código de exemplo a seguir mostra como usar Write-S3LifecycleConfiguration.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo grava/substitui a configuração fornecida no \$NewRule. Essa configuração garante limitar os objetos do escopo com determinados valores de prefixo e tag.

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{  
    Expiration = @{  
        Days= 50
```

```

}
Id = "Test-From-Write-cmdlet-1"
Filter= @{
LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]{
  Operands= @(
    [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" = "py"
    },
    [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"= @{
        "Key" = "non-use"
        "Value" = "yes"
      }
    }
  )
}
)
}
}
"Status"= 'Enabled'
NoncurrentVersionExpiration = @{
  NoncurrentDays = 75
}
}

Write-S3LifecycleConfiguration -BucketName my-review-scrap -Configuration_Rule
$NewRule

```

Exemplo 2: Este exemplo define várias regras com filtragem. \$ ArchiveRule define os objetos a serem arquivados em 30 dias no Glacier e 120 no. DeepArchive \$ ExpireRule expira as versões atual e anterior em 150 dias para objetos com prefixo 'py' e tag:key 'archieved' definida como 'sim'.

```

$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration = @{
    Days= 150
  }
  Id = "Remove-in-150-days"
  Filter= @{
    LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]{
      Operands= @(
        [Amazon.S3.Model.LifecyclePrefixPredicate] @{
          "Prefix" = "py"
        },
        [Amazon.S3.Model.LifecycleTagPredicate] @{

```

```
        "Tag"= @{
            "Key" = "archived"
            "Value" = "yes"
        }
    }
)
}
}
Status= 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 150
}
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = $null
    Id = "Archive-to-Glacier-in-30-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "reviewed"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    Status = 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 75
    }
    Transitions = @(
        @{
            Days = 30
            "StorageClass"= 'Glacier'
        },
        @{
            Days = 120
            "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
        }
    )
}
```

```
}  
)  
}  
  
Write-S3LifecycleConfiguration -BucketName my-review-scrap -Configuration_Rule  
$ExpireRule,$ArchiveRule
```

- Para obter detalhes da API, consulte [PutLifecycleConfiguration](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3Object

O código de exemplo a seguir mostra como usar Write-S3Object.

### Ferramentas para PowerShell

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S3.

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

Exemplo 3: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “prefix/to/sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-  
sample.txt
```

Exemplo 4: Esse comando carrega todos os arquivos no subdiretório “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename", onde 'filename' varia.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

Exemplo 5: Esse comando carrega todos os arquivos\*.ps1 no diretório local “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename.ps1", onde 'filename' varia.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave “sample.txt”.

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet @{{Key="key1";Value="value1"}},{Key="key2";Value="value2"}}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse -TagSet @{{Key="key1";Value="value1"}},{Key="key2";Value="value2"}}
```

- Para obter detalhes da API, consulte [PutObject](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-S3ObjectRetention

O código de exemplo a seguir mostra como usar Write-S3ObjectRetention.

### Ferramentas para PowerShell

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obter detalhes da API, consulte [PutObjectRetention](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do S3 Glacier usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o S3 Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Get-GLCJob

O código de exemplo a seguir mostra como usar Get-GLCJob.

### Ferramentas para PowerShell

Exemplo 1: Retorna detalhes do trabalho especificado. Quando o trabalho é concluído com êxito, o JobOutput cmdlet Read-GC pode ser usado para recuperar o conteúdo do trabalho (um arquivo ou uma lista de inventário) para o sistema de arquivos local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Saída:

```
Action : ArchiveRetrieval
```

```

ArchiveId                : o909j...X-TpIhQJw
ArchiveSHA256TreeHash   : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes      : 38034480
Completed                : False
CompletionDate          : 1/1/0001 12:00:00 AM
CreationDate            : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes    : 0
JobDescription           :
JobId                   : op1x...JSbthM
JobOutputPath           :
OutputLocation          :
RetrievalByteRange      : 0-38034479
SelectParameters        :
SHA256TreeHash          : 79f3ea754c02f58...dc57bf4395b
SNSTopic                :
StatusCode               : InProgress
StatusMessage           :
Tier                     : Standard
VaultARN                : arn:aws:glacier:us-west-2:012345678912:vaults/test

```

- Para obter detalhes da API, consulte [DescribeJob](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-GLCVault

O código de exemplo a seguir mostra como usar `New-GLCVault`.

### Ferramentas para PowerShell

Exemplo 1: cria um novo cofre para a conta do usuário. Como nenhum valor foi fornecido ao `AccountId` parâmetro -, os cmdlets usam o padrão "-" indicando a conta atual.

```
New-GLCVault -VaultName myvault
```

Saída:

```
/01234567812/vaults/myvault
```

- Para obter detalhes da API, consulte [CreateVault](#) em Referência de AWS Tools for PowerShell cmdlet.

## Read-GLCJobOutput

O código de exemplo a seguir mostra como usar Read-GLCJobOutput.

### Ferramentas para PowerShell

Exemplo 1: baixa o conteúdo do arquivamento que foi agendado para recuperação na tarefa especificada e armazena o conteúdo em um arquivo no disco. O download valida a soma de verificação para você, se houver uma disponível. Se necessário, a soma de verificação pode ser obtida do histórico de respostas do serviço da seguinte forma (supondo que esse cmdlet tenha sido o último executado):. **\$AWSHistory.LastServiceResponse** Se o cmdlet não foi executado mais recentemente, inspecione a **\$AWSHistory.Commands** coleção para obter a resposta de serviço relevante.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp\blue.bin"
```

- Para obter detalhes da API, consulte [GetJobOutput](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-GLCJob

O código de exemplo a seguir mostra como usar Start-GLCJob.

### Ferramentas para PowerShell

Exemplo 1: inicia um trabalho para recuperar um arquivo do cofre especificado de propriedade do usuário. O status do trabalho pode ser verificado usando o cmdlet Get-GlcJob. Quando o trabalho é concluído com êxito, o JobOutput cmdlet Read-GC pode ser usado para recuperar o conteúdo do arquivamento no sistema de arquivos local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription "archive retrieval" -ArchiveId "o909j...TX-TpIhQJw"
```

Saída:

```
JobId                JobOutputPath Location
-----                -
op1x...JSbthM        /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```



- Para obter detalhes da API, consulte [InitiateJob](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-GLCArchive

O código de exemplo a seguir mostra como usar Write-GLCArchive.

### Ferramentas para PowerShell

Exemplo 1: carrega um único arquivo no cofre especificado, retornando o ID do arquivo e a soma de verificação computada.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Saída:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b

Exemplo 2: Carrega o conteúdo de uma hierarquia de pastas para o cofre especificado na conta do usuário. Para cada arquivo carregado, o cmdlet emite o nome do arquivo, o ID do arquivo correspondente e a soma de verificação computada do arquivo.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Saída:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b
C:\temp\green.bin	qXAf0dSG...czo729UHXrw	d50a1...9184b9
C:\temp\lum.bin	39aNifP3...q9nb8nZkFIg	28886...5c3e27
C:\temp\red.bin	vp7E6rU...Ejk_HhjAxKA	e05f7...4e34f5
C:\temp\Folder1\file1.txt	_eRINlip...5Sxy7dD2BaA	d0d2a...c8a3ba
C:\temp\Folder2\file2.iso	-Ix3jlm...iXiDh-Xf0PA	7469e...3e86f1

- Para obter detalhes da API, consulte [UploadArchive](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon SES usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Get-SESIentity**

O código de exemplo a seguir mostra como usar `Get-SESIentity`.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna uma lista contendo todas as identidades (endereços de e-mail e domínios) de uma AWS conta específica, independentemente do status da verificação.

```
Get-SESIentity
```

- Para obter detalhes da API, consulte [ListIdentities](#) em Referência de AWS Tools for PowerShell cmdlet.

#### **Get-SESSendQuota**

O código de exemplo a seguir mostra como usar `Get-SESSendQuota`.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna os limites de envio atuais do usuário.

### Get-SESSendQuota

- Para obter detalhes da API, consulte [GetSendQuota](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SESSendStatistic

O código de exemplo a seguir mostra como usar `Get-SESSendStatistic`.

### Ferramentas para PowerShell

Exemplo 1: Esse comando retorna as estatísticas de envio do usuário. O resultado é uma lista de pontos de dados, representando as duas últimas semanas da atividade de envio. Cada ponto de dados na lista contém estatísticas para um intervalo de 15 minutos.

### Get-SESSendStatistic

- Para obter detalhes da API, consulte [GetSendStatistics](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon SNS usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Publish-SNSMessage

O código de exemplo a seguir mostra como usar Publish-SNSMessage.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message  
"Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]  
@{DataType='String';  
StringValue ='AnyCity'}}
```

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$cityAttributeValue = New-Object  
Amazon.SimpleNotificationService.Model.MessageAttributeValue  
$cityAttributeValue.DataType = "String"  
$cityAttributeValue.StringValue = "AnyCity"  
  
$populationAttributeValue = New-Object  
Amazon.SimpleNotificationService.Model.MessageAttributeValue  
$populationAttributeValue.DataType = "Number"  
$populationAttributeValue.StringValue = "1250800"  
  
$messageAttributes = New-Object System.Collections.Hashtable  
$messageAttributes.Add("City", $cityAttributeValue)  
$messageAttributes.Add("Population", $populationAttributeValue)  
  
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message  
"Hello" -MessageAttribute $messageAttributes
```

- Para obter detalhes da API, consulte [Publicar](#) no AWS Tools for PowerShell Cmdlet Reference.

## Exemplos do Amazon SQS usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

### Ações

#### **Add-SQSPermission**

O código de exemplo a seguir mostra como usar `Add-SQSPermission`.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label  
SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/  
MyQueue
```

- Para obter detalhes da API, consulte [AddPermission](#) em Referência de AWS Tools for PowerShell cmdlet.

#### **Clear-SQSQueue**

O código de exemplo a seguir mostra como usar `Clear-SQSQueue`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [PurgeQueue](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-SQSMessageVisibility

O código de exemplo a seguir mostra como usar `Edit-SQSMessageVisibility`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (10 horas \* 60 minutos \* 60 segundos = 36000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Para obter detalhes da API, consulte [ChangeMessageVisibility](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-SQSMessageVisibilityBatch

O código de exemplo a seguir mostra como usar `Edit-SQSMessageVisibilityBatch`.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo altera o tempo limite de visibilidade de duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de visibilidade da primeira mensagem é alterado para 10 horas (10 horas\* 60 minutos\* 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas\* 60 minutos\* 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object  
Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
```

```

$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
    Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessagesVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
    $changeVisibilityRequest2

```

Saída:

```

Failed      Successful
-----
{}          {Request2, Request1}

```

- Para obter detalhes da API, consulte [ChangeMessageVisibilityBatch](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SQSDeadLetterSourceQueue

O código de exemplo a seguir mostra como usar `Get-SQSDeadLetterSourceQueue`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os URLs de todas as filas que dependem da fila especificada como fila de letras mortas.

```

Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue

```

Saída:

```

https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue

```

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes da API, consulte [ListDeadLetterSourceQueues](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SQSQueue

O código de exemplo a seguir mostra como usar Get-SQSQueue.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: Este exemplo lista todas as filas que começam com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue  
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes da API, consulte [ListQueues](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SQSQueueAttribute

O código de exemplo a seguir mostra como usar Get-SQSQueueAttribute.



## Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp      : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                     :
  {"Version":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}},
{"Sid":
  "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}]}
Attributes                  : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                [ApproximateNumberOfMessagesNotVisible, 0],
                                [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemplo 2: Este exemplo lista separadamente somente os atributos especificados para a fila especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

**Saída:**

```

VisibilityTimeout           : 30
DelaySeconds                : 0
MaximumMessageSize         : 262144
MessageRetentionPeriod     : 345600
ApproximateNumberOfMessages : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed : 0
CreatedTimestamp           : 2/11/2015 5:53:35 PM
LastModifiedTimestamp      : 12/29/2015 2:23:17 PM
QueueARN                   : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                     :
  {"Version":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":
  "SendMessageFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}

```

- Para obter detalhes da API, consulte [GetQueueAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SQSQueueUrl**

O código de exemplo a seguir mostra como usar `Get-SQSQueueUrl`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

**Saída:**

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SQSQueue

O código de exemplo a seguir mostra como usar New-SQSQueue.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [CreateQueue](#) em Referência de AWS Tools for PowerShell cmdlet.

## Receive-SQSMessage

O código de exemplo a seguir mostra como usar Receive-SQSMessage.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName  
StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes           : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,  
1451495923744]}
```

```
Body : Information about John Doe's grade.
MD5ofBody : ea572796e3c231f974fe75d89EXAMPLE
MD5ofMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle : AQEBpfGp...20Q5cg==
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SQSMessage

O código de exemplo a seguir mostra como usar `Remove-SQSMessage`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a mensagem com o identificador de recebimento especificado da fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
-ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes da API, consulte [DeleteMessage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SQSMessageBatch

O código de exemplo a seguir mostra como usar `Remove-SQSMessageBatch`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui 2 mensagens com os identificadores de recebimento especificados da fila especificada.

```
$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
```

```
$deleteMessageRequest2.Id = "Request2"  
$deleteMessageRequest2.ReceiptHandle = "AQEBq0VY...KTsLYg=="  
  
Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/  
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Saída:

```
Failed      Successful  
-----  
{}
```

- Para obter detalhes da API, consulte [DeleteMessageBatch](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SQSPermission

O código de exemplo a seguir mostra como usar `Remove-SQSPermission`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo remove as configurações de permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-  
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [RemovePermission](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SQSQueue

O código de exemplo a seguir mostra como usar `Remove-SQSQueue`.

Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [DeleteQueue](#) em Referência de AWS Tools for PowerShell cmdlet.

## Send-SQSMessage

O código de exemplo a seguir mostra como usar Send-SQSMessage.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada, com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl https://
sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

### Saída:

MD5ofMessageAttributes	MD5ofMessageBody	MessageId
-----	-----	-----
1d3e51347bc042efbdf6dda31EXAMPLE c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- Para obter detalhes da API, consulte [SendMessage](#) em Referência de AWS Tools for PowerShell cmdlet.

## Send-SQSMessageBatch

O código de exemplo a seguir mostra como usar Send-SQSMessageBatch.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo envia 2 mensagens com os atributos e corpos de mensagens especificados para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

**Saída:**

```
Failed      Successful
-----
{}          {FirstMessage, SecondMessage}
```

- Para obter detalhes da API, consulte [SendMessageBatch](#) em Referência de AWS Tools for PowerShell cmdlet.

**Set-SQSQueueAttribute**

O código de exemplo a seguir mostra como usar Set-SQSQueueAttribute.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo mostra como definir uma política para inscrever uma fila em um tópico do SNS. Quando uma mensagem é publicada no tópico, uma mensagem é enviada para a fila inscrita.

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

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

# construct the policy and inject arns
$policy = @"
{
  "Version": "2008-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
```



```
"Condition": {
  "ArnEquals": {
    "aws:SourceArn": "$topicarn"
  }
}
]
}
"@

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

Exemplo 2: Este exemplo define os atributos especificados para a fila especificada.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
"131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte [SetQueueAttributes](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS STS exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### Convert-STSAuthorizationMessage

O código de exemplo a seguir mostra como usar `Convert-STSAuthorizationMessage`.

#### Ferramentas para PowerShell

Exemplo 1: decodifica as informações adicionais contidas no conteúdo da mensagem codificada fornecida que foi retornada em resposta a uma solicitação. As informações adicionais são codificadas porque os detalhes do status da autorização podem constituir informações privilegiadas que o usuário responsável por solicitar a ação não deve ver.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Para obter detalhes da API, consulte [DecodeAuthorizationMessage](#) em Referência de AWS Tools for PowerShell cmdlet.

### Get-STSFederationToken

O código de exemplo a seguir mostra como usar `Get-STSFederationToken`.

#### Ferramentas para PowerShell

Exemplo 1: solicita um token federado válido por uma hora usando "Bob" como nome do usuário federado. Esse nome pode ser usado para referenciar o nome do usuário federado em uma política baseada em recursos (como uma política de bucket do Amazon S3). A política do IAM fornecida, no formato JSON, é usada para definir o escopo das permissões que estão disponíveis para o usuário do IAM. A política fornecida não pode conceder mais permissões do que as concedidas ao usuário solicitante, com as permissões finais do usuário federado sendo o conjunto mais restritivo com base na interseção da política aprovada com a política de usuário do IAM.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds 3600
```

- Para obter detalhes da API, consulte [GetFederationToken](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-STSSessionToken

O código de exemplo a seguir mostra como usar `Get-STSSessionToken`.

### Ferramentas para PowerShell

Exemplo 1: retorna uma instância **Amazon.RuntimeAWSCredentials** contendo credenciais temporárias válidas por um determinado período. As credenciais usadas para solicitar credenciais temporárias são inferidas dos padrões atuais do shell. Para especificar outras credenciais, use os parâmetros `- ProfileName` ou `- AccessKey /-SecretKey`.

```
Get-STSSessionToken
```

Saída:

```
AccessKeyId                Expiration
SecretAccessKey            SessionToken
-----
-----
EXAMPLEACCESSKEYID        2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleTokenN.....
```

Exemplo 2: retorna uma instância **Amazon.RuntimeAWSCredentials** contendo credenciais temporárias válidas por uma hora. As credenciais usadas para fazer a solicitação são obtidas do perfil especificado.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Saída:

```
AccessKeyId                Expiration
SecretAccessKey            SessionToken
-----
-----
EXAMPLEACCESSKEYID        2/16/2015 9:12:28 PM
examplesecretaccesskey... SamPleTokenN.....
```

Exemplo 3: retorna uma instância **Amazon.RuntimeAWSCredentials** contendo credenciais temporárias válidas por uma hora usando o número de identificação do dispositivo de MFA

associado à conta cujas credenciais estão especificadas no perfil 'myprofile' e o valor fornecido pelo dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
YourMFADeviceSerialNumber -TokenCode 123456
```

Saída:

AccessKeyId	Expiration
SecretAccessKey	SessionToken
-----	-----
-----	-----
EXAMPLEACCESSKEYID	2/16/2015 9:12:28 PM
examplesecretaccesskey...	SamPleTokenN.....

- Para obter detalhes da API, consulte [GetSessionToken](#) em Referência de AWS Tools for PowerShell cmdlet.

## Use-STSRole

O código de exemplo a seguir mostra como usar Use-STSRole.

### Ferramentas para PowerShell

Exemplo 1: retorna um conjunto de credenciais temporárias (chave de acesso, chave secreta e token de sessão) que podem ser usadas por uma hora para acessar AWS recursos aos quais o usuário solicitante normalmente não teria acesso. As credenciais retornadas têm as permissões permitidas pela política de acesso do perfil assumido e pela política fornecida (não é possível usar a política fornecida para conceder permissões além das definidas pela política de acesso do perfil que está sendo assumido).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Exemplo 2: retorna um conjunto de credenciais temporárias, válidas por uma hora, que têm as mesmas permissões definidas na política de acesso do perfil que está sendo assumido.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Exemplo 3: retorna um conjunto de credenciais temporárias que fornecem o número de série e o token gerado de uma MFA associada às credenciais do usuário usadas para executar o cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Exemplo 4: retorna um conjunto de credenciais temporárias que assumiram um perfil definido em uma conta de cliente. Para cada função que o terceiro possa assumir, a conta do cliente deve criar uma função usando um identificador que deve ser passado no ExternalId parâmetro - sempre que a função for assumida.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Para obter detalhes da API, consulte [AssumeRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Use-STSWebIdentityRole

O código de exemplo a seguir mostra como usar Use-STSWebIdentityRole.

### Ferramentas para PowerShell

Exemplo 1: retorna um conjunto temporário de credenciais, válido por uma hora, para um usuário que foi autenticado com o provedor de identidade Login with Amazon. As credenciais assumem a política de acesso associada ao perfil identificado pelo ARN do perfil. Opcionalmente, você pode transmitir uma política JSON ao parâmetro -Policy que refina ainda mais as permissões de acesso (não é possível conceder mais permissões do que as disponíveis nas permissões associadas ao perfil). O valor fornecido ao -WebIdentityToken é o identificador de usuário exclusivo que foi retornado pelo provedor de identidade.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
-RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Para obter detalhes da API, consulte [AssumeRoleWithWebIdentity](#) em Referência de AWS Tools for PowerShell cmdlet.

# AWS Support exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS Support.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-ASACommunicationToCase

O código de exemplo a seguir mostra como usar Add-ASACommunicationToCase.

## Ferramentas para PowerShell

Exemplo 1: adiciona o corpo de uma comunicação por e-mail ao caso especificado.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CommunicationBody "Some text about the case"
```

Exemplo 2: adiciona o corpo de uma comunicação por e-mail ao caso especificado mais um ou mais endereços de e-mail contidos na linha CC do e-mail.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -  
CcEmailAddress @"email1@address.com", "email2@address.com") -CommunicationBody  
"Some text about the case"
```

- Para obter detalhes da API, consulte [AddCommunicationToCase](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASACase

O código de exemplo a seguir mostra como usar Get-ASACase.

### Ferramentas para PowerShell

Exemplo 1: retorna os detalhes de todos os casos de suporte.

```
Get-ASACase
```

Exemplo 2: retorna os detalhes de todos os casos de suporte desde a data e a hora especificadas.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Exemplo 3: retorna os detalhes dos primeiros 10 casos de suporte, incluindo aqueles que foram resolvidos.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Exemplo 4: retorna os detalhes do único caso de suporte especificado.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 5: retorna os detalhes dos casos de suporte especificados.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",  
"case-18929034710-2011-c4fdeabf33c5cf47")
```

Exemplo 6: retorna todos os casos de suporte usando paginação manual. As caixas são recuperadas em lotes de 20.

```
$nextToken = $null  
do {  
    Get-ASACase -NextToken $nextToken -MaxResult 20  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeCases](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASACommunication

O código de exemplo a seguir mostra como usar Get-ASACommunication.

### Ferramentas para PowerShell

Exemplo 1: Retorna todas as comunicações do caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 2: retorna todas as comunicações desde a meia-noite UTC de 1º de janeiro de 2012 para o caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime  
"2012-01-10T00:00Z"
```

Exemplo 3: retorna todas as comunicações desde a meia-noite UTC de 1º de janeiro de 2012 para o caso especificado, usando paginação manual. As comunicações são recuperadas em lotes de 20.

```
$nextToken = $null  
do {  
    Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken  
    $nextToken -MaxResult 20  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- Para obter detalhes da API, consulte [DescribeCommunications](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASAService

O código de exemplo a seguir mostra como usar Get-ASAService.

### Ferramentas para PowerShell

Exemplo 1: Retorna todos os códigos de serviço, nomes e categorias disponíveis.

```
Get-ASAService
```



Exemplo 2: retorna o nome e as categorias do serviço com o código especificado.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Exemplo 3: Retorna o nome e as categorias dos códigos de serviço especificados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Exemplo 4: retorna o nome e as categorias (em japonês) dos códigos de serviço especificados. Atualmente, os códigos de idioma inglês (“en”) e japonês (“ja”) são suportados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -  
Language "ja"
```

- Para obter detalhes da API, consulte [DescribeServices](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASASeverityLevel

O código de exemplo a seguir mostra como usar Get-ASASeverityLevel.

### Ferramentas para PowerShell

Exemplo 1: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support.

```
Get-ASASeverityLevel
```

Exemplo 2: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support. Os nomes dos níveis são retornados em japonês.

```
Get-ASASeverityLevel -Language "ja"
```

- Para obter detalhes da API, consulte [DescribeSeverityLevels](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASATrustedAdvisorCheck

O código de exemplo a seguir mostra como usar Get-ASATrustedAdvisorCheck.

## Ferramentas para PowerShell

Exemplo 1: Retorna a coleção de cheques do Trusted Advisor. Você deve especificar o parâmetro `Language`, que pode aceitar “en” para saída em inglês ou “ja” para saída em japonês.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Para obter detalhes da API, consulte [DescribeTrustedAdvisorChecks](#) em Referência de AWS Tools for PowerShell cmdlet.

### **Get-ASATrustedAdvisorCheckRefreshStatus**

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckRefreshStatus`.

## Ferramentas para PowerShell

Exemplo 1: retorna o status atual das solicitações de atualização para as verificações especificadas. O `Request-ASA TrustedAdvisorCheckRefresh` pode ser usado para solicitar que as informações de status das verificações sejam atualizadas.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte [DescribeTrustedAdvisorCheckRefreshStatuses](#) em Referência de AWS Tools for PowerShell cmdlet.

### **Get-ASATrustedAdvisorCheckResult**

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckResult`.

## Ferramentas para PowerShell

Exemplo 1: Retorna os resultados de uma verificação do Trusted Advisor. A lista de verificações disponíveis do Trusted Advisor pode ser obtida usando o `TrustedAdvisorChecks Get-ASA`. A saída é o status geral da verificação, a data e hora em que a verificação foi executada pela última vez e o ID de verificação exclusivo da verificação específica. Para que os resultados sejam exibidos em japonês, adicione o parâmetro `-Language "ja"`.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Para obter detalhes da API, consulte [DescribeTrustedAdvisorCheckResult](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-ASATrustedAdvisorCheckSummary

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckSummary`.

### Ferramentas para PowerShell

Exemplo 1: Retorna o resumo mais recente da verificação especificada do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Exemplo 2: Retorna os resumos mais recentes das verificações especificadas do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte [DescribeTrustedAdvisorCheckSummaries](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-ASACase

O código de exemplo a seguir mostra como usar `New-ASACase`.

### Ferramentas para PowerShell

Exemplo 1: Cria um novo caso no AWS Support Center. Os valores dos `CategoryCode` parâmetros `-ServiceCode` e `-` podem ser obtidos usando o cmdlet `Get-asaService`. O valor do `SeverityCode` parâmetro `-` pode ser obtido usando o cmdlet `Get-ASASeverityLevel`. O valor do `IssueType` parâmetro `-` pode ser "atendimento ao cliente" ou "técnico". Se for bem-sucedido, o número do caso de AWS Support será exibido. Por padrão, o caso será tratado em inglês. Para usar o japonês, adicione o parâmetro `-Language "ja"`. Os `CommunicationBody` parâmetros `-ServiceCode`, `-CategoryCode`, `-Assunto` e `-` são obrigatórios.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode "low" -Subject "subject text" -CommunicationBody "description of the case" -CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Para obter detalhes da API, consulte [CreateCase](#) em Referência de AWS Tools for PowerShell cmdlet.

## Request-ASATrustedAdvisorCheckRefresh

O código de exemplo a seguir mostra como usar Request-ASATrustedAdvisorCheckRefresh.

### Ferramentas para PowerShell

Exemplo 1: Solicita uma atualização para a verificação especificada do Trusted Advisor.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Para obter detalhes da API, consulte [RefreshTrustedAdvisorCheck](#) em Referência de AWS Tools for PowerShell cmdlet.

## Resolve-ASACase

O código de exemplo a seguir mostra como usar Resolve-ASACase.

### Ferramentas para PowerShell

Exemplo 1: retorna o estado inicial do caso especificado e o estado atual após a conclusão da chamada para resolvê-lo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Para obter detalhes da API, consulte [ResolveCase](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Systems Manager usando o Tools for PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with Systems Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Add-SSMResourceTag

O código de exemplo a seguir mostra como usar Add-SSMResourceTag.

#### Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza uma janela de manutenção com novas tags. Não haverá saída se o comando for bem-sucedido. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$option1 = @{Key="Stack";Value=@"Production"}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -Tag $option1
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag. Não haverá saída se o comando for bem-sucedido.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -Tag $tag1
```

- Para obter detalhes da API, consulte [AddTagsToResource](#) em Referência de AWS Tools for PowerShell cmdlet.

### Edit-SSMDocumentPermission

O código de exemplo a seguir mostra como usar Edit-SSMDocumentPermission.

#### Ferramentas para PowerShell

Exemplo 1: esse exemplo adiciona permissões de "compartilhamento" a todas as contas para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Exemplo 2: esse exemplo adiciona permissões de "compartilhamento" a uma conta específica para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Para obter detalhes da API, consulte [ModifyDocumentPermission](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMActivation

O código de exemplo a seguir mostra como usar `Get-SSMActivation`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo fornece detalhes sobre as ativações em sua conta.

```
Get-SSMActivation
```

Saída:

```
ActivationId      : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate       : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description        :
ExpirationDate    : 3/2/2017 12:01:51 AM
Expired           : False
IAMRole           : AutomationRole
RegistrationLimit  : 10
RegistrationsCount : 0
```

- Para obter detalhes da API, consulte [DescribeActivations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAssociation

O código de exemplo a seguir mostra como usar `Get-SSMAssociation`.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo descreve a associação entre uma instância e um documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name           : AWS-UpdateSSMAgent
InstanceId      : i-0000293ffd8c57862
Date           : 2/23/2017 6:55:22 PM
Status.Name     : Pending
Status.Date     : 2/20/2015 8:31:11 AM
Status.Message  : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte [DescribeAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAssociationExecution

O código de exemplo a seguir mostra como usar Get-SSMAssociationExecution.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo retorna as execuções para o ID de associação fornecido

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion  : 2
CreatedTime        : 3/2/2019 8:53:29 AM
DetailedStatus     :
ExecutionId        : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate  : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status             : Success
```

- Para obter detalhes da API, consulte [DescribeAssociationExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAssociationExecutionTarget

O código de exemplo a seguir mostra como usar Get-SSMAssociationExecutionTarget.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe o ID do recurso e seu status de execução que fazem parte dos destinos de execução da associação

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
Select-Object ResourceId, Status
```

Saída:

ResourceId	Status
-----	-----
i-0b1b2a3456f7a890b	Success
i-01c12a45d6fc7a89f	Success
i-0a1caf234f56d7dc8	Success
i-012a3fd45af6dbcfe	Failed
i-0ddc1df23c4a5fb67	Success

Exemplo 2: esse comando verifica a execução específica de uma automação específica desde ontem, onde documento de comandos está associado. Além disso, ele verifica se a execução da associação falhou e, em caso afirmativo, exibe os detalhes da invocação do comando para a execução junto com o ID da instância

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
12345ca12-3456-2345-2b45-23456789012 |
Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
    }
}
```



```

    Get-SSMCommandInvocation -CommandId $execution.OutputSource.OutputSourceId -
    Detail:$true | Select-Object -ExpandProperty CommandPlugins
  }
}

```

**Saída:**

```

There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0 on
i-0a1caf234f56d7dc8

```

```

Name           : aws:runPowerShellScript
Output         :
                -----ERROR-----
                failed to run commands: exit status 1
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region    : eu-west-1
ResponseCode      : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime : 5/29/2019 11:04:49 AM
StandardErrorUrl   :
StandardOutputUrl  :
Status           : Failed
StatusDetails     : Failed

```

- Para obter detalhes da API, consulte [DescribeAssociationExecutionTargets](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SSMAssociationList**

O código de exemplo a seguir mostra como usar `Get-SSMAssociationList`.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo lista todas as associações para uma instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```

$filter1 = @{Key="InstanceId";Value=@"i-0000293ffd8c57862"}
Get-SSMAssociationList -AssociationFilterList $filter1

```

**Saída:**

```

AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion   :
InstanceId        : i-0000293ffd8c57862
LastExecutionDate : 2/20/2015 8:31:11 AM
Name              : AWS-UpdateSSMAgent
Overview          : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets           : {InstanceIds}

```

Exemplo 2: esse exemplo lista todas as associações para um documento de configuração. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```

$filter2 = @{Key="Name";Value=@"AWS-UpdateSSMAgent"}
Get-SSMAssociationList -AssociationFilterList $filter2

```

Saída:

```

AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion   :
InstanceId        : i-0000293ffd8c57862
LastExecutionDate : 2/20/2015 8:31:11 AM
Name              : AWS-UpdateSSMAgent
Overview          : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets           : {InstanceIds}

```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```

$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1

```

Saída:

```

AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion   :
InstanceId        : i-0000293ffd8c57862
LastExecutionDate : 2/20/2015 8:31:11 AM
Name              : AWS-UpdateSSMAgent

```

```

Overview      : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets       : {InstanceIds}

```

- Para obter detalhes da API, consulte [ListAssociations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAssociationVersionList

O código de exemplo a seguir mostra como usar Get-SSMAssociationVersionList.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo recupera todas as versões da associação fornecida.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

### Saída:

```

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate       : 3/12/2019 9:21:01 AM
DocumentVersion   :
MaxConcurrency    :
MaxErrors         :
Name              : AWS-GatherSoftwareInventory
OutputLocation    :
Parameters        : {}
ScheduleExpression :
Targets           : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate       : 3/2/2019 8:53:29 AM
DocumentVersion   :
MaxConcurrency    :
MaxErrors         :
Name              : AWS-GatherSoftwareInventory

```

```

OutputLocation      :
Parameters          : {}
ScheduleExpression  : rate(30minutes)
Targets             : {InstanceIds}

```

- Para obter detalhes da API, consulte [ListAssociationVersions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAutomationExecution

O código de exemplo a seguir mostra como usar Get-SSMAutomationExecution.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe os detalhes de uma execução do Automation.

```

Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"

```

### Saída:

```

AutomationExecutionId      : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus  : Failed
DocumentName               : AWS-UpdateLinuxAmi
DocumentVersion            : 1
ExecutionEndTime           : 2/22/2017 9:17:08 PM
ExecutionStartTime         : 2/22/2017 9:17:02 PM
FailureMessage             : Step launchInstance failed maximum allowed times. You
                             are not authorized to perform this operation. Encoded
                             authorization failure message:
                             B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
                             fv1x-04q5Fjff6glh
                             Yb6TI5b0GQeeNrpwNvpDzm0-
                             PSR1swlAbg9fdM9BcNjyrznsPukWpuKu9EC10u6v30XU1KC9nZ7mPlWFMFZNkSioQqpWWEvMw-
                             GZktsQzm67q0hUhBN0LWYhbS
                             pkfiqzY-5nw3S0obx30fhd3EJa50_-
                             GjV_a0nFXQJa70ik40bF0rEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84X0F5WmA60f1pi5GI08i-
                             nRfZS6oDeU
                             gELBjjoFKD8s3L2aI0B6umWVxnQ0jqhQRxwJ53b54sZJ2PW3v_mtg9-
                             q0CK0ezS3xfh_y0ilaUG0AZG-xjQFuvU_JZedWpla3xi-MZsmb1AifBI
                             (Service: AmazonEC2; Status Code: 403; Error Code:
                             UnauthorizedOperation; Request ID:

```

```

        6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs      : {[createImage.ImageId,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters  : {[AutomationAssumeRole,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions : {launchInstance, updateOSSoftware, stopInstance,
  createImage...}

```

Exemplo 2: esse exemplo lista os detalhes da etapa para o ID de execução do Automation fornecido

```

Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
StepName, Action, StepStatus, ValidNextSteps

```

Saída:

StepName	Action	StepStatus	ValidNextSteps
LaunchInstance	aws:runInstances	Success	{OSCompatibilityCheck}
OSCompatibilityCheck	aws:runCommand	Success	{RunPreUpdateScript}
RunPreUpdateScript	aws:runCommand	Success	{UpdateEC2Config}
UpdateEC2Config	aws:runCommand	Cancelled	{}
UpdateSSMAgent	aws:runCommand	Pending	{}
UpdateAWSPVDriver	aws:runCommand	Pending	{}
UpdateAWSEnaNetworkDriver	aws:runCommand	Pending	{}
UpdateAWSNVMe	aws:runCommand	Pending	{}
InstallWindowsUpdates	aws:runCommand	Pending	{}
RunPostUpdateScript	aws:runCommand	Pending	{}
RunSysprepGeneralize	aws:runCommand	Pending	{}
StopInstance	aws:changeInstanceState	Pending	{}
CreateImage	aws:createImage	Pending	{}
TerminateInstance	aws:changeInstanceState	Pending	{}

- Para obter detalhes da API, consulte [GetAutomationExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAutomationExecutionList

O código de exemplo a seguir mostra como usar Get-SSMAutomationExecutionList.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo descreve todas as execuções do Automation ativas e encerradas associadas à sua conta.

```
Get-SSMAutomationExecutionList
```

#### Saída:

```
AutomationExecutionId      : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus  : Failed
DocumentName                : AWS-UpdateLinuxAmi
DocumentVersion             : 1
ExecutedBy                  : admin
ExecutionEndTime            : 2/22/2017 9:17:08 PM
ExecutionStartTime         : 2/22/2017 9:17:02 PM
LogFile                     :
Outputs                     : {[createImage.ImageId,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Exemplo 2: Este exemplo exibe ExecutionID, documento, data e hora de início/término da execução para execuções que não sejam “Sucesso” AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
-ne "Success" | Select-Object AutomationExecutionId, DocumentName,
AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

#### Saída:

```
AutomationExecutionId      DocumentName
AutomationExecutionStatus  ExecutionStartTime  ExecutionEndTime
-----
-----
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
Cancelled                    4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
```

```
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
Cancelled          4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM Failed
```

- Para obter detalhes da API, consulte [DescribeAutomationExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAutomationStepExecution

O código de exemplo a seguir mostra como usar `Get-SSMAutomationStepExecution`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe informações sobre todas as execuções de etapas ativas e encerradas em um fluxo de trabalho do Automation.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Saída:

StepName	Action	StepStatus
-----	-----	-----
LaunchInstance	aws:runInstances	Success
OSCompatibilityCheck	aws:runCommand	Success
RunPreUpdateScript	aws:runCommand	Success
UpdateEC2Config	aws:runCommand	Cancelled
UpdateSSMAgent	aws:runCommand	Pending
UpdateAWSPVDriver	aws:runCommand	Pending
UpdateAWSEnaNetworkDriver	aws:runCommand	Pending
UpdateAWSNVMe	aws:runCommand	Pending
InstallWindowsUpdates	aws:runCommand	Pending
RunPostUpdateScript	aws:runCommand	Pending
RunSysprepGeneralize	aws:runCommand	Pending
StopInstance	aws:changeInstanceState	Pending
CreateImage	aws:createImage	Pending
TerminateInstance	aws:changeInstanceState	Pending

- Para obter detalhes da API, consulte [DescribeAutomationStepExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMAvailablePatch

O código de exemplo a seguir mostra como usar Get-SSMAvailablePatch.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém todos os patches disponíveis para o Windows Server 2012 que apresentam gravidade MSRC crítica. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

### Saída:

```
Classification : SecurityUpdates
ContentUrl      : https://support.microsoft.com/en-us/kb/2727528
Description     : A security issue has been identified that could allow an
                  unauthenticated remote attacker to compromise your system and gain control
                  over it. You can help protect your system by installing this update
                  from Microsoft. After you install this update, you may have to
                  restart your system.
Id              : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber        : KB2727528
Language        : All
MsrcNumber      : MS12-072
MsrcSeverity     : Critical
Product         : WindowsServer2012
ProductFamily   : Windows
ReleaseDate     : 11/13/2012 6:00:00 PM
Title           : Security Update for Windows Server 2012 (KB2727528)
Vendor          : Microsoft
...
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
```



```
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Exemplo 3: Este exemplo busca todas as atualizações lançadas nos últimos 20 dias e aplicáveis aos produtos correspondentes WindowsServer a 2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
Title
```

Saída:

ReleaseDate	Product	Title
-----	-----	-----
4/9/2019 5:00:12 PM	WindowsServer2019	2019-04 Security Update for Adobe Flash Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM	WindowsServer2019	2019-04 Cumulative Update for Windows Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM	WindowsServer2019	2019-03 Servicing Stack Update for Windows Server 2019 for x64-based Systems (KB4493510)

- Para obter detalhes da API, consulte [DescribeAvailablePatches](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMCommand

O código de exemplo a seguir mostra como usar Get-SSMCommand.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todos os comandos solicitados.

```
Get-SSMCommand
```

Saída:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
```

```

CompletedCount      : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}

```

Exemplo 2: esse exemplo obtém o status de um comando específico

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Exemplo 3: esse exemplo recupera todos os comandos SSM invocados após 2019-04-01T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property RequestedDateTime -Descending
```

Saída:

CommandId	DocumentName	Status
RequestedDateTime		
-----	-----	-----
-----		
edb1b23e-456a-7adb-aef8-90e-012ac34f	AWS-RunPowerShellScript	Cancelled 4/16/2019
5:45:23 AM		
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9	AWS-ConfigureAWSPackage	Success 4/6/2019
9:19:42 AM		

```

12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed 4/2/2019
4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks Failed 4/1/2019
2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript Success 4/1/2019
1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks Failed 4/1/2019
11:13:09 AM

```

- Para obter detalhes da API, consulte [ListCommands](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMCommandInvocation

O código de exemplo a seguir mostra como usar Get-SSMCommandInvocation.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as invocações de um comando.

```

Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
>true

```

### Saída:

```

CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId          : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :

```

Exemplo 2: Este exemplo lista a invocação do ID de comando CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

### Saída:

```
Name                : aws:runPowerShellScript
Output              : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
                    remainingdownload: s3://dd-aess-r-ctmer/KUM0.png to ..\..\programdata\KUM0.png
                    kumo available

OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     : eu-west-1
ResponseCode       : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime : 4/3/2019 11:53:21 AM
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
```

- Para obter detalhes da API, consulte [ListCommandInvocations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMCommandInvocationDetail

O código de exemplo a seguir mostra como usar `Get-SSMCommandInvocationDetail`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe os detalhes de um comando executado em uma instância.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
"b8eac879-0541-439d-94ec-47a80d554f44"
```

### Saída:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
Comment            : IP config
DocumentName       : AWS-RunShellScript
ExecutionElapsedTime : PT0.004S
```

```

ExecutionEndDateTime : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId           : i-0cb2b964d3e14fd9f
PluginName          : aws:runShellScript
ResponseCode        : 0
StandardErrorContent :
StandardErrorUrl    :
StandardOutputContent :
StandardOutputUrl   :
Status              : Success
StatusDetails       : Success

```

- Para obter detalhes da API, consulte [GetCommandInvocation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMComplianceItemList

O código de exemplo a seguir mostra como usar `Get-SSMComplianceItemList`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo mostra a lista de itens de conformidade para o ID e tipo de recurso fornecidos, filtrados pelo tipo de conformidade "Associação"

```

Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}

```

Saída:

```

ComplianceType : Association
Details        : {[DocumentName, AWS-GatherSoftwareInventory], [DocumentVersion,
1]}
ExecutionSummary : Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id              : 123a45a1-c234-1234-1245-67891236db4e
ResourceId     : i-1a2caf345f67d0dc2
ResourceType   : ManagedInstance
Severity       : UNSPECIFIED
Status        : COMPLIANT
Title         :

```

- Para obter detalhes da API, consulte [ListComplianceItems](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMComplianceSummaryList

O código de exemplo a seguir mostra como usar `Get-SSMComplianceSummaryList`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo devolve uma contagem resumida de recursos em ou fora de conformidade para todos os tipos de conformidade.

```
Get-SSMComplianceSummaryList
```

Saída:

```
ComplianceType CompliantSummary
NonCompliantSummary
-----
-----
FleetTotal      Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association     Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Para obter detalhes da API, consulte [ListComplianceSummaries](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMConnectionStatus

O código de exemplo a seguir mostra como usar `Get-SSMConnectionStatus`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo recupera o status de conexão do Gerenciador de Sessões de uma instância para determinar se ela está conectada e pronta para receber conexões do Gerenciador de Sessões.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

**Saída:**

```
Status      Target
-----      -
Connected  i-0a1caf234f12d3dc4
```

- Para obter detalhes da API, consulte [GetConnectionStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SSMDefaultPatchBaseline**

O código de exemplo a seguir mostra como usar `Get-SSMDefaultPatchBaseline`.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo exibe a lista de referência de patches padrão.

```
Get-SSMDefaultPatchBaseline
```

**Saída:**

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Para obter detalhes da API, consulte [GetDefaultPatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SSMDeployablePatchSnapshotForInstance**

O código de exemplo a seguir mostra como usar `Get-SSMDeployablePatchSnapshotForInstance`.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo exibe o instantâneo atual da lista de referência de patches usada por uma instância. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que use as credenciais da instância, o exemplo passa um objeto **Amazon.Runtime.InstanceProfileAWSCredentials** para o parâmetro `Credentials`.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
```

```
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Saída:

```
InstanceId          SnapshotDownloadUrl
-----
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Exemplo 2: Este exemplo mostra como obter o completo SnapshotDownloadUrl. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que ele use as credenciais da instância, o exemplo configura a PowerShell sessão para usar um **Amazon.Runtime.InstanceProfileAWSCredentials** objeto.

```
Set-AWSCredential -Credential
([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Saída:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Para obter detalhes da API, consulte [GetDeployablePatchSnapshotForInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMDocument

O código de exemplo a seguir mostra como usar Get-SSMDocument.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo retorna o conteúdo de um documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Saída:



```
Content
```

```
-----  
{...}
```

Exemplo 2: esse exemplo exibe o conteúdo completo de um documento.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
  "schemaVersion":"2.0",
  "description":"Run an updated script",
  "parameters":{
    "commands":{
      "type":"StringList",
      "description":"(Required) Specify a shell script or a command to run.",
      "minItems":1,
      "displayType":"textarea"
    }
  },
  "mainSteps":[
    {
      "action":"aws:runShellScript",
      "name":"runShellScript",
      "inputs":{
        "commands":"{{ commands }}"
      }
    },
    {
      "action":"aws:runPowerShellScript",
      "name":"runPowerShellScript",
      "inputs":{
        "commands":"{{ commands }}"
      }
    }
  ]
}
```

- Para obter detalhes da API, consulte [GetDocument](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMDocumentDescription

O código de exemplo a seguir mostra como usar `Get-SSMDocumentDescription`.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo retorna informações sobre um documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Saída:

```
CreatedDate      : 2/24/2017 5:25:13 AM
DefaultVersion   : 1
Description      : Run an updated script
DocumentType     : Command
DocumentVersion  : 1
Hash             : f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion    : 1
Name             : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion    : 2.0
Sha1             :
Status          : Active
```

- Para obter detalhes da API, consulte [DescribeDocument](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMDocumentList

O código de exemplo a seguir mostra como usar Get-SSMDocumentList.

## Ferramentas para PowerShell

Exemplo 1: lista todos os documentos de configuração em sua conta.

```
Get-SSMDocumentList
```

Saída:

```
DocumentType     : Command
```

```

DocumentVersion : 1
Name             : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name           : AWS-ConfigureAWSPackage
Owner         : Amazon
PlatformTypes  : {Windows, Linux}
SchemaVersion  : 2.0

DocumentType    : Command
DocumentVersion : 1
Name           : AWS-ConfigureCloudWatch
Owner         : Amazon
PlatformTypes  : {Windows}
SchemaVersion  : 1.2
...

```

Exemplo 2: esse exemplo recupera todos os documentos de automação com o nome correspondente a "Plataform"

```

Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
Where-Object Name -Match "Plataform"

```

Saída:

```

DocumentFormat : JSON
DocumentType   : Automation
DocumentVersion : 7
Name          : KT-Get-Platform
Owner         : 987654123456
PlatformTypes  : {Windows, Linux}
SchemaVersion  : 0.3
Tags          : {}
TargetType    :
VersionName   :

```

- Para obter detalhes da API, consulte [ListDocuments](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMDocumentPermission

O código de exemplo a seguir mostra como usar `Get-SSMDocumentPermission`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Saída:

CreatedDate	DocumentVersion	IsDefaultVersion	Name
-----	-----	-----	----
2/24/2017 5:25:13 AM	1	True	RunShellScript

- Para obter detalhes da API, consulte [DescribeDocumentPermission](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMDocumentVersionList

O código de exemplo a seguir mostra como usar `Get-SSMDocumentVersionList`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo retorna a lista de permissões para um documento.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

Saída:

```
all
```

- Para obter detalhes da API, consulte [ListDocumentVersions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMEffectiveInstanceAssociationList

O código de exemplo a seguir mostra como usar `Get-SSMEffectiveInstanceAssociationList`.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo descreve as associações efetivas de uma instância.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
5
```

Saída:

```
AssociationId          Content
-----
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Exemplo 2: esse exemplo exibe o conteúdo das associações efetivas de uma instância.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Saída:

```
{
  "schemaVersion": "1.2",
  "description": "Update the Amazon SSM Agent to the latest version or specified
version.",
  "parameters": {
    "version": {
      "default": "",
      "description": "(Optional) A specific version of the Amazon SSM Agent to
install. If not specified, the agen
t will be updated to the latest version.",
      "type": "String"
    },
    "allowDowngrade": {
      "default": "false",
      "description": "(Optional) Allow the Amazon SSM Agent service to be
downgraded to an earlier version. If set
to false, the service can be upgraded to newer versions only (default). If set to
true, specify the earlier version.",
      "type": "String",
      "allowedValues": [
        "true",
```

```

        "false"
      ]
    }
  },
  "runtimeConfig": {
    "aws:updateSsmAgent": {
      "properties": [
        {
          "agentName": "amazon-ssm-agent",
          "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
          "allowDowngrade": "{{ allowDowngrade }}",
          "targetVersion": "{{ version }}"
        }
      ]
    }
  }
}

```

- Para obter detalhes da API, consulte [DescribeEffectiveInstanceAssociations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMEffectivePatchesForPatchBaseline

O código de exemplo a seguir mostra como usar Get-SSMEffectivePatchesForPatchBaseline.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo mostra todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Saída:

```

Patch                                PatchStatus
-----                                -
Amazon.SimpleSystemsManagement.Model.Patch
Amazon.SimpleSystemsManagement.Model.PatchStatus

```

Exemplo 2: esse exemplo mostra o status do patch para todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Saída:

```
ApprovalDate          DeploymentStatus
-----
12/21/2010 6:00:00 PM APPROVED
```

- Para obter detalhes da API, consulte [DescribeEffectivePatchesForPatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInstanceAssociationsStatus

O código de exemplo a seguir mostra como usar Get-SSMInstanceAssociationsStatus.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo mostra detalhes das associações de uma instância.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus    : Pending
DocumentVersion   : 1
ErrorCode         :
ExecutionDate     : 2/20/2015 8:31:11 AM
ExecutionSummary  : temp_status_change
InstanceId        : i-0000293ffd8c57862
Name              : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Exemplo 2: esse exemplo verifica o status da associação da instância para o ID da instância fornecido e exibe o status de execução dessas associações

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
{Get-SSMAssociationExecution -AssociationId .AssociationId}
```

**Saída:**

```
AssociationId      : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion : 2
CreatedTime       : 3/2/2019 8:53:29 AM
DetailedStatus    :
ExecutionId       : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status            : Success
```

- Para obter detalhes da API, consulte [DescribeInstanceAssociationsStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SSMInstanceInformation**

O código de exemplo a seguir mostra como usar Get-SSMInstanceInformation.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo mostra detalhes de cada uma de suas instâncias.

```
Get-SSMInstanceInformation
```

**Saída:**

```
ActivationId      :
AgentVersion      : 2.0.672.0
AssociationOverview :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus : Success
ComputerName      : ip-172-31-44-222.us-west-2.compute.internal
IamRole           :
InstanceId        : i-0cb2b964d3e14fd9f
IPAddress         : 172.31.44.222
IsLatestVersion   : True
LastAssociationExecutionDate : 2/24/2017 3:18:09 AM
LastPingDateTime  : 2/24/2017 3:35:03 AM
```



```

LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

```

Exemplo 2: Este exemplo mostra como usar o parâmetro `-Filter` para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **us-east-1** com um **AgentVersion** de **2.2.800.0**. Você pode encontrar uma lista de valores-chave válidos de `-Filter` no tópico de referência da InstanceInformation API ([https://docs.aws.amazon.com/systems-manager/latest/APIReference/API\\_InstanceInformation.html#systemsmanager-Type-InstanceInformation](https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type-InstanceInformation)).

```

$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters

```

Saída:

```

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                      :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                        : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                  :
InstanceId                               : i-EXAMPLEb0792d98ce
IPAddress                                : 10.0.0.01
IsLatestVersion                          : False
LastAssociationExecutionDate             : 8/16/2018 12:02:50 AM
LastPingDateTime                         : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate   : 8/16/2018 12:02:50 AM
Name                                      :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393

```

```

RegistrationDate           : 1/1/0001 12:00:00 AM
ResourceType               : EC2Instance

ActivationId               :
AgentVersion               : 2.2.800.0
AssociationOverview       :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus         : Success
ComputerName               : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                   :
InstanceId                 : i-EXAMPLEac7501d023
IPAddress                  : 10.0.0.02
IsLatestVersion           : False
LastAssociationExecutionDate : 8/16/2018 12:00:20 AM
LastPingDateTime          : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                       :
PingStatus                 : Online
PlatformName               : Microsoft Windows Server 2016 Datacenter
PlatformType              : Windows
PlatformVersion            : 10.0.14393
RegistrationDate           : 1/1/0001 12:00:00 AM
ResourceType               : EC2Instance

```

Exemplo 3: Este exemplo mostra como usar o `InstanceInformationFilterList` parâmetro - para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **us-east-1** com **PlatformTypes Windows** ou **Linux**. Você pode encontrar uma lista de `InstanceInformationFilterList` valores-chave válidos no tópico de referência da `InstanceInformationFilter` API ([https://docs.aws.amazon.com/systems-manager/latest/APIReference/API\\_InstanceInformationFilter.html](https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html)).

```

$Filters = @{
    Key="PlatformTypes"
    ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters

```

Saída:

```

ActivationId               :
AgentVersion               : 2.2.800.0

```

```

AssociationOverview          :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus            : Success
ComputerName                 : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                      :
InstanceId                   : i-EXAMPLEb0792d98ce
IPAddress                    : 10.0.0.27
IsLatestVersion              : False
LastAssociationExecutionDate  : 8/16/2018 12:02:50 AM
LastPingDateTime             : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                         :
PingStatus                   : Online
PlatformName                 : Ubuntu Server 18.04 LTS
PlatformType                 : Linux
PlatformVersion              : 18.04
RegistrationDate              : 1/1/0001 12:00:00 AM
ResourceType                 : EC2Instance

ActivationId                 :
AgentVersion                 : 2.2.800.0
AssociationOverview          :
  Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus            : Success
ComputerName                 : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                      :
InstanceId                   : i-EXAMPLEac7501d023
IPAddress                    : 10.0.0.100
IsLatestVersion              : False
LastAssociationExecutionDate  : 8/16/2018 12:00:20 AM
LastPingDateTime             : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                         :
PingStatus                   : Online
PlatformName                 : Microsoft Windows Server 2016 Datacenter
PlatformType                 : Windows
PlatformVersion              : 10.0.14393
RegistrationDate              : 1/1/0001 12:00:00 AM
ResourceType                 : EC2Instance

```

Exemplo 4: Este exemplo lista as instâncias e exportações InstanceId gerenciadas por ssm LastPingDateTime e PlatformName em um arquivo csv. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,  
PlatformName | Export-Csv Instance-details.csv -NoTypeInfo
```

- Para obter detalhes da API, consulte [DescribeInstanceInformation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInstancePatch

O código de exemplo a seguir mostra como usar Get-SSMInstancePatch.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém os detalhes de conformidade do patch para uma instância.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Para obter detalhes da API, consulte [DescribeInstancePatches](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInstancePatchState

O código de exemplo a seguir mostra como usar Get-SSMInstancePatchState.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém os estados resumidos de patches para uma instância.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Exemplo 2: esse exemplo obtém os estados resumidos de patches para duas instâncias.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Para obter detalhes da API, consulte [DescribeInstancePatchStates](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInstancePatchStatesForPatchGroup

O código de exemplo a seguir mostra como usar `Get-SSMInstancePatchStatesForPatchGroup`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém os estados de resumo de patches por instância de um grupo de patches.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Para obter detalhes da API, consulte [DescribeInstancePatchStatesForPatchGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInventory

O código de exemplo a seguir mostra como usar `Get-SSMInventory`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém os metadados personalizados do seu inventário.

```
Get-SSMInventory
```

Saída:

```
Data
  Id
----
--
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte [GetInventory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInventoryEntriesList

O código de exemplo a seguir mostra como usar `Get-SSMInventoryEntriesList`.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as entradas de inventário personalizadas para uma instância.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo"
```

Saída:

```
CaptureTime    : 2016-08-22T10:01:01Z
Entries        :
  {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId     : i-0cb2b964d3e14fd9f
NextToken      :
SchemaVersion  : 1.0
TypeName       : Custom:RackInfo
```

Exemplo 2: esse exemplo lista os detalhes.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
"Custom:RackInfo").Entries
```

Saída:

```
Key           Value
---           -
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Para obter detalhes da API, consulte [ListInventoryEntries](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMInventoryEntryList

O código de exemplo a seguir mostra como usar Get-SSMInventoryEntryList.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo recupera entradas **AWS:Network** de inventário de tipo para a instância.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |  
Select-Object -ExpandProperty Entries
```

### Saída:

Key	Value
---	-----
DHCPServer	172.31.11.2
DNSServer	172.31.0.1
Gateway	172.31.11.2
IPV4	172.31.11.222
IPV6	fe12::3456:7da8:901a:12a3
MacAddress	1A:23:4E:5B:FB:67
Name	Amazon Elastic Network Adapter
SubnetMask	255.255.240.0

- Para obter detalhes da API, consulte [Get-SSM InventoryEntryList](#) em AWS Tools for PowerShell Cmdlet Reference.

## Get-SSMInventorySchema

O código de exemplo a seguir mostra como usar Get-SSMInventorySchema.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo retorna uma lista de nomes de tipos de inventário para a conta.

```
Get-SSMInventorySchema
```

- Para obter detalhes da API, consulte [GetInventorySchema](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMLatestEC2Image

O código de exemplo a seguir mostra como usar Get-SSMLatestEC2Image.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista todas as AMIs mais recentes do Windows.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

**Saída:**

Name	Value
----	-----
Windows_Server-2008-R2_SP1-English-64Bit-SQL_2012_SP4_Express ami-0e5ddd288daff4fab	
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base ami-0c5ea64e6bec1cb50	
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base ami-09775eff0bf8c113d	
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base ami-025064b67e28cf5df	
...	

Exemplo 2: Este exemplo recupera o ID da AMI de uma imagem específica do Amazon Linux para a região us-west-2.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-x86_64-ebs -Region us-west-2
```

**Saída:**

```
ami-09b92cd132204c704
```

Exemplo 3: Este exemplo lista todas as AMIs mais recentes do Windows que correspondem à expressão curinga especificada.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

**Saída:**

Name	Value
----	-----
Windows_Server-2019-English-Full-SQL_2017_Web	ami-085e9d27da5b73a42
Windows_Server-2019-English-STIG-Core	ami-0bfd85c29148c7f80
Windows_Server-2019-English-Full-SQL_2019_Web	ami-02099560d7fb11f20
Windows_Server-2019-English-Full-SQL_2016_SP2_Standard	ami-0d7ae2d81c07bd598
...	



- Para obter detalhes da API, consulte [Get-SSMLatestec2Image em Cmdlet Reference.AWS Tools for PowerShell](#)

## Get-SSMMaintenanceWindow

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindow.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém detalhes sobre uma janela de manutenção.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte [GetMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowExecution

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindowExecution.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista informações sobre uma tarefa que foi executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

**Saída:**

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte [GetMaintenanceWindowExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-SSMMaintenanceWindowExecutionList**

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecutionList`.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo lista todas as execuções para uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

**Saída:**

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId         : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo lista todas as execuções para uma janela de manutenção antes de uma data especificada.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

Exemplo 3: esse exemplo lista todas as execuções para uma janela de manutenção após uma data especificada.

```
$option1 = @{Key="ExecutedAfter";Values=@"2016-11-04T05:00:00Z"}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
$option1
```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindowExecutions](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowExecutionTask

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindowExecutionTask.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista informações sobre uma tarefa que fazia parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
-WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

### Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole      : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime        : 2/21/2017 4:00:34 PM
Status           : FAILED
StatusDetails    : The maximum error count was exceeded.
TaskArn          : AWS-RunShellScript
TaskExecutionId  : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters   :
  {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsManag
    meterValueExpression]}
Type             : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte [GetMaintenanceWindowExecutionTask](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowExecutionTaskInvocationList

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindowExecutionTaskInvocationList.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista as invocações de uma tarefa executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-  
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-  
da3b2a638355"
```

### Saída:

```
EndTime           : 2/21/2017 4:00:34 PM  
ExecutionId       :  
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b  
OwnerInformation  :  
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":  
["i-0000293ffd8c57862"],"parameters":{"commands":["df"],"maxConcurrency":"1",  
"maxErrors":"1"}  
StartTime         : 2/21/2017 4:00:34 PM  
Status            : FAILED  
StatusDetails     : The instance IDs list contains an invalid entry.  
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586  
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355  
WindowTargetId    :
```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindowExecutionTaskInvocations](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowExecutionTaskList

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindowExecutionTaskList.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as tarefas associadas à execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId  
"518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM  
StartTime        : 2/21/2017 4:00:34 PM  
Status           : SUCCESS  
TaskArn          : AWS-RunShellScript  
TaskExecutionId  : ac0c6ae1-daa3-4a89-832e-d384503b6586  
TaskType         : RUN_COMMAND  
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindowExecutionTasks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowList

O código de exemplo a seguir mostra como usar Get-SSMMaintenanceWindowList.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as janelas de manutenção em sua conta.

```
Get-SSMMaintenanceWindowList
```

Saída:

```
Cutoff   : 1  
Duration : 4  
Enabled  : True  
Name     : My-First-Maintenance-Window  
WindowId : mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindows](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowTarget

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowTarget`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todos os destinos para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

#### Saída:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets         : {InstanceIds}
WindowId        : mw-06cf17cbefcb4bf4f
WindowTargetId  : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets         : {InstanceIds}
WindowId        : mw-06cf17cbefcb4bf4f
WindowTargetId  : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindowTargets](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMMaintenanceWindowTaskList

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowTaskList`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todas as tarefas para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

#### Saída:

```
LoggingInfo      :
MaxConcurrency   : 1
```

```

MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets       : {InstanceIds}
TaskArn       : AWS-RunShellScript
TaskParameters : {[commands,
  Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type          : RUN_COMMAND
WindowId      : mw-06cf17cbefcb4bf4f
WindowTaskId  : a23e338d-ff30-4398-8aa3-09cd052ebf17

```

- Para obter detalhes da API, consulte [DescribeMaintenanceWindowTasks](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMParameterHistory

O código de exemplo a seguir mostra como usar Get-SSMParameterHistory.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista o histórico de valores de um parâmetro.

```
Get-SSMParameterHistory -Name "Welcome"
```

Saída:

```

Description      :
KeyId           :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name            : Welcome
Type            : String
Value           : helloWorld

```

- Para obter detalhes da API, consulte [GetParameterHistory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMParameterList

O código de exemplo a seguir mostra como usar Get-SSMParameterList.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista todos os parâmetros.

```
Get-SSMParameterList
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type            : String
```

- Para obter detalhes da API, consulte [DescribeParameters](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMParameterValue

O código de exemplo a seguir mostra como usar Get-SSMParameterValue.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista os valores de um parâmetro.

```
Get-SSMParameterValue -Name "Welcome"
```

Saída:

```
InvalidParameters Parameters
-----
{}                  {Welcome}
```

Exemplo 2: esse exemplo retorna os detalhes do valor.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Saída:

```
Name    Type    Value
```



```

-----
Welcome String Good day, Sunshine!

```

- Para obter detalhes da API, consulte [GetParameters](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMPatchBaseline

O código de exemplo a seguir mostra como usar Get-SSMPatchBaseline.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo mostra todas as listas de referência de patches.

```
Get-SSMPatchBaseline
```

Saída:

```

BaselineDescription                                     BaselineId
-----
Default Patch Baseline Provided by AWS.                arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
Production-B...

```

Exemplo 2: Este exemplo lista todas as linhas de base de patch fornecidas pelo. AWS A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 3: esse exemplo mostra todas as listas de referência de patches pertencentes a você. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{"Key"="OWNER";Values=@("Self")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 4: Com a PowerShell versão 2, você deve usar `New-Object` para criar cada tag.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Saída:

BaselineDescription	BaselineName	BaselineId	DefaultBaseline
-----	-----	-----	e
Default Patch Baseline Provided by AWS.		arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966	AWS-DefaultPatchBaseline True

- Para obter detalhes da API, consulte [DescribePatchBaselines](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMPatchBaselineDetail

O código de exemplo a seguir mostra como usar `Get-SSMPatchBaselineDetail`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe os detalhes de uma lista de referência de patches.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```

ApprovalRules      : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches    : {}
BaselineId         : pb-03da896ca3b68b639
CreatedDate        : 3/3/2017 5:02:19 PM
Description        : Baseline containing all updates approved for production systems
GlobalFilters      : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate       : 3/3/2017 5:02:19 PM
Name               : Production-Baseline
PatchGroups        : {}
RejectedPatches    : {}

```

- Para obter detalhes da API, consulte [GetPatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMPatchBaselineForPatchGroup

O código de exemplo a seguir mostra como usar Get-SSMPatchBaselineForPatchGroup.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exibe a lista de referência de patches para um grupo de patches.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Saída:

```

BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production

```

- Para obter detalhes da API, consulte [GetPatchBaselineForPatchGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMPatchGroup

O código de exemplo a seguir mostra como usar Get-SSMPatchGroup.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo lista os registros do grupo de patches.

```
Get-SSMPatchGroup
```

Saída:

```
BaselineIdentity                PatchGroup
-----
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Para obter detalhes da API, consulte [DescribePatchGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMPatchGroupState

O código de exemplo a seguir mostra como usar Get-SSMPatchGroupState.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém o resumo da conformidade de patches de alto nível para um grupo de patches.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Saída:

```
Instances                : 4
InstancesWithFailedPatches : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches : 3
InstancesWithMissingPatches : 0
InstancesWithNotApplicablePatches : 0
```

- Para obter detalhes da API, consulte [DescribePatchGroupState](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMResourceComplianceSummaryList

O código de exemplo a seguir mostra como usar Get-SSMResourceComplianceSummaryList.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo obtém uma contagem resumida em nível de recurso. O resumo inclui informações sobre status de conformidade e não conformidade e contagens detalhadas de gravidade de itens de conformidade para produtos que correspondem a "Windows10". Como o MaxResult padrão é 100 se o parâmetro não for especificado e esse valor não for válido, o MaxResult parâmetro será adicionado e o valor será definido como 50.

```
$FilterValues = @{
    "Key"="Product"
    "Type"="EQUAL"
    "Values"="Windows10"
}

Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Para obter detalhes da API, consulte [ListResourceComplianceSummaries](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-SSMResourceTag

O código de exemplo a seguir mostra como usar Get-SSMResourceTag.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo lista as tags para uma janela de manutenção.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow"
```

Saída:

```
Key    Value
---    -
Stack  Production
```

- Para obter detalhes da API, consulte [ListTagsForResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SSMActivation

O código de exemplo a seguir mostra como usar New-SSMActivation.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cria uma instância gerenciada.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

### Saída:

```
ActivationCode      ActivationId
-----
KWChh0xBTiwDcKE9B1KC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Para obter detalhes da API, consulte [CreateActivation](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SSMAssociation

O código de exemplo a seguir mostra como usar New-SSMAssociation.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo associa um documento de configuração a uma instância usando IDs de instância.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

### Saída:

```
Name           : AWS-UpdateSSMAgent
InstanceId      : i-0000293ffd8c57862
Date           : 2/23/2017 6:55:22 PM
Status.Name     : Associated
Status.Date    : 2/20/2015 8:31:11 AM
Status.Message  : Associated with AWS-UpdateSSMAgent
```

```
Status.AdditionalInfo :
```

Exemplo 2: esse exemplo associa um documento de configuração a uma instância usando destinos.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}  
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Saída:

```
Name                : AWS-UpdateSSMAgent  
InstanceId           :  
Date                : 3/1/2017 6:22:21 PM  
Status.Name         :  
Status.Date         :  
Status.Message      :  
Status.AdditionalInfo :
```

Exemplo 3: esse exemplo associa um documento de configuração a uma instância usando destinos e parâmetros.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}  
$params = @{  
  "action"="configure"  
  "mode"="ec2"  
  "optionalConfigurationSource"="ssm"  
  "optionalConfigurationLocation"=""  
  "optionalRestart"="yes"  
}  
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName "CWConfiguration" -  
Target $target -Parameter $params
```

Saída:

```
Name                : Configure-CloudWatch  
InstanceId           :  
Date                : 5/17/2018 3:17:44 PM  
Status.Name         :  
Status.Date         :  
Status.Message      :  
Status.AdditionalInfo :
```

Exemplo 4: esse exemplo cria uma associação com todas as instâncias na região, com **AWS-GatherSoftwareInventory**. Ele também fornece arquivos personalizados e locais de registro nos parâmetros a serem coletados

```
$params =
  [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]::new()
$params["windowsRegistry"] = '[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
-Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Saída:

```
Name           : AWS-GatherSoftwareInventory
InstanceId      :
Date           : 6/9/2019 8:57:56 AM
Status.Name     :
Status.Date    :
Status.Message  :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte [CreateAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SSMAssociationFromBatch

O código de exemplo a seguir mostra como usar `New-SSMAssociationFromBatch`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo associa um documento de configuração a várias instâncias. A saída retorna uma lista de operações bem e malsucedidas, se aplicável.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
New-SSMAssociationFromBatch -Entry $option1,$option2
```



**Saída:**

```
Failed Successful
-----
{}          {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Exemplo 2: esse exemplo mostrará os detalhes completos de uma operação bem-sucedida.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@"AWS-UpdateSSMAgent"}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@"AWS-UpdateSSMAgent"}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Para obter detalhes da API, consulte [CreateAssociationBatch](#) em Referência de AWS Tools for PowerShell cmdlet.

**New-SSMDocument**

O código de exemplo a seguir mostra como usar New-SSMDocument.

**Ferramentas para PowerShell**

Exemplo 1: esse exemplo cria um documento na sua conta. O documento deve estar em formato JSON. Para obter mais informações sobre como escrever um documento de configuração, consulte Documento de configuração na Referência da API do SSM.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
"RunShellScript" -DocumentType "Command"
```

**Saída:**

```
CreatedDate      : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType       : Sha256
LatestVersion   : 1
Name           : RunShellScript
```

```
Owner      : 809632081692
Parameters : {commands}
PlatformTypes : {Linux}
SchemaVersion : 2.0
Sha1       :
Status     : Creating
```

- Para obter detalhes da API, consulte [CreateDocument](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SSMMaintenanceWindow

O código de exemplo a seguir mostra como usar `New-SSMMaintenanceWindow`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cria uma nova janela de manutenção com o nome especificado que é executada às 16h toda terça-feiras por 4 horas, com um limite de 1 hora, e que permite destinos não associados.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Saída:

```
mw-03eb53e1ea7383998
```

- Para obter detalhes da API, consulte [CreateMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-SSMPatchBaseline

O código de exemplo a seguir mostra como usar `New-SSMPatchBaseline`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cria uma lista de referência de patches que aprova patches, sete dias após serem lançados pela Microsoft, para instâncias gerenciadas que executam o Windows Server 2019 em um ambiente de produção.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description "Baseline
containing all updates approved for production systems" -ApprovalRules_PatchRule
$rule
```

Saída:

```
pb-0z4z6221c4296b23z
```

- Para obter detalhes da API, consulte [CreatePatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-SSMDefaultPatchBaseline

O código de exemplo a seguir mostra como usar Register-SSMDefaultPatchBaseline.

## Ferramentas para PowerShell

Exemplo 1: este exemplo registra uma lista de referência de patches como a lista de referência de patches padrão.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
pb-03da896ca3b68b639
```

- Para obter detalhes da API, consulte [RegisterDefaultPatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-SSMPatchBaselineForPatchGroup

O código de exemplo a seguir mostra como usar Register-SSMPatchBaselineForPatchGroup.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo registra uma lista de referência de patches para um grupo de patches.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -  
PatchGroup "Production"
```

Saída:

```
BaselineId          PatchGroup  
-----  
pb-03da896ca3b68b639 Production
```

- Para obter detalhes da API, consulte [RegisterPatchBaselineForPatchGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-SSMTargetWithMaintenanceWindow

O código de exemplo a seguir mostra como usar Register-SSMTargetWithMaintenanceWindow.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo registra uma instância com uma janela de manutenção.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target  
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Exemplo 2: esse exemplo registra várias instâncias com uma janela de manutenção.

```
$option1 =  
  @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target  
$option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Exemplo 3: esse exemplo registra uma instância com uma janela de manutenção usando tags do EC2.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}  
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target  
$option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Saída:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Para obter detalhes da API, consulte [RegisterTargetWithMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-SSMTaskWithMaintenanceWindow

O código de exemplo a seguir mostra como usar Register-SSMTaskWithMaintenanceWindow.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de instância. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
    Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
    @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
    Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 2: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de destino. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
    Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
    @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -TaskType
    "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 3: esse exemplo cria um objeto de parâmetro para o documento de comandos de execução **AWS-RunPowerShellScript** e cria uma tarefa com uma janela de manutenção determinada usando o ID de destino. A saída devolvida é o ID da tarefa.

```
$parameters =
[Collections.Generic.Dictionary[String,Collections.Generic.List[String]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Saída:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Exemplo 4: Este exemplo registra uma tarefa do AWS Systems Manager Automation usando um documento chamado **Create-Snapshots**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{ TARGET_ID }") )
$automationParameters.Add( "AutomationAssumeRole",
    @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456 `
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role" `
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots" `
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" } `
    -TaskType "AUTOMATION" `
```

```
-Priority 4`  
-Automation_DocumentVersion '$DEFAULT' -Automation_Parameter  
$automationParameters -Name "Create-Snapshots"
```

- Para obter detalhes da API, consulte [RegisterTaskWithMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMActivation

O código de exemplo a seguir mostra como usar `Remove-SSMActivation`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exclui uma ativação. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Para obter detalhes da API, consulte [DeleteActivation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMAssociation

O código de exemplo a seguir mostra como usar `Remove-SSMAssociation`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exclui a associação entre uma instância e um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Para obter detalhes da API, consulte [DeleteAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMDocument

O código de exemplo a seguir mostra como usar `Remove-SSMDocument`.



## Ferramentas para PowerShell

Exemplo 1: esse exemplo exclui um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Para obter detalhes da API, consulte [DeleteDocument](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMMaintenanceWindow

O código de exemplo a seguir mostra como usar Remove-SSMMaintenanceWindow.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo remove uma janela de manutenção.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Saída:

```
mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte [DeleteMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMParameter

O código de exemplo a seguir mostra como usar Remove-SSMParameter.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo exclui um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMParameter -Name "helloWorld"
```

- Para obter detalhes da API, consulte [DeleteParameter](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMPatchBaseline

O código de exemplo a seguir mostra como usar Remove-SSMPatchBaseline.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo exclui uma lista de referência de patches.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Saída:

```
pb-045f10b4f382baeda
```

- Para obter detalhes da API, consulte [DeletePatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-SSMResourceTag

O código de exemplo a seguir mostra como usar Remove-SSMResourceTag.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo remove uma tag de uma janela de manutenção. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType  
"MaintenanceWindow" -TagKey "Production"
```

- Para obter detalhes da API, consulte [RemoveTagsFromResource](#) em Referência de AWS Tools for PowerShell cmdlet.

## Send-SSMCommand

O código de exemplo a seguir mostra como usar Send-SSMCommand.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo executa um comando echo em uma instância de destino.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
"echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

### Saída:

```
CommandId      : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment       :
CompletedCount : 0
DocumentName  : AWS-RunPowerShellScript
ErrorCount    : 0
ExpiresAfter  : 3/7/2017 10:48:37 PM
InstanceIds   : {}
MaxConcurrency : 50
MaxErrors     : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix :
OutputS3Region :
Parameters    : {[commands,
  Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime : 3/7/2017 9:48:37 PM
ServiceRole   :
Status        : Pending
StatusDetails : Pending
TargetCount   : 0
Targets       : {instanceids}
```

Exemplo 2: esse exemplo mostra como executar um comando que aceita parâmetros aninhados.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
@{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win320penSSH"}'; "commandLine"=".\\Install-
Win320penSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte [SendCommand](#) em Referência de AWS Tools for PowerShell cmdlet.

## Start-SSMAutomationExecution

O código de exemplo a seguir mostra como usar Start-SSMAutomationExecution.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo executa um documento especificando um perfil do Automation, um ID de origem da AMI e um perfil de instância do Amazon EC2.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -  
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/  
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Saída:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Para obter detalhes da API, consulte [StartAutomationExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-SSMAutomationExecution

O código de exemplo a seguir mostra como usar Stop-SSMAutomationExecution.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo interrompe uma execução do Automation. Não haverá saída se o comando for bem-sucedido.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-  
f944-11e6-9d32-8fb2db27a909"
```

- Para obter detalhes da API, consulte [StopAutomationExecution](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-SSMCommand

O código de exemplo a seguir mostra como usar Stop-SSMCommand.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo tenta cancelar um comando. Não haverá saída se a operação for bem-sucedida.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Para obter detalhes da API, consulte [CancelCommand](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-SSMManagedInstance

O código de exemplo a seguir mostra como usar `Unregister-SSMManagedInstance`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cancela o registro de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Para obter detalhes da API, consulte [DeregisterManagedInstance](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-SSMPatchBaselineForPatchGroup

O código de exemplo a seguir mostra como usar `Unregister-SSMPatchBaselineForPatchGroup`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cancela o registro de um grupo de patches de uma lista de referência de patches.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -  
PatchGroup "Production"
```

Saída:

```
BaselineId          PatchGroup  
-----  
pb-045f10b4f382baeda Production
```

- Para obter detalhes da API, consulte [DeregisterPatchBaselineForPatchGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-SSMTargetFromMaintenanceWindow

O código de exemplo a seguir mostra como usar `Unregister-SSMTargetFromMaintenanceWindow`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo remove um destino de uma janela de manutenção.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

### Saída:

```
WindowId           WindowTargetId
-----
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Para obter detalhes da API, consulte [DeregisterTargetFromMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-SSMTaskFromMaintenanceWindow

O código de exemplo a seguir mostra como usar `Unregister-SSMTaskFromMaintenanceWindow`.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo remove uma tarefa de uma janela de manutenção.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

### Saída:

```
WindowId           WindowTaskId
-----
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Para obter detalhes da API, consulte [DeregisterTaskFromMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMAssociation

O código de exemplo a seguir mostra como usar Update-SSMAssociation.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza uma associação com uma nova versão de documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Saída:

```
Name           : AWS-UpdateSSMAgent
InstanceId      :
Date           : 3/1/2017 6:22:21 PM
Status.Name     :
Status.Date     :
Status.Message  :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte [UpdateAssociation](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMAssociationStatus

O código de exemplo a seguir mostra como usar Update-SSMAssociationStatus.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza o status da associação entre uma instância e um documento de configuração.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
"i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
-AssociationStatus_Name "Pending" -AssociationStatus_Message
"temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Saída:

```
Name           : AWS-UpdateSSMAgent
InstanceId      : i-0000293ffd8c57862
Date           : 2/23/2017 6:55:22 PM
Status.Name    : Pending
Status.Date    : 2/20/2015 8:31:11 AM
Status.Message : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte [UpdateAssociationStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMDocument

O código de exemplo a seguir mostra como usar Update-SSMDocument.

### Ferramentas para PowerShell

Exemplo 1: isso cria uma nova versão de um documento com o conteúdo atualizado do arquivo json que você especificar. O documento deve estar em formato JSON. Você pode obter a versão do documento com o cmdlet "Get-SSMDocumentVersionList".

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

### Saída:

```
CreatedDate      : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType       : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters     : {commands}
PlatformTypes  : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status         : Updating
```



- Para obter detalhes da API, consulte [UpdateDocument](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMDocumentDefaultVersion

O código de exemplo a seguir mostra como usar Update-SSMDocumentDefaultVersion.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza a versão padrão de um documento. Você pode obter as versões disponíveis do documento com o cmdlet "Get-SSMDocumentVersionList".

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Saída:

```
DefaultVersion Name
-----
2                RunShellScript
```

- Para obter detalhes da API, consulte [UpdateDocumentDefaultVersion](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMMaintenanceWindow

O código de exemplo a seguir mostra como usar Update-SSMMaintenanceWindow.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza o nome de uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
```

```
Enabled           : True
Name              : My-Renamed-MW
Schedule          : cron(0 */30 * * * ? *)
WindowId         : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo habilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 3: esse exemplo desabilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte [UpdateMaintenanceWindow](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMManagedInstanceRole

O código de exemplo a seguir mostra como usar Update-SSMManagedInstanceRole.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo atualiza o perfil de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
"AutomationRole"
```

- Para obter detalhes da API, consulte [UpdateManagedInstanceRole](#) em Referência de AWS Tools for PowerShell cmdlet.

## Update-SSMPatchBaseline

O código de exemplo a seguir mostra como usar Update-SSMPatchBaseline.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo adiciona dois patches como rejeitados e um patch como aprovado a uma lista de referência de patches existente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
"KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Saída:

```
ApprovalRules      : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches    : {KB2124261}
BaselineId         : pb-03da896ca3b68b639
CreatedDate        : 3/3/2017 5:02:19 PM
Description         : Baseline containing all updates approved for production systems
GlobalFilters      : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate       : 3/3/2017 5:22:10 PM
Name               : Production-Baseline
RejectedPatches    : {KB2032276, MS10-048}
```

- Para obter detalhes da API, consulte [UpdatePatchBaseline](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-SSMComplianceItem

O código de exemplo a seguir mostra como usar Write-SSMComplianceItem.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo grava um item de conformidade personalizado para a instância gerenciada especificada

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
  Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
  ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Para obter detalhes da API, consulte [PutComplianceItems](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-SSMInventory

O código de exemplo a seguir mostra como usar Write-SSMInventory.

## Ferramentas para PowerShell

Exemplo 1: esse exemplo atribui informações de localização de rack a uma instância. Não haverá saída se o comando for bem-sucedido.

```
$data = New-Object
  "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
  "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
  System.String]]"
$items.Add($data)

$customInventoryItem = New-Object Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)
```

```
Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Para obter detalhes da API, consulte [PutInventory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Write-SSMParameter

O código de exemplo a seguir mostra como usar Write-SSMParameter.

### Ferramentas para PowerShell

Exemplo 1: esse exemplo cria um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Exemplo 2: esse exemplo altera um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -  
Overwrite $true
```

- Para obter detalhes da API, consulte [PutParameter](#) em Referência de AWS Tools for PowerShell cmdlet.

## Exemplos do Amazon Translate usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell com o Amazon Translate.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### ConvertTo-TRNTargetLanguage

O código de exemplo a seguir mostra como usar ConvertTo-TRNTargetLanguage.

#### Ferramentas para PowerShell

Exemplo 1: converte o texto em inglês especificado em francês. O texto a ser convertido também pode ser passado como o parâmetro -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -  
TargetLanguageCode fr
```

- Para obter detalhes da API, consulte [TranslateText](#) em Referência de AWS Tools for PowerShell cmdlet.

## AWS WAFV2 exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with AWS WAFV2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### New-WAF2WebACL

O código de exemplo a seguir mostra como usar New-WAF2WebACL.

#### Ferramentas para PowerShell

Exemplo 1: Esse comando cria uma nova ACL da web chamada “waf-test”. Observe que, de acordo com a documentação da API de serviço, 'DefaultAction' é uma propriedade obrigatória. Portanto, o valor de '- DefaultAction \_Allow' e/ou '- DefaultAction \_Block' deve ser especificado. Como '- DefaultAction \_Allow' e '- DefaultAction \_Block' não são as propriedades obrigatórias, o valor '@ {}' pode ser usado como espaço reservado, conforme mostrado no exemplo acima.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-test" -Description "Test" -DefaultAction_Allow @{}
```

#### Saída:

```
ARN          : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description  : Test
Id           : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken    : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name         : waf-test
```

- Para obter detalhes da API, consulte [CreateWebAcl](#) em Referência de AWS Tools for PowerShell cmdlet.

## WorkSpaces exemplos usando ferramentas para PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS Tools for PowerShell with WorkSpaces.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Ações](#)

## Ações

### Approve-WKSIpRule

O código de exemplo a seguir mostra como usar Approve-WKSIpRule.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona regras a um grupo IP existente

```
$Rule = @(
@{IPRule = "10.1.0.0/0"; RuleDesc = "First Rule Added"},
@{IPRule = "10.2.0.0/0"; RuleDesc = "Second Rule Added"}
)

Approve-WKSIpRule -GroupId wsipg-abcnx2fcw -UserRule $Rule
```

- Para obter detalhes da API, consulte [AuthorizeIpRules](#) em Referência de AWS Tools for PowerShell cmdlet.

### Copy-WKSWorkspaceImage

O código de exemplo a seguir mostra como usar Copy-WKSWorkspaceImage.

#### Ferramentas para PowerShell

Exemplo 1: Este exemplo copia a imagem do espaço de trabalho com o ID especificado de us-west-2 para a região atual com o nome "" CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId wsi-djfoedhw6
```



**Saída:**

```
wsi-456abaqfe
```

- Para obter detalhes da API, consulte [CopyWorkspacelImage](#) em Referência de AWS Tools for PowerShell cmdlet.

**Edit-WKSClientProperty**

O código de exemplo a seguir mostra como usar `Edit-WKSClientProperty`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo permite a reconexão para o cliente do Workspaces

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled  
"ENABLED" -ResourceId d-123414a369
```

- Para obter detalhes da API, consulte [ModifyClientProperties](#) em Referência de AWS Tools for PowerShell cmdlet.

**Edit-WKSSelfServicePermission**

O código de exemplo a seguir mostra como usar `Edit-WKSSelfServicePermission`.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo permite permissões de autoatendimento para alterar o tipo de computação e aumentar o tamanho do volume para o diretório especificado

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId  
d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -  
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Para obter detalhes da API, consulte [ModifySelfservicePermissions](#) em Referência de AWS Tools for PowerShell cmdlet.

**Edit-WKSWorkspaceAccessProperty**

O código de exemplo a seguir mostra como usar `Edit-WKSWorkspaceAccessProperty`.

## Ferramentas para PowerShell

Exemplo 1: este exemplo permite o acesso ao Workspace no Android e no Chrome OS para o diretório especificado

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -  
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Para obter detalhes da API, consulte [ModifyWorkspaceAccessProperties](#) em Referência de AWS Tools for PowerShell cmdlet.

### Edit-WKSWorkspaceCreationProperty

O código de exemplo a seguir mostra como usar Edit-WKSWorkspaceCreationProperty.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo permite que o Modo de Acesso e Manutenção à Internet seja verdadeiro como valores padrão ao criar um espaço de trabalho

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId  
d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -  
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Para obter detalhes da API, consulte [ModifyWorkspaceCreationProperties](#) em Referência de AWS Tools for PowerShell cmdlet.

### Edit-WKSWorkspaceProperty

O código de exemplo a seguir mostra como usar Edit-WKSWorkspaceProperty.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo altera a propriedade do modo de execução do espaço de trabalho para parada automática para o espaço de trabalho especificado

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -  
WorkspaceProperties_RunningMode AUTO_STOP
```

- Para obter detalhes da API, consulte [ModifyWorkspaceProperties](#) em Referência de AWS Tools for PowerShell cmdlet.

## Edit-WKSSpaceState

O código de exemplo a seguir mostra como usar `Edit-WKSSpaceState`.

### Ferramentas para PowerShell

Exemplo 1: Esse exemplo altera o estado do espaço de trabalho especificado para Disponível

```
Edit-WKSSpaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState AVAILABLE
```

- Para obter detalhes da API, consulte [ModifyWorkspaceState](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSSClientProperty

O código de exemplo a seguir mostra como usar `Get-WKSSClientProperty`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém as propriedades do cliente do Workspace Client para o diretório especificado

```
Get-WKSSClientProperty -ResourceId d-223562a123
```

- Para obter detalhes da API, consulte [DescribeClientProperties](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSSIPGroup

O código de exemplo a seguir mostra como usar `Get-WKSSIPGroup`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo obtém os detalhes do grupo IP especificado na região especificada

```
Get-WKSSIPGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

**Saída:**

```

GroupDesc GroupId      GroupName UserRules
-----
wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
Amazon.WorkSpaces.Model.IpRuleItem}

```

- Para obter detalhes da API, consulte [DescribeIpGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-WKSTag**

O código de exemplo a seguir mostra como usar Get-WKSTag.

**Ferramentas para PowerShell**

Exemplo 1: Este exemplo busca a tag para o espaço de trabalho fornecido

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

**Saída:**

```

Key      Value
---      -
auto-delete no
purpose  Workbench

```

- Para obter detalhes da API, consulte [DescribeTags](#) em Referência de AWS Tools for PowerShell cmdlet.

**Get-WKSWorkspace**

O código de exemplo a seguir mostra como usar Get-WKSWorkspace.

**Ferramentas para PowerShell**

Exemplo 1: recupera detalhes de todos os seus WorkSpaces para o pipeline.

```
Get-WKSWorkspace
```

**Saída:**

```
BundleId           : wsb-1a2b3c4d
ComputerName       :
DirectoryId        : d-1a2b3c4d
ErrorCode          :
ErrorMessage       :
IpAddress          :
RootVolumeEncryptionEnabled : False
State              : PENDING
SubnetId           :
UserName           : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey :
WorkspaceId        : ws-1a2b3c4d
WorkspaceProperties : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Exemplo 2: Esse comando mostra os valores das propriedades secundárias

**WorkspaceProperties** de um espaço de trabalho na **us-west-2** região. Para obter mais informações sobre as propriedades secundárias do **WorkspaceProperties**, consulte [https://docs.aws.amazon.com/workspaces/latest/api/API\\_WorkspaceProperties.html](https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html).

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties
```

Saída:

```
ComputeTypeName      : STANDARD
RootVolumeSizeGib    : 80
RunningMode          : AUTO_STOP
RunningModeAutoStopTimeoutInMinutes : 60
UserVolumeSizeGib    : 50
```

Exemplo 3: Esse comando mostra o valor da propriedade filha **RootVolumeSizeGib** **WorkspaceProperties** de um espaço de trabalho na **us-west-2** região. O tamanho do volume raiz, em GiB, é 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Saída:

```
80
```

- Para obter detalhes da API, consulte [DescribeWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSWorkspaceBundle

O código de exemplo a seguir mostra como usar Get-WKSWorkspaceBundle.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca detalhes de todos os pacotes do Workspace na região atual

```
Get-WKSWorkspaceBundle
```

Saída:

```
BundleId       : wsb-sfhdgv342
ComputeType    : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId        : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name           : CustomBundleTest
Owner          : 233816212345
RootStorage    : Amazon.WorkSpaces.Model.RootStorage
UserStorage    : Amazon.WorkSpaces.Model.UserStorage
```

- Para obter detalhes da API, consulte [DescribeWorkspaceBundles](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSWorkspaceDirectory

O código de exemplo a seguir mostra como usar Get-WKSWorkspaceDirectory.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo lista os detalhes dos diretórios registrados

```
Get-WKSWorkspaceDirectory
```

Saída:

```

Alias                : TestWorkspace
CustomerUserName    : Administrator
DirectoryId         : d-123414a369
DirectoryName       : TestDirectory.com
DirectoryType       : MicrosoftAD
DnsIpAddresses      : {172.31.43.45, 172.31.2.97}
IamRoleId           : arn:aws:iam::761234567801:role/workspaces_RoleDefault
IpGroupIds          : {}
RegistrationCode    : WSpdx+4RRT43
SelfservicePermissions : Amazon.WorkSpaces.Model.SelfservicePermissions
State               : REGISTERED
SubnetIds           : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy             : SHARED
WorkspaceAccessProperties : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
  Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId : sg-0ed2441234a123c43

```

- Para obter detalhes da API, consulte [DescribeWorkspaceDirectories](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSWorkspaceImage

O código de exemplo a seguir mostra como usar Get-WKSWorkspaceImage.

### Ferramentas para PowerShell

Exemplo 1: esta amostra busca todos os detalhes de todas as imagens na região

```
Get-WKSWorkspaceImage
```

Saída:

```

Description      :This image is copied from another image
ErrorCode        :
ErrorMessage     :
ImageId          : wsi-345ahdjgo
Name             : CopiedImageTest
OperatingSystem  : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy  : DEFAULT
State            : AVAILABLE

```

- Para obter detalhes da API, consulte [DescribeWorkspaceImages](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSWorkspaceSnapshot

O código de exemplo a seguir mostra como usar Get-WKSWorkspaceSnapshot.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo mostra o timestamp do snapshot mais recente criado para o espaço de trabalho especificado

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Saída:

```
RebuildSnapshots          RestoreSnapshots
-----
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Para obter detalhes da API, consulte [DescribeWorkspaceSnapshots](#) em Referência de AWS Tools for PowerShell cmdlet.

## Get-WKSWorkspacesConnectionStatus

O código de exemplo a seguir mostra como usar Get-WKSWorkspacesConnectionStatus.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo busca o status da conexão para o espaço de trabalho especificado

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Para obter detalhes da API, consulte [DescribeWorkspacesConnectionStatus](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-WKSIpGroup

O código de exemplo a seguir mostra como usar New-WKSIpGroup.



## Ferramentas para PowerShell

Exemplo 1: Esse exemplo cria um grupo de IP vazio chamado FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Saída:

```
wsipg-w45rty4ty
```

- Para obter detalhes da API, consulte [CreatelpGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-WKSTag

O código de exemplo a seguir mostra como usar New-WKSTag.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo adiciona uma nova tag a um espaço de trabalho chamado **wsname**. A tag tem uma chave de "Nome" e um valor-chave de **AWS\_Workspace**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Exemplo 2: Este exemplo adiciona várias tags a um espaço de trabalho chamado **wsname**. Uma tag tem uma chave de "Nome" e um valor-chave de **AWS\_Workspace**; a outra tag tem uma chave de tag de "Estágio" e um valor-chave de "Teste".

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Para obter detalhes da API, consulte [CreateTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## New-WKSWorkspace

O código de exemplo a seguir mostra como usar New-WKSWorkspace.

### Ferramentas para PowerShell

Exemplo 1: Crie um Workspace para o pacote, o diretório e o usuário fornecidos.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Exemplo 2: Este exemplo cria vários WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Para obter detalhes da API, consulte [CreateWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-WKSIpGroup

O código de exemplo a seguir mostra como usar Register-WKSIpGroup.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo registra o grupo IP especificado com o diretório especificado

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Para obter detalhes da API, consulte [AssociateIpGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

## Register-WKSWorkspaceDirectory

O código de exemplo a seguir mostra como usar Register-WKSWorkspaceDirectory.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo registra o diretório especificado para o Workspaces Service

```
Register-WKWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Para obter detalhes da API, consulte [RegisterWorkspaceDirectory](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-WKSIpGroup

O código de exemplo a seguir mostra como usar Remove-WKSIpGroup.

## Ferramentas para PowerShell

Exemplo 1: Este exemplo exclui o grupo IP especificado

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSIpGroup (DeleteIpGroup)" on target
"wsipg-32fhgtred".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteIpGroup](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-WKSTag

O código de exemplo a seguir mostra como usar Remove-WKSTag.

## Ferramentas para PowerShell

Exemplo 1: Esse exemplo remove a tag associada ao espaço de trabalho

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

## Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- Para obter detalhes da API, consulte [DeleteTags](#) em Referência de AWS Tools for PowerShell cmdlet.

## Remove-WKSWorkspace

O código de exemplo a seguir mostra como usar Remove-WKSWorkspace.

### Ferramentas para PowerShell

Exemplo 1: Encerra vários WorkSpaces. O uso da opção -Force impede que o cmdlet solicite confirmação.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Exemplo 2: recupera a coleção de todas as suas WorkSpaces e canaliza as IDs para o WorkspaceId parâmetro - de Remove-WKSworkspace, encerrando todas as WorkSpaces. O cmdlet avisará antes que cada um Workspace seja encerrado. Para suprimir o prompt de confirmação, adicione a opção -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar TerminateRequest objetos definindo o WorkSpaces a ser encerrado. O cmdlet solicitará a confirmação antes de continuar, a menos que o parâmetro -Force switch também seja especificado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
```

```
$arrRequest += $request2  
Remove-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte [TerminateWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Reset-WKSWorkspace

O código de exemplo a seguir mostra como usar `Reset-WKSWorkspace`.

### Ferramentas para PowerShell

Exemplo 1: reconstrói o especificado. `WorkSpace`

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: recupera a coleção de todos os seus `WorkSpaces` e canaliza os IDs para o `WorkspaceId` parâmetro - de `Reset-WKSworkspace`, fazendo com que o seja reconstruído.  
`WorkSpaces`

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Para obter detalhes da API, consulte [RebuildWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Restart-WKSWorkspace

O código de exemplo a seguir mostra como usar `Restart-WKSWorkspace`.

### Ferramentas para PowerShell

Exemplo 1: Reinicializa o especificado `WorkSpace`.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: Reinicializa várias `WorkSpaces`.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Exemplo 3: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Restart-WKSworkspace, fazendo com que o seja reiniciado.  
WorkSpaces

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Para obter detalhes da API, consulte [RebootWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Stop-WKSWorkspace

O código de exemplo a seguir mostra como usar Stop-WKSWorkspace.

### Ferramentas para PowerShell

Exemplo 1: Interrompe várias WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Stop-WKSworkspace, fazendo com que o seja interrompido.  
WorkSpaces

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar StopRequest objetos definindo o WorkSpaces a ser parado.

```
$arrRequest = @()  
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest  
$request1.WorkspaceId = 'ws-12345678'  
$arrRequest += $request1  
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest  
$request2.WorkspaceId = 'ws-abcdefgh'  
$arrRequest += $request2  
Stop-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte [StopWorkspaces](#) em Referência de AWS Tools for PowerShell cmdlet.

## Unregister-WKSIpGroup

O código de exemplo a seguir mostra como usar `Unregister-WKSIpGroup`.

### Ferramentas para PowerShell

Exemplo 1: Este exemplo cancela o registro do grupo IP especificado do diretório especificado

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Para obter detalhes da API, consulte [DisassociateIpGroups](#) em Referência de AWS Tools for PowerShell cmdlet.

# Segurança para este AWS produto ou serviço

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS, você 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 você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa todos os serviços oferecidos na AWS nuvem e fornecer serviços que você possa usar com segurança. Nossa responsabilidade de segurança é a maior prioridade em AWS, e a eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [Programas de AWS Conformidade](#).

Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você está usando e por outros fatores, incluindo a sensibilidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

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

## Tópicos

- [Proteção de dados neste produto ou serviço da AWS](#)
- [Identity and Access Management](#)
- [Validação de conformidade para este AWS produto ou serviço](#)
- [Aplicar uma versão mínima do TLS nas ferramentas para PowerShell](#)
- [Considerações adicionais de segurança para as 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 as Conta 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, recomendamos 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](#)
- [Autenticando com identidades](#)
- [Como gerenciar acesso usando políticas](#)
- [Como Serviços da AWS funcionam com o IAM](#)
- [Solução de problemas de identidade e acesso do AWS](#)

## 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, consulte [Assinar AWSsolicitações de API da](#) no Guia 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 [Altere 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 e 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 as Organizações e SCPs, consulte [How SCPs work \(Como os SCPs funcionam\)](#) no AWS 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 avaliação](#) de 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 `aws:GetWidget` fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws: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 `aws: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 AWS produto ou serviço

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte [Serviços da AWS Escopo por Programa de Conformidade](#) [Serviços da AWS](#) e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte [Baixar relatórios em AWS Artifact](#) .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- [Guias de início rápido sobre segurança e conformidade](#) — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- [Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services](#) — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

### Note

Nem todos Serviços da AWS são elegíveis para a HIPAA. Para obter mais informações, consulte [Referência dos Serviços Qualificados pela HIPAA](#).

- AWS Recursos de <https://aws.amazon.com/compliance/resources/> de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.

- [AWS Guias de conformidade do cliente](#) — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção 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)).
- [Avaliação de recursos com regras](#) no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- [AWS Security Hub](#) — Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno 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 com suporte e controles aceitos, consulte a [Referência de controles do Security Hub](#).
- [Amazon GuardDuty](#) — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- [AWS Audit Manager](#) — Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

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

# Considerações adicionais de segurança para as Ferramentas para PowerShell

Este tópico contém considerações de segurança, além dos tópicos de segurança abordados nas seções anteriores.

## Registro de informações confidenciais

Algumas operações dessa ferramenta podem retornar informações que podem ser consideradas confidenciais, incluindo informações de variáveis de ambiente. A exposição dessas informações pode representar um risco de segurança em determinados cenários; por exemplo, as informações podem ser incluídas nos registros de integração contínua e implantação contínua (CI/CD). Portanto, é importante que você revise quando está incluindo essa saída como parte de seus registros e suprima a saída quando não for necessária. Para obter informações adicionais sobre a proteção de dados confidenciais, consulte [Proteção de dados neste produto ou serviço da AWS](#).

Considere as seguintes práticas recomendadas:

- Não use variáveis de ambiente para armazenar valores confidenciais para seus recursos sem servidor. Em vez disso, faça com que seu código sem servidor recupere programaticamente o segredo de um armazenamento de segredos (por exemplo,). AWS Secrets Manager
- Analise o conteúdo dos seus registros de compilação para garantir que eles não contenham informações confidenciais. Considere abordagens como canalizar para /dev/null ou capturar a saída como bash ou variável para suprimir as saídas do comando. PowerShell
- Considere o acesso aos seus registros e defina o escopo do acesso de acordo com seu caso de uso.

## 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
<a href="#">Exemplos de código</a>	Incluiu um capítulo com exemplos de cmdlets.	17 de abril de 2024
<a href="#">Considerações adicionais de segurança</a>	Informações incluídas sobre o possível registro de dados confidenciais.	16 de abril de 2024
<a href="#">Configure a autenticação da ferramenta com AWS</a>	Foram adicionadas informações sobre o suporte para SSO no AWS Tools for PowerShell.	15 de março de 2024
<a href="#">Referência de cmdlet para as Ferramentas para PowerShell</a>	Seção adicionada com um link para a referência do PowerShell cmdlet Tools for.	17 de novembro de 2023
<a href="#">Inclusão de mais atualizações de práticas recomendadas do IAM</a>	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte <a href="#">Práticas recomendadas de segurança no IAM</a> .	12 de outubro de 2023
<a href="#">Instalar no Windows</a>	Foram removidas as informações sobre a instalação do Tools for Windows	25 de setembro de 2023



---

	PowerShell usando o MSI, que se tornou obsoleto.	
<a href="#">Atualizações de práticas recomendadas do IAM</a>	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte <a href="#">Práticas recomendadas de segurança no IAM</a> .	8 de setembro de 2023
<a href="#">Pipelining e \$ AWSHistory</a>	O parâmetro <code>IncludeSensitiveData</code> foi adicionado ao cmdlet <code>Set-AWSHistoryConfiguration</code> .	9 de março de 2023
<a href="#">Usando o ClientConfig parâmetro em cmdlets</a>	Foram adicionadas informações sobre o suporte para o ClientConfig parâmetro.	28 de outubro de 2022
<a href="#">Execute uma instância do Amazon EC2 usando o Windows PowerShell</a>	Adicionadas notas sobre a retirada do EC2-Classic.	26 de julho de 2022
<a href="#">AWS Tools for PowerShell Versão 4</a>	Adicionadas informações sobre a versão 4, incluindo instruções de instalação para <a href="#">Windows</a> e <a href="#">Linux/macOS</a> e um tópico sobre <a href="#">migração</a> 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](#)

15 de maio de 2013

A saída da coleção dos cmdlets é sempre enumerada no pipeline. PowerShell Suporte automático para chamadas de serviço pagináveis. A nova variável \$ AWSHistory shell coleta respostas de serviço e, opcionalmente, solicitações de serviço. AWSRegion as instâncias usam o campo Região em vez de SystemName para ajudar no pipeline. Remove-S3 Bucketsuporta a opção - DeleteObjects switch. Problema de usabilidade corrigido com Set-AWSCredentials Inicializar - AWSDefaults relata de onde obteve credenciais e dados da região. Stop-EC2Instanceac eita instâncias de Amazon.ec 2.model.Reservation como entrada. Tipos de parâmetro Generic List<T> substituídos por tipos de matriz (T[]). Cmdlets que excluem ou encerram o prompt de recursos para confirmação antes da exclusão. Write-S3Objectsuporta conteúdo de texto em linha para upload no Amazon S3.

## [AWS Tools for Windows PowerShell 1.0.1.0](#)

21 de dezembro de 2012

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 com `Initialize-AWSDefaults`. Para sistemas com Windows PowerShell versão 3, o atalho do menu Iniciar é atualizado para remover o `Import-Module` comando, deixando apenas `Initialize`

`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.