ユーザーガイド

AWS Tools for PowerShell

AWS Tools for PowerShell: ユーザーガイド

Table of Contents

AWS Tools for PowerShellとは何ですか?

AWS Tools for PowerShell は、 によって公開される機能に基づいて構築された PowerShell モジュールのセットです AWS SDK for .NET。 AWS Tools for PowerShell を使用すると、コマンドラインから AWS リソースに対するオペレーションを PowerShellスクリプト化できます。

コマンドレットは、さまざまな AWS サービス HTTP クエリ APIs を使用して実装されている場合でも、パラメータを指定し、結果を処理するためのイディオマティックな PowerShell エクスペリエンスを提供します。例えば、 AWS Tools for PowerShell サポート PowerShell パイプラインのコマンドレット、つまりコマンドレットに出入りする PowerShell オブジェクトをパイプできます。

AWS Tools for PowerShell は、 AWS Identity and Access Management (IAM) インフラストラクチャのサポートなど、認証情報の処理方法に柔軟性があります。これらのツールは、IAM ユーザーの認証情報、一時的なセキュリティトークン、IAM ロールとともに使用できます。

は、 SDK でサポートされているのと同じサービスと AWS リージョンのセット AWS Tools for PowerShell をサポートします。Windows、Linux、または macOS オペレーティングシステムを実行しているコンピュータ AWS Tools for PowerShell に をインストールできます。

> **Note**
>
> AWS Tools for PowerShell バージョン 4 は最新のメジャーリリースであり、 AWS Tools for PowerShell バージョン 3.3 への下位互換性がある更新です。既存のコマンドレットの動作を維持しながら、大幅な機能強化を追加します。新しいバージョンにアップグレードした後も、既存のスクリプトは引き続き動作しますが、アップグレードする前に十分にテストすることをお勧めします。バージョン 4 での変更点の詳細については、「AWS Tools for PowerShell バージョン 3.3 からバージョン 4 への移行」を参照してください。

AWS Tools for PowerShell は、次の 3 つの異なるパッケージとして利用できます。

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

SDK メジャーバージョンのメンテナンスとサポート

SDK メジャーバージョンのメンテナンスとサポート、およびその基礎的な依存関係については、AWS SDK とツール共有設定および認証情報リファレンスガイドで以下を参照してください。

- AWS SDKsメンテナンスポリシー
- AWS SDKsとツールのバージョンサポートマトリックス

`AWS.Tools` - のモジュール化されたバージョン AWS Tools for PowerShell

このバージョンの AWS Tools for PowerShell は、本番環境 PowerShell で実行されているすべてのコンピュータに推奨されるバージョンです。モジュール化されているため、使用するサービスのモジュールのみをダウンロードしてロードする必要があります。これにより、ダウンロード時間とメモリ使用量が削減されます。ほとんどの場合、最初に `Import-Module` を手動で呼び出すことなく `AWS.Tools` コマンドレットの自動インポートが可能になります。

これは の最新バージョン AWS Tools for PowerShell であり、Windows、Linux、macOS など、サポートされているすべてのオペレーティングシステムで実行されます。このパッケージは、サービスごとに 1 つのインストールモジュール`AWS.Tools.Installer`、、1 `AWS.Tools.Common`つの共通モジュール AWS 、`AWS.Tools.EC2`、 `AWS.Tools.IdentityManagement` `AWS.Tools.S3`などを提供します。

`AWS.Tools.Installer` モジュールには、各 AWS サービスのモジュールをインストール、更新、削除できるコマンドレットが用意されています。このモジュールのコマンドレットを利用することで、使用するモジュールをサポートするために必要なすべての依存モジュールが自動的に取得されます。

`AWS.Tools.Common` モジュールには、サービス固有ではない設定および認証のコマンドレットが用意されています。 AWS サービスのコマンドレットを使用するには、 コマンドを実行するだけです。 PowerShell は、コマンドレットを実行する AWS サービスの`AWS.Tools.Common`モジュールとモジュールを自動的にインポートします。このモジュールは、`AWS.Tools.Installer` モジュールを使用してサービスモジュールをインストールすると、自動的にインストールされます。

このバージョンの は、実行中のコンピュータ AWS Tools for PowerShell にインストールできます。

- PowerShell Windows、Linux、または macOS の Core 6.0 以降。
- .NET Framework 4.7.2 以降を搭載した Windows の Windows PowerShell 5.1 以降。

このガイドでは、このバージョンのみを指定する必要がある場合は、モジュール名 *AWS.Tools* で参照します。

AWSPowerShellNetCore .- の単一モジュールバージョン AWS Tools for PowerShell

このバージョンは、すべての AWS サービスのサポートを含む単一の大きなモジュールで構成されています。このモジュールを使用する前に、手動でインポートする必要があります。

このバージョンの は、実行中のコンピュータ AWS Tools for PowerShell にインストールできます。

- PowerShell Windows、Linux、または macOS の Core 6.0 以降。
- .NET Framework 4.7.2 以降を搭載した Windows の Windows PowerShell 3.0 以降。

このガイドでは、このバージョンのみを指定する必要がある場合は、モジュール名 で参照しますAWSPowerShell。NetCore

AWSPowerShell - Windows 用の単一モジュールバージョン PowerShell

このバージョンの AWS Tools for PowerShell は と互換性があり、Windows PowerShell バージョン 2.0 から 5.1 を実行している Windows コンピュータにのみインストールできます。 PowerShell Core 6.0 以降、またはその他のオペレーティングシステム (Linux または macOS) と互換性がありませ

ん。このバージョンは、すべての AWS サービスのサポートを含む単一の大きなモジュールで構成されています。

このガイドでは、このバージョンのみを指定する必要がある場合は、モジュール名 で参照しますAWSPowerShell。

このガイドの使い方

このガイドは、大きく次のセクションに分かれています。

AWS Tools for PowerShell のインストール

このセクションでは、 をインストールする方法について説明します AWS Tools for PowerShell。これには、アカウントをまだお持ち AWS でない場合は にサインアップする方法と、コマンドレットの実行に使用できる IAM ユーザーを作成する方法が含まれます。

AWS Tools for Windows PowerShell の開始方法

このセクションでは、認証情報と AWS リージョンの指定 AWS Tools for PowerShell、特定のサービスのコマンドレットの検索、コマンドレットのエイリアスの使用など、 の使用の基本について説明します。

AWS Tools for PowerShell での AWS サービスの操作

このセクションでは、 を使用して最も一般的な AWS タスクの一部 AWS Tools for PowerShell を実行する方法について説明します。

AWS Tools for PowerShell のインストール

AWS Tools for PowerShell コマンドレットを正しくインストールして使用するには、以下のトピックの手順を参照してください。

トピック

Windows AWS Tools for PowerShell への のインストール

Windows ベースのコンピュータでは、次の AWS Tools for PowerShell パッケージオプションのいずれかを実行できます。

- **`AWS.Tools`** - のモジュール化されたバージョン AWS Tools for PowerShell。各 AWS サービスは、個別の小さなモジュールでサポート`AWS.Tools.Common`され、共有サポートモジュールと があります`AWS.Tools.Installer`。
- AWSPowerShell.NetCore - の単一のラージモジュールバージョン AWS Tools for PowerShell。すべての AWS サービスは、この単一の大きなモジュールでサポートされています。

  > **Note**
  >
  > 単一のモジュールは大きすぎて AWS Lambda 関数で使用できない場合があることに注意してください。代わりに、上記のモジュール化されたバージョンを使用します。

- AWSPowerShell - Windows 固有の大きな単独モジュールのレガシーバージョンの AWS Tools for PowerShell。すべての AWS サービスは、この単一の大きなモジュールでサポートされています。

選択するパッケージは、実行している Windows のリリースとエディションによって異なります。

> **Note**
>
> Tools for Windows PowerShell （ AWSPowerShell モジュール) は、すべての Windows ベースの Amazon マシンイメージ (AMIs。

のセットアップ AWS Tools for PowerShell には、このトピックで詳しく説明されている以下の大まかなタスクが含まれます。

1. 環境に適した AWS Tools for PowerShell パッケージオプションをインストールします。
2. `Get-ExecutionPolicy` コマンドレットを実行して、スクリプトの実行が有効になっていることを確認します。
3. AWS Tools for PowerShell モジュールを PowerShell セッションにインポートします。

前提条件

PowerShell Core PowerShellを含む の新しいバージョンは、Microsoft のウェブサイトでの のさまざまなバージョンのインストール PowerShellで Microsoft からのダウンロードとして利用できます。

Windows に **`AWS.Tools`** をインストールする

モジュール化されたバージョンの は、Windows PowerShell 5.1 または PowerShell Core 6.0 以降で Windows を実行しているコンピュータ AWS Tools for PowerShell にインストールできます。Core のインストール PowerShell方法については、Microsoft の Web サイトに「 のさまざまなバージョン PowerShellをインストールする」を参照してください。

次の 3 つの方法のいずれかで `AWS.Tools` をインストールできます。

- `AWS.Tools.Installer` モジュール内のコマンドレットを使用する。このモジュールは、他のAWS.Toolsモジュールのインストールと更新を簡素化します。 AWS.Tools.Installerには が必要でPowerShellGet、更新されたバージョンを自動的にダウンロードしてインストールします。 はモジュールバージョンAWS.Tools.Installerを自動的に同期させます。1 つのモジュールの新しいバージョンをインストールまたは更新すると、 のコマンドレットによって他のすべてのAWS.Toolsモジュールが同じバージョンAWS.Tools.Installerに自動的に更新されます。

  この方法については、以下の手順で説明します。
- `AWS.Tools.zip` からモジュールをダウンロードし、モジュールフォルダの 1 つにそのモジュールを展開する。PSModulePath 環境変数の値を表示することで、モジュールフォルダを検出できます。
- コマンドレットを使用して PowerShell Gallery `Install-Module` から各サービスモジュールをインストールします。

AWS.Tools.Installer モジュールを使用して Windows **AWS.Tools**に をインストールするには

1. PowerShell セッションを開始します。

   > **Note**
   >
   > タスクが必要とする場合を除き、昇格されたアクセス許可を持つ管理者 PowerShell として を実行しないことをお勧めします。これは、潜在的なセキュリティリスクを避けるためであり、最小限の特権の原則にも反します。

2. モジュール化された AWS.Tools パッケージをインストールするには、次のコマンドを実行します。

```
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
```

   リポジトリについて「信頼されていない」という通知を受けた場合は、これをインストールするかどうかを確認するメッセージが表示されます。**y** を入力して、 PowerShell がモジュールをインストールできるようにします。プロンプトを回避し、リポジトリを信頼せずにモジュールをインストールするには、-Force パラメータを指定してコマンドを実行します。

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

3. Install-AWSToolsModule コマンドレットを使用して、使用する各 AWS サービスのモジュールをインストールできるようになりました。例えば、次のコマンドは Amazon EC2 モジュールと Amazon S3 モジュールをインストールします。このコマンドは、指定したモジュールの動作に必要な依存モジュールもインストールします。たとえば、最初の AWS.Tools サービスモジュールをインストールすると、AWS.Tools.Common もインストールされます。これは、すべての AWS サービスモジュールに必要な共有モジュールです。また、古いバージョンのモジュールを削除し、他のモジュールを同じ新しいバージョンに更新します。

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
```

```
 Confirm
 Are you sure you want to perform this action?
 Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
4.0.0.0".
 [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):

 Installing module AWS.Tools.Common version 4.0.0.0
 Installing module AWS.Tools.EC2 version 4.0.0.0
 Installing module AWS.Tools.Glacier version 4.0.0.0
 Installing module AWS.Tools.S3 version 4.0.0.0

 Uninstalling AWS.Tools version 3.3.618.0
 Uninstalling module AWS.Tools.Glacier
 Uninstalling module AWS.Tools.S3
 Uninstalling module AWS.Tools.SimpleNotificationService
 Uninstalling module AWS.Tools.SQS
 Uninstalling module AWS.Tools.Common
```

> **Note**
>
> `Install-AWSToolsModule` コマンドレットは、すべての要求されたモジュールを PSGallery という名前の PSRepository (https://www.powershellgallery.com/) からダウンロードし、これを信頼できるソースと見なします。この PSRepository の詳細を参照するには、`Get-PSRepository -Name PSGallery` コマンドを使用します。

デフォルトでは、前のコマンドはモジュールを `%USERPROFILE%\Documents\WindowsPowerShell\Modules` フォルダにインストールします。コンピュータ AWS Tools for PowerShell のすべてのユーザーに をインストールするには、管理者として開始した PowerShell セッションで次のコマンドを実行する必要があります。例えば、次のコマンドは、すべてのユーザーがアクセスできる `%ProgramFiles%\WindowsPowerShell\Modules` フォルダに IAM モジュールをインストールします。

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

他のモジュールをインストールするには、PowerShell ギャラリー にあるように、適切なモジュール名を使用して同様のコマンドを実行します。

Windows NetCore に をインストールします AWSPowerShell。

PowerShell バージョン 3 から 5 AWSPowerShell.1、または PowerShell Core 6.0 以降で Windows を実行している .NetCore on コンピュータをインストールできます。 PowerShell Core のインストール方法については、Microsoft PowerShell ウェブサイトの「 のさまざまなバージョン PowerShellのインストール」を参照してください。

2 つの方法のいずれかNetCore で をインストールできます AWSPowerShell。

- AWSPowerShell.NetCore.zip からモジュールをダウンロードし、モジュールディレクトリの 1 つに抽出します。PSModulePath 環境変数の値を表示することで、モジュールディレクトリを検出できます。
- 次の手順で説明するように、 コマンドレットを使用して PowerShell Gallery `Install-Module` から をインストールします。

Install-Module コマンドレットを使用して PowerShell ギャラリーNetCore から AWSPowerShell.. をインストールするには

PowerShell Gallery AWSPowerShellNetCore から .. をインストールするには、コンピュータが PowerShell 5.0 以降、または PowerShell 3 以降PowerShellGetを実行している必要があります。以下のコマンドを実行します。

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

管理者 PowerShell として実行している場合、前のコマンドはコンピュータ上の AWS Tools for PowerShell すべてのユーザーに をインストールします。管理者権限のない標準ユーザー PowerShell として実行している場合、同じコマンドが現在のユーザー AWS Tools for PowerShell に対してのみインストールされます。

現在のユーザーが管理者権限を持っている場合に、そのユーザーに対してのみインストールするには、次のように `-Scope CurrentUser` パラメータセットを使用してコマンドを実行します。

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

PowerShell 3.0 以降のリリースでは、通常、モジュールで初めてコマンドレットを実行したときにモジュールが PowerShell セッションにロードされますが、 AWSPowerShell.NetCore module は大きすぎてこの機能をサポートできません。代わりに、次のコマンドを実行して、

AWSPowerShell.NetCore Core モジュールを PowerShell セッションに明示的にロードする必要があります。

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

AWSPowerShell.NetCore module を PowerShell セッションに自動的にロードするには、そのコマンドを PowerShell プロファイルに追加します。 PowerShell プロファイルの編集の詳細については、PowerShell ドキュメントの「プロファイルについて」を参照してください。

Windows AWSPowerShell に をインストールする PowerShell

は、次の 2 つの方法のいずれか AWS Tools for Windows PowerShell でインストールできます。

- AWSPowerShell.zip からモジュールをダウンロードし、モジュールディレクトリの 1 つに展開します。PSModulePath 環境変数の値を表示することで、モジュールディレクトリを検出できます。
- 次の手順で説明するように、 コマンドレットを使用して PowerShell Gallery Install-Module から をインストールします。

Install-Module コマンドレットを使用して PowerShell Gallery AWSPowerShell から をインストールするには

PowerShell 5.0 以降を実行している場合、または PowerShell 3 以降PowerShellGetで をインストールしている場合は、 PowerShell Gallery AWSPowerShell から をインストールできます。次のコマンドを実行して、Microsoft の PowerShell Gallery AWSPowerShell からインストールおよび更新できます。

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

AWSPowerShell モジュールを PowerShell セッションに自動的にロードするには、前のimport-moduleコマンドレットを PowerShell プロファイルに追加します。 PowerShell プロファイルの編集の詳細については、 PowerShell ドキュメントの「プロファイルについて」を参照してください。

> **Note**
>
> Tools for Windows PowerShell は、すべての Windows ベースの Amazon マシンイメージ (AMIs。

スクリプト実行の有効化

AWS Tools for PowerShell モジュールをロードするには、 PowerShell スクリプト実行を有効にする必要があります。スクリプトの実行を有効にするには、`Set-ExecutionPolicy` のポリシーを設定するために `RemoteSigned` コマンドレットを実行します。詳細については、Microsoft Technet ウェブサイトの「About Execution Policies」を参照してください。

> **Note**
>
> この必要条件は、Windows を実行しているコンピュータのみに適用されます。`ExecutionPolicy` セキュリティ制限は、他のオペレーティングシステムには存在しません。

スクリプト実行を有効化するには

1. 実行ポリシーを設定するには管理者権限が必要です。管理者権限を持つユーザーとしてログインしていない場合は、管理者として PowerShell セッションを開きます。[スタート] ボタンをクリックし、[すべてのプログラム] を選択します。を選択し、Windows PowerShellを選択します。Windows PowerShellを右クリックし、コンテキストメニューで管理者として実行を選択します。
2. コマンドプロンプトで次のコマンドを入力します。

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

> **Note**
>
> 64 ビットシステムでは、32 ビットバージョンの PowerShell、Windows PowerShell (x86) でこれを個別に実行する必要があります。

実行ポリシーが正しく設定されていない場合、プロファイルなどのスクリプトを実行しようとすると、 に次のエラー PowerShell が表示されます。

```
File C:\Users\username\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
 cannot be loaded because the execution
 of scripts is disabled on this system. Please see "get-help about_signing" for more
 details.
```

```
At line:1 char:2
+ . <<<<  'C:\Users\username\Documents\WindowsPowerShell
\Microsoft.PowerShell_profile.ps1'
    + CategoryInfo          : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException
```

Tools for Windows PowerShell インストーラは PSModulePath を自動的に更新して、AWSPowerShellモジュールを含むディレクトリの場所を含めます。

には AWS モジュールの ディレクトリの場所PSModulePathが含まれているため、 Get-Module -ListAvailable コマンドレットにはモジュールが表示されます。

```
PS > Get-Module -ListAvailable

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

バージョニング

AWS は、 の新バージョン AWS Tools for PowerShell を定期的にリリースし、新しい AWS のサービスと機能をサポートします。インストールしたツールのバージョンを確認するには、Get-AWSPowerShellVersioncmdlet を実行します。

```
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]
```

-ListServiceVersionInfo パラメータを Get-AWSPowerShellVersion コマンドに追加して、現在のバージョンのツールでサポートされている AWS サービスのリストを表示することもできます。モジュール化 AWS.Tools.* オプションを使用すると、現在インポートしているモジュールのみが表示されます。

```
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
...
```

実行 PowerShell している のバージョンを確認するには、 と入力$PSVersionTableして $PS VersionTable 自動変数 の内容を表示します。

```
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
```

Windows AWS Tools for PowerShell での の更新

の更新バージョン AWS Tools for PowerShell がリリースされたら、ローカルで実行しているバージョンを定期的に更新する必要があります。

モジュール化された**AWS.Tools**モジュールを更新する

AWS.Tools モジュールを最新バージョンに更新するには、次のコマンドを実行します。

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

このコマンドは、現在インストールされているすべての AWS.Tools モジュールを更新し、正常に更新されると、他のインストール済みバージョンを削除します。

> **Note**
>
> Update-AWSToolsModule コマンドレットは、すべてのモジュールを PSGallery という名前の PSRepository (https://www.powershellgallery.com/) からダウンロードし、これを信頼できるソースと見なします。この PSRepository の詳細を参照するには、Get-PSRepository -Name PSGallery コマンドを使用します。

Tools for PowerShell Core を更新する

コマンドレットを実行して実行中のバージョンを確認し、それを PowerShell Gallery `Get-AWSPowerShellVersion` ウェブサイト PowerShell にある Tools for Windows のバージョンと比較します。2～3 週間ごとにチェックすることをお勧めします。新しいコマンドと AWS サービスのサポートは、そのサポートがあるバージョンに更新した後にのみ利用できます。

の新しいリリースをインストールする前に AWSPowerShellNetCore、既存のモジュールをアンインストールします。既存のパッケージをアンインストールする前に、開いている PowerShell セッションをすべて閉じます。次のコマンドを実行して、パッケージをアンインストールします。

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

パッケージがアンインストールされたら、次のコマンドを実行して、更新されたモジュールをインストールします。

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

インストール後、 コマンドを実行して`Import-Module AWSPowerShell.NetCore`、更新されたコマンドレットを PowerShell セッションにロードします。

Tools for Windows を更新する PowerShell

コマンドレットを実行して実行中のバージョンを確認し、それを PowerShell Gallery `Get-AWSPowerShellVersion` ウェブサイト PowerShell にある Tools for Windows のバージョンと比較します。2～3 週間ごとにチェックすることをお勧めします。新しいコマンドと AWS サービスのサポートは、そのサポートがあるバージョンに更新した後にのみ利用できます。

- `Install-Module` コマンドレットを使用してインストールした場合は、次のコマンドを実行します。

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

- ダウンロードした ZIP ファイルを使用してインストールした場合:

  1. Tools for PowerShell ウェブサイトから最新バージョンをダウンロードします。ダウンロードしたファイル名に含まれるパッケージのバージョン番号と、`Get-AWSPowerShellVersion` コマンドレットの実行時に取得したバージョン番号を比較します。

2. ダウンロードバージョンがインストールしたバージョンよりも高い場合は、すべての Tools for Windows PowerShell コンソールを閉じます。
3. Tools for Windows の新しいバージョンをインストールします PowerShell。

インストール後、 `Import-Module AWSPowerShell`を実行して、更新されたコマンドレットを PowerShell セッションにロードします。または、スタートメニューからカスタム AWS Tools for PowerShell コンソールを実行します。

Linux または macOS AWS Tools for PowerShell での のインストール

このトピックでは、Linux または macOS AWS Tools for PowerShell に をインストールする方法について説明します。

セットアップの概要

Linux または macOS コンピュータ AWS Tools for PowerShell に をインストールするには、次の 2 つのパッケージオプションから選択できます。

- **`AWS.Tools`** – モジュール化されたバージョンの AWS Tools for PowerShell。各 AWS サービスは、個々の小さなモジュールでサポートされ、共有サポートモジュール を持ちます`AWS.Tools.Common`。
- AWSPowerShell.NetCore – の単一のラージモジュールバージョン AWS Tools for PowerShell。すべての AWS サービスは、この単一の大きなモジュールでサポートされています。

> **Note**
>
> 単一のモジュールは大きすぎて AWS Lambda 関数で使用できない場合があることに注意してください。代わりに、上記のモジュール化されたバージョンを使用します。

Linux または macOS を実行しているコンピュータでこれらのいずれかをセットアップする方法は、このトピックの後半で詳しく説明します。

1. サポートされているシステムに PowerShell Core 6.0 以降をインストールします。
2. PowerShell Core をインストールしたら、まずシステムシェル`pwsh`で PowerShell を実行します。

3. AWS.Tools または をインストール AWSPowerShellしますNetCore。
4. 適切なImport-Moduleコマンドレットを実行して、モジュールをセッションにインポートしますPowerShell。
5. Initialize-AWSDefaultConfigurationcmdlet を実行して、 AWS 認証情報を指定します。

前提条件

を実行するには AWS Tools for PowerShell Core、コンピュータが PowerShell Core 6.0 以降を実行している必要があります。

- サポートされている Linux プラットフォームリリースのリストと PowerShell 、Linux ベースのコンピュータに の最新バージョンをインストールする方法については、Microsoft のウェブサイト PowerShell の「Linux へのインストール」を参照してください。一部の Linux ベースのオペレーティングシステム (Arch、Kali、Raspbian など) は、公式にはサポートされていませんが、さまざまなレベルのコミュニティサポートがあります。
- サポートされている macOS バージョンと macOS に の最新バージョンをインストールする方法については PowerShell macOS 、Microsoft のウェブサイト PowerShell の「macOS での のインストール」を参照してください。

Linux または MacOS での **AWS.Tools** のインストール

Core 6.0 以降を実行 PowerShellしているコンピュータ AWS Tools for PowerShell に、モジュール化されたバージョンの をインストールできます。 PowerShell Core のインストール方法については、Microsoft PowerShell ウェブサイトの「 のさまざまなバージョン PowerShellのインストール」を参照してください。

次の 3 つの方法のいずれかで AWS.Tools をインストールできます。

- AWS.Tools.Installer モジュール内のコマンドレットを使用する。このモジュールは、他のAWS.Toolsモジュールのインストールと更新を簡素化します。 AWS.Tools.Installerには が必要でPowerShellGet、更新されたバージョンを自動的にダウンロードしてインストールします。 はモジュールバージョンAWS.Tools.Installerを自動的に同期させます。1 つのモジュールの新しいバージョンをインストールまたは更新すると、 のコマンドレットによって他のすべてのAWS.Toolsモジュールが同じバージョンAWS.Tools.Installerに自動的に更新されます。

  この方法については、以下の手順で説明します。

- AWS.Tools.zip からモジュールをダウンロードし、モジュールディレクトリの 1 つにそのモジュールを展開します。$Env:PSModulePath 変数の値を出力することで、モジュールディレクトリを検出できます。
- コマンドレットを使用して PowerShell Gallery Install-Module から各サービスモジュールをインストールします。

AWS.Tools.Installer モジュールを使用して AWS.Tools Linux または macOS に をインストールするには

1. 次のコマンドを実行して、 PowerShell Core セッションを開始します。

```
$ pwsh
```

> **Note**
>
> タスクが必要とする場合を除き、昇格されたアクセス許可を持つ管理者 PowerShell として を実行しないことをお勧めします。これは、潜在的なセキュリティリスクを避けるためであり、最小限の特権の原則にも反します。

2. AWS.Tools.Installer モジュールを使用してモジュール化された AWS.Tools パッケージをインストールするには、次のコマンドを実行します。

```
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
```

リポジトリが「信頼されていない」という通知を受けた場合は、インストールするかどうかを尋ねられます。**y** を入力して、 PowerShell がモジュールをインストールできるようにします。プロンプトを回避し、リポジトリを信頼せずにモジュールをインストールするには、次のコマンドを実行します。

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

3. これで、使用するサービスごとにモジュールをインストールできます。例えば、次のコマンドは Amazon EC2 モジュールと Amazon S3 モジュールをインストールします。このコマンドは、指定したモジュールの動作に必要な依存モジュールもインストールします。たとえば、最初の AWS.Tools サービスモジュールをインストールすると、AWS.Tools.Common もインストールされます。これは、すべての AWS サービスモジュールに必要な共有モジュールです。また、古いバージョンのモジュールを削除し、他のモジュールを同じ新しいバージョンに更新します。

```
PS > Install-AWSToolsModule AWS.Tools.EC2,AWS.Tools.S3 -CleanUp
Confirm
Are you sure you want to perform this action?
  Performing the operation "Install-AWSToolsModule" on target "AWS Tools version
 4.0.0.0".
  [Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

  Installing module AWS.Tools.Common version 4.0.0.0
  Installing module AWS.Tools.EC2 version 4.0.0.0
  Installing module AWS.Tools.Glacier version 4.0.0.0
  Installing module AWS.Tools.S3 version 4.0.0.0

  Uninstalling AWS.Tools version 3.3.618.0
  Uninstalling module AWS.Tools.Glacier
  Uninstalling module AWS.Tools.S3
  Uninstalling module AWS.Tools.SimpleNotificationService
  Uninstalling module AWS.Tools.SQS
  Uninstalling module AWS.Tools.Common
```

> **Note**
>
> Install-AWSToolsModule コマンドレットは、すべての要求されたモジュールを PSGallery という名前の PSRepository (https://www.powershellgallery.com/) からダウンロードし、このリポジトリを信頼できるソースと見なします。この PSRepository の詳細を参照するには、Get-PSRepository -Name PSGallery コマンドを使用します。

前のコマンドは、システムのデフォルトディレクトリにモジュールをインストールします。実際のディレクトリは、オペレーティングシステムのディストリビューションとバージョン、およびインストール PowerShell した のバージョンによって異なります。例えば、RHEL に似たシステムに PowerShell 7 をインストールした場合、デフォルトのモジュールは `/opt/microsoft/powershell/7/Modules` (または `$PSHOME/Modules`) にあり、ユーザーモジュールは `~/.local/share/powershell/Modules` にある可能性が最も高くなります。詳細については、Microsoft PowerShell ウェブサイトの PowerShell 「Linux へのインストール」を参照してください。モジュールがインストールされている場所を確認するには、次のコマンドを実行します。

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

他のモジュールをインストールするには、PowerShell ギャラリー にあるように、適切なモジュール名を使用して同様のコマンドを実行します。

Linux または macOS NetCore に をインストールします AWSPowerShell。

の新しいリリースにアップグレードするには AWSPowerShellNetCore、「」の手順に従いますLinux または macOS AWS Tools for PowerShell での の更新。以前のバージョンの AWSPowerShell.NetCore first をアンインストールします。

AWSPowerShell. は、次の 2 つの方法のいずれかNetCore でインストールできます。

- AWSPowerShell.NetCore.zip からモジュールをダウンロードし、モジュールディレクトリの 1 つにそのモジュールを展開します。`$Env:PSModulePath` 変数の値を出力することで、モジュールディレクトリを検出できます。
- 次の手順で説明するように、 コマンドレットを使用して PowerShell Gallery `Install-Module` から をインストールします。

Install-Module コマンドレットを使用して AWSPowerShell.NetCore on Linux または macOS をインストールするには

次のコマンドを実行して、 PowerShell Core セッションを開始します。

```
$ pwsh
```

> **Note**
>
> を実行する PowerShell ことから始めて、昇格された管理者権限 PowerShell でsudo pwsh実行しないことをお勧めします。これは、潜在的なセキュリティリスクを避けるためであり、最小限の特権の原則にも反します。

PowerShell Gallery から AWSPowerShell.NetCore single-module パッケージをインストールするには、次のコマンドを実行します。

```
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
```

リポジトリが「信頼されていない」という通知を受けた場合は、インストールするかどうかを尋ねられます。**y** を入力して、 PowerShell がモジュールをインストールできるようにします。リポジトリを信頼せずにプロンプトを回避するには、次のコマンドを実行します。

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

コンピュータ AWS Tools for PowerShell のすべてのユーザーに をインストールする場合を除き、このコマンドを root として実行する必要はありません。これを行うには、 で開始した PowerShell セッションで次のコマンドを実行しますsudo pwsh。

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

スクリプトの実行

Set-ExecutionPolicy コマンドは、Windows 以外のシステムでは使用できません。を実行できます。これはGet-ExecutionPolicy、Windows 以外のシステムで実行されている Core の PowerShellデフォルトの実行ポリシー設定が であることを示していますUnrestricted。詳細については、Microsoft Technet ウェブサイトの「About Execution Policies」を参照してください。

には AWS モジュールの ディレクトリの場所PSModulePathが含まれているため、 `Get-Module -ListAvailable` コマンドレットにはインストールしたモジュールが表示されます。

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
```

を使用する AWS Tools for PowerShell Core ように PowerShell コンソールを設定する (AWSPowerShell.NetCore Only)

PowerShell Core は通常、モジュールでコマンドレットを実行するたびにモジュールを自動的にロードします。ただし、サイズが大きい AWSPowerShellNetCore ため、これは では機能しません。 AWSPowerShell.NetCore cmdlets の実行を開始するには、まず `Import-Module AWSPowerShell.NetCore` コマンドを実行する必要があります。これは、`AWS.Tools` モジュール内のコマンドレットには必要ありません。

PowerShell セッションを初期化する

をインストールした後に PowerShell Linux ベースまたは macOS ベースのシステムで起動する場合は AWS Tools for PowerShell、Initialize-AWSDefaultConfiguration を実行して、使用する AWS アクセスキーを指定する必要があります。`Initialize-AWSDefaultConfiguration` の詳細については、「AWS 認証情報の使用」を参照してください。

> **Note**
>
> の以前のリリース (3.3.96.0 以前) では AWS Tools for PowerShell、このコマンドレットの名前は でしたInitialize-AWSDefaults。

バージョニング

AWS は、 の新バージョン AWS Tools for PowerShell を定期的にリリースし、新しい AWS のサービスと機能をサポートします。インストール AWS Tools for PowerShell した のバージョンを確認するには、Get-AWSPowerShellVersioncmdlet を実行します。

```
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]
```

現在のバージョンのツールでサポートされている AWS サービスのリストを表示するには、Get-AWSPowerShellVersioncmdlet に -ListServiceVersionInfoパラメータを追加します。

実行 PowerShell している のバージョンを確認するには、 $PSVersionTableを入力して$PSVersionTable自動変数 の内容を表示します。

```
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
```

Linux または macOS AWS Tools for PowerShell での の更新

の更新バージョン AWS Tools for PowerShell がリリースされたら、ローカルで実行しているバージョンを定期的に更新する必要があります。

モジュール化された**AWS.Tools**モジュールを更新する

AWS.Tools モジュールを最新バージョンに更新するには、次のコマンドを実行します。

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

このコマンドは、現在インストールされているすべての AWS.Tools モジュールを更新し、正常に更新されたモジュールについては、以前のバージョンを削除します。

> **Note**
>
> Update-AWSToolsModule コマンドレットは、すべてのモジュールを PSGallery という名前の PSRepository (https://www.powershellgallery.com/) からダウンロードし、これを信頼できるソースと見なします。この PSRepository の詳細を参照するには、Get-PSRepository -Name PSGallery コマンドを使用します。

Tools for PowerShell Core を更新する

コマンドレットを実行して実行中のバージョンを確認し、それを PowerShell Gallery Get-AWSPowerShellVersion ウェブサイト PowerShell にある Tools for Windows のバージョンと比較します。2～3 週間ごとにチェックすることをお勧めします。新しいコマンドと AWS サービスのサポートは、そのサポートがあるバージョンに更新した後にのみ利用できます。

の新しいリリースをインストールする前に AWSPowerShellNetCore、既存のモジュールをアンインストールします。既存のパッケージをアンインストールする前に、開いている PowerShell セッションをすべて閉じます。次のコマンドを実行して、パッケージをアンインストールします。

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

パッケージがアンインストールされたら、次のコマンドを実行して、更新されたモジュールをインストールします。

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

インストール後、 コマンドを実行して`Import-Module AWSPowerShell.NetCore`、更新されたコマンドレットを PowerShell セッションにロードします。

関連情報

- AWS Tools for Windows PowerShell の開始方法
- AWS Tools for PowerShell での AWS サービスの操作

AWS Tools for PowerShell バージョン 3.3 からバージョン 4 への移行

AWS Tools for PowerShell バージョン 4 は、AWS Tools for PowerShell バージョン 3.3 への下位互換性のあるアップデートです。既存のコマンドレットの動作を維持しながら、大幅な機能強化を追加します。

既存のスクリプトは、新しいバージョンにアップグレードした後も引き続き動作しますが、本番環境をアップグレードする前に十分にテストすることをお勧めします。

このセクションでは、変更点を示し、これらがスクリプトに与える影響について説明します。

完全モジュール化された新しい **AWS.Tools** バージョン

AWSPowerShell.NetCore and AWSPowerShell パッケージは「モノリシック」でした。つまり、AWS のすべてのサービスが同一のモジュールでサポートされるため、モジュールサイズが非常に大きく、AWS の新しいサービスや機能が追加されるたびにさらに肥大化します。新しい `AWS.Tools` パッケージは複数の小さいモジュールに分割されているため、使用する AWS のサービスに必要なモジュールだけをダウンロードしてインストールできます。パッケージには、他のすべてのモジュールと共有される必須の `AWS.Tools.Common` モジュールと、必要に応じてモジュールのインストール、更新、および削除を簡素化するための `AWS.Tools.Installer` モジュールが含まれています。

これにより、最初に Import-module を呼び出すことなく、最初の呼び出しでコマンドレットを自動的にインポートできます。ただし、コマンドレットを呼び出す前に、関連付けられた .NET オブジェクトを操作するには、 Import-Moduleを呼び出して、関連する .NET タイプについて PowerShell に通知する必要があります。

たとえば、次のコマンドには Amazon.EC2.Model.Filter への参照があります。このタイプの参照は自動インポートをトリガーできないため、Import-Module を最初に呼び出さないと、コマンドは失敗します。

```
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
  InvalidOperation: Unable to find type [Amazon.EC2.Model.Filter].
```

```
PS > Import-Module AWS.Tools.EC2
PS > $filter = [Amazon.EC2.Model.Filter]@{Name="vpc-id";Values="vpc-1234abcd"}
PS > Get-EC2Instance -Filter $filter -Select Reservations.Instances.InstanceId
  i-0123456789abcdefg
  i-0123456789hijklmn
```

新しい Get-AWSService コマンドレット

モジュールの AWS.Tools コレクションで AWS のサービス別のモジュールの名前を確認するには、Get-AWSService コマンドレットを使用します。

```
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
  ...
```

コマンドレットから返されるオブジェクトを制御するための新しい -Select パラメータ

バージョン 4 のほとんどのコマンドレットは、新しい -Select パラメータをサポートしています。各コマンドレットは、AWS SDK for .NET を使用してユーザーに代わって AWS のサービス API を呼び出します。次に、AWS Tools for PowerShellクライアントはレスポンスを PowerShell スクリプトで使用できるオブジェクトに変換し、他のコマンドにパイプします。最終 PowerShell オブジェクトには、元のレスポンスに必要な数よりも多くのフィールドまたはプロパティが含まれていることがあり、また、他の場合は、デフォルトでは含まれていないレスポンスのフィールドまたはプロパティをオブジェクトに含める必要があります。-Select パラメータを使用すると、コマンドレットから返される .NET オブジェクトに含める内容を指定できます。

例えば、Get-S3Object コマンドレットは Amazon S3 SDK オペレーション を呼び出しますListObjects。このオペレーションは ListObjectsResponse オブジェクトを返します。ただし、デフォルトでは、 コマンドレットは SDK Get-S3Object レスポンスの S3Objects要素のみを PowerShell ユーザーに返します。次の例では、そのオブジェクトは 2 つの要素を持つ配列です。

```
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
```

AWS Tools for PowerShell バージョン 4 では、SDK API コールから返された完全な .NET レスポンスオブジェクトを返すように -Select * を指定できます。

```
PS > Get-S3Object -BucketName mybucket -Select *
  IsTruncated    : False
```

```
  NextMarker     :
  S3Objects      : {file1.txt, file2.txt}
  Name           : mybucket
  Prefix         :
  MaxKeys        : 1000
  CommonPrefixes : {}
  Delimiter      :
```

特定のネストされたプロパティへのパスを指定することもできます。次の例では、S3Objects 配列内の各要素の Key プロパティのみを返します。

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

状況によっては、コマンドレットパラメータを返すと便利です。そのためには、-Select ^ParameterName を使用します。この機能は、-PassThru パラメータに代わるものです。このパラメータはまだ使用可能ですが、非推奨です。

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

各コマンドレットの参照トピックは、-Select パラメータをサポートしているかどうかを示します。

より一貫した方法による出力内の項目数の制限

以前のバージョンの AWS Tools for PowerShell では、-MaxItems パラメータを使用して、最終出力で返すオブジェクトの最大数を指定しました。

この動作は AWS.Tools から削除されています。

この動作は AWSPowerShell. NetCore および で廃止され AWSPowerShell、将来のリリースでこれらのバージョンから削除されます。

基になるサービス API が MaxItems パラメータをサポートしている場合は、依然として使用可能であり、API が指定するとおりに機能します。ただし、コマンドレットの出力で返される項目の数を制限する追加動作はなくなりました。

最終出力で返される項目の数を制限するには、出力を Select-Object コマンドレットにパイプし、-First *n* パラメータを指定します。*n* は最終出力に含める項目の最大数です。

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

AWS のすべてのサービスで -MaxItems のサポート方法が統一されていたわけではないため、これにより、不整合や予期しない結果が発生しなくなります。また、-MaxItems と新しい -Select パラメータと組み合わせて使用すると、混乱する結果が生じることがあります。

ストリームパラメータの使用の容易化

Stream タイプまたは string タイプのパラメータが、string[]、FileInfo、byte[] の各値を受け入れるようになりました。

次のいずれかの例を使用できます。

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream '{
>> "some": "json"
>> }'
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream (ls .\some.json)
```

```
PS > Invoke-LMFunction -FunctionName MyTestFunction -PayloadStream @('{', '"some":
 "json"', '}')
```

AWS Tools for PowerShell は、UTF-8 エンコーディングを使用してすべての文字列を byte[] に変換します。

プロパティ名によるパイプの拡張

ユーザーエクスペリエンスをより一貫させるために、任意のパラメータにプロパティ名を指定して、パイプライン入力を渡すことができるようになりました。

次の例では、ターゲットコマンドレットのパラメータ名と一致する名前を持つプロパティを持つカスタムオブジェクトを作成します。コマンドレットを実行すると、これらのプロパティが自動的にパラメータとして使用されます。

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

> **Note**
>
> 一部のプロパティは、以前のバージョンの AWS Tools for PowerShell でこれをサポートしていました。バージョン 4 では、これをすべてのパラメータで有効にすることで、より一貫させています。

一般的な静的パラメータ

バージョン 4.0 の AWS Tools for PowerShell で整合性を向上させるために、すべてのパラメータは静的になっています。

以前のバージョンの AWS Tools for PowerShell では、AccessKey、SecretKey、ProfileName、Region などの一般的なパラメータの一部は動的であり、その他すべてのパラメータは静的でした。は静的パラメータを動的パラメータの前に PowerShell バインドするため、これにより問題が発生する可能性があります。たとえば、次のコマンドを実行したとします。

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

の以前のバージョンでは、動的パラメータではなく-RegionName静的-Regionパラメータus-west-2に値が PowerShell バインドされています。これも、ユーザーの混乱につながる可能性があります。

AWS.Tools による必須パラメータの宣言と適用

AWS.Tools.* モジュールは、必須のコマンドレットパラメータを宣言して適用するようになりました。AWS サービスが API のパラメータが必要であることを宣言すると、 は、それを指定しなかった場合、対応する コマンドレットパラメータの入力 PowerShell を求めます。これは AWS.Tools にのみ適用されます。下位互換性を確保するために、これは AWSPowerShell.NetCore or には適用されません AWSPowerShell。

すべてのパラメータが NULL を使用可能

値タイプパラメータ (数値と日付) に $null に割り当てることができるようになりました。この変更は、既存のスクリプトには影響しません。これにより、必須パラメータを要求するプロンプトをバイパスできます。必須パラメータは、AWS.Tools でのみ要求されます。

バージョン 4 を使用して次の例を実行すると、各必須パラメータに「値」を指定するため、クライアント側の検証は効果的にバイパスされます。ただし、Amazon EC2 API サービスの呼び出しは失敗します。この AWS のサービスでは、当該情報がまだ必要とされるためです。

```
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
```

以前の非推奨機能の削除

以下は、以前のリリースの AWS Tools for PowerShell で非推奨であった機能であり、バージョン 4 では削除されています。

- Stop-EC2Instance コマンドレットから -Terminate パラメータを削除しました。代わりに Remove-EC2Instance を使用します。
- Clear-AWSCredential cmdlet から -ProfileNameパラメータを削除しました。代わりに Remove-AWSCredentialProfile を使用します。
- Import-EC2Instance コマンドレットと Import-EC2Volume コマンドレットを削除しました。

AWS Tools for Windows PowerShell の開始方法

このセクションの一部のトピックでは、ツールをインストールした後で Tools for Windows PowerShell を使用する際の基本について説明します。例えば、AWS と連係動作する場合に Tools for Windows PowerShell が使用する認証情報と AWS リージョンを指定する方法などを説明します。

このセクションの他のトピックでは、ツール、環境、およびプロジェクトを設定する高度な方法について情報を提供します。

トピック

- によるツール認証の設定 AWS
- AWS リージョンを指定する
- AWS Tools for PowerShell を使用したフェデレーティッド ID の設定
- コマンドレットの検出とエイリアス
- パイプライン処理と $AWSHistory
- 認証情報とプロファイルの解決
- ユーザーとロールに関する追加情報
- レガシー認証情報の使用

によるツール認証の設定 AWS

で開発 AWS する場合、コードが でどのように認証されるかを確立する必要があります AWS のサービス。 AWS リソースへのプログラムによるアクセスを設定するには、環境と利用可能な AWS アクセスに応じて、さまざまな方法があります。

Tools for のさまざまな認証方法については PowerShell、「 SDK とツールリファレンスガイド」の「認証とアクセス」を参照してください。 AWS SDKs

このトピックでは、新しいユーザーがローカルで開発中で、雇用主から認証方法が与えられておらず、 AWS IAM Identity Center を使用して一時的な認証情報を取得することを前提としています。ご使用の環境がこれらの前提条件に当てはまらない場合、このトピックの情報の一部はお客様に該当しない場合や、既に提供されている可能性があります。

この環境を構成するにはいくつかのステップが必要で、その概要は以下のとおりです。

1. IAM Identity Center の有効化と設定

2. IAM Identity Center を使用する PowerShell ように のツールを設定します。
3. AWS アクセスポータルセッションを開始する

IAM Identity Center の有効化と設定

を使用するには AWS IAM Identity Center、まず有効にして設定する必要があります。でこれを行う方法の詳細については PowerShell、「 SDK とツールのリファレンスガイド」の「IAM Identity Center 認証」トピックのステップ 1 を参照してください。 AWS SDKs 具体的には、「IAM Identity Center 経由のアクセスを確立していません」にある必要な指示に従ってください。

IAM Identity Center を使用する PowerShell ように のツールを設定します。

> **Note**
>
> Tools for のバージョン 4.1.538 以降 PowerShell、 SSO 認証情報を設定し、 AWS アクセスポータルセッションを開始するために推奨される方法は、このトピックで説明されているように、 `Invoke-AWSSSOLogin` `Initialize-AWSSSOConfiguration`および コマンドレットを使用することです。Tools for PowerShell （ またはそれ以降) のそのバージョンにアクセスできない場合、またはそれらのコマンドレットを使用できない場合は、 を使用してこれらのタスクを実行できます AWS CLI。この方法については、「」を参照してくださいポータルログイン AWS CLI に を使用する。

次の手順では、Tools for が一時的な認証情報を取得 PowerShell するために使用する SSO 情報で共有 AWS `config`ファイルを更新します。この手順の結果として、 AWS アクセスポータルセッションも開始されます。共有`config`ファイルにすでに SSO 情報があり、Tools for を使用して アクセスポータルセッションを開始する方法だけを知りたい場合は PowerShell、このトピックの次のセクション「」を参照してくださいAWS アクセスポータルセッションを開始する。

1. まだ開いていない場合は、一般的なコマンドレットを含め、オペレーティングシステムと環境 AWS Tools for PowerShell に適した を開いて PowerShell インストールします。これを行う方法については、「AWS Tools for PowerShell のインストール」を参照してください。

   例えば、Windows PowerShell で Tools for のモジュール化されたバージョンをインストールする場合、次のようなコマンドを実行する可能性が高くなります。

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

```
Install-AWSToolsModule AWS.Tools.Common
```

2. 以下のコマンドを実行します。サンプルプロパティ値を IAM Identity Center 設定の値に置き換えます。これらのプロパティとその検索方法については、「 SDK とツールのリファレンスガイド」の「IAM Identity Center 認証情報プロバイダーの設定」を参照してください。 AWS SDKs

```
$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
```

または、 コマンドレットを単独で使用して、、Initialize-AWSSSOConfigurationTools for でプロパティ値 PowerShell の入力を求めることもできます。

特定のプロパティ値に関する考慮事項：

- IAM Identity Center を有効にして設定する手順に従った場合、 の値は である-RoleName可能性がありますPowerUserAccess。ただし、IAM Identity Center アクセス許可セットを PowerShell 作業用に作成した場合は、代わりにそれを使用します。
- IAM Identity Center を設定した を必ず使用 AWS リージョン してください。

3. この時点で、共有 AWS configファイルには、Tools for から参照できる設定値のセットmy-sso-profileを含む というプロファイルが含まれています PowerShell。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「共有ファイルの場所」を参照してください。

Tools for PowerShell は、リクエストを に送信する前に、プロファイルの SSO トークンプロバイダーを使用して認証情報を取得します AWS。IAM Identity Center アクセス許可セットに接続された IAM ロールである sso_role_name値は、アプリケーションで AWS のサービス 使用されている へのアクセスを許可する必要があります。

次のサンプルは、上記の コマンドを使用して作成されたプロファイルを示しています。一部のプロパティ値とその順序は、実際のプロファイルで異なる場合があります。プロファイルの

`sso-session`プロパティは`my-sso-session`、 AWS アクセスポータルセッションを開始するための設定を含む という名前のセクションを参照します。

```
[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. 既にアクティブな AWS アクセスポータルセッションがある場合は、Tools for から既にログインしていることが PowerShell 通知されます。

   そうでない場合、Tools for はデフォルトのウェブブラウザで SSO 認証ページを自動的に開 PowerShell こうとします。ブラウザのプロンプトに従います。これには、SSO 認証コード、ユーザー名とパスワード、 AWS IAM Identity Center アカウントとアクセス権限セットへのアクセス権限が含まれる場合があります。

   Tools for は、 SSO ログインが成功したこと PowerShell を通知します。

AWS アクセスポータルセッションを開始する

にアクセスするコマンドを実行する前に AWS のサービス、Tools for が IAM Identity Center 認証を使用して認証情報を解決できるように PowerShell 、アクティブな AWS アクセスポータルセッションが必要です。 AWS アクセスポータルにサインインするには、 で次のコマンドを実行します。ここで PowerShell、 `-ProfileName my-sso-profile`は、このトピックの前のセクションの手順に従ったときに共有`config`ファイルで作成されたプロファイルの名前です。

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

既にアクティブな AWS アクセスポータルセッションがある場合は、Tools for から既にログインしていることが PowerShell 通知されます。

そうでない場合、Tools for はデフォルトのウェブブラウザで SSO 認証ページを自動的に開 PowerShell こうとします。ブラウザのプロンプトに従います。これには、SSO 認証コード、ユー

ザー名とパスワード、 AWS IAM Identity Center アカウントとアクセス権限セットへのアクセス権限が含まれる場合があります。

Tools for は、 SSO ログインが成功したこと PowerShell を通知します。

既にアクティブなセッションがあるかどうかをテストするには、必要に応じてAWS.Tools.SecurityTokenモジュールをインストールまたはインポートした後に次のコマンドを実行します。

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

Get-STSCallerIdentity コマンドレットへの応答は、共有 config ファイルに設定されている IAM Identity Center アカウントとアクセス許可セットを報告します。

例

以下は、Tools for で IAM Identity Center を使用する方法の例です PowerShell。以下を想定しています。

- IAM Identity Center を有効にし、このトピックで前述したように構成している。SSO プロパティは、このトピックの前半で設定した my-sso-profileプロファイルにあります。
- Initialize-AWSSSOConfiguration または コマンドレットを通じてログインすると、ユーザーには Amazon Invoke-AWSSSOLogin Amazon S3に対する少なくとも読み取り専用のアクセス許可が付与されます。
- そのユーザーは一部の S3 バケットを閲覧できる。

必要に応じてAWS.Tools.S3モジュールをインストールまたはインポートし、次の PowerShell コマンドを使用して S3 バケットのリストを表示します。

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

追加情報

- プロファイルや環境変数の使用など PowerShell、Tools for の認証に関するその他のオプションについては、「 SDK とツールのリファレンスガイド」の「設定」の章を参照してください。 AWS SDKs
- 一部のコマンドでは、 AWS リージョンを指定する必要があります。そのためには、 コマンドレットオプション、 -Region [default]プロファイル、AWS_REGION環境変数など、さまざ

まな方法があります。詳細については、このガイドAWS リージョンを指定するの「」と AWS SDKsAWS 「リージョン」を参照してください。

- ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」を参照してください。
- 短期 AWS 認証情報を作成するには、「IAM ユーザーガイド」の「一時的なセキュリティ認証情報」を参照してください。
- その他の認証情報プロバイダーについては、AWS SDK とツールのリファレンスガイドの「標準化された認証情報プロバイダー」を参照してください。

トピック

- ポータルログイン AWS CLI に を使用する

ポータルログイン AWS CLI に を使用する

Tools for のバージョン 4.1.538 以降 PowerShell、 SSO 認証情報を設定し、 AWS アクセスポータルセッションを開始するための推奨方法は、「」で説明されているように、 `Invoke-AWSSSOLogin` `Initialize-AWSSSOConfiguration`および コマンドレットを使用することですによるツール認証の設定 AWS。Tools for PowerShell （ またはそれ以降) のそのバージョンにアクセスできない場合、またはこれらのコマンドレットを使用できない場合は、 を使用してこれらのタスクを実行できます AWS CLI。

を介して IAM Identity Center を使用する PowerShell ように のツールを設定します AWS CLI。

まだ有効にしていない場合は、続行する前に IAM Identity Center を有効にして設定してください。

を使用して IAM Identity Center を使用する PowerShell ように のツールを設定する方法については AWS CLI 、「 SDK とツールリファレンスガイド」の「IAM Identity Center 認証」トピックのステップ 2 を参照してください。 AWS SDKs この設定を完了すると、システムには以下の要素が含まれるようになります。

- アプリケーションを実行する前に AWS アクセスポータルセッションを開始 AWS CLIするために使用する 。
- Tools for から参照できる設定値のセットを含む`[default]`プロファイルを含む共有 AWS `config`ファイル PowerShell。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「共有ファイルの場所」を参照してください。Tools for PowerShell は、リ

クエストを に送信する前に、プロファイルの SSO トークンプロバイダーを使用して認証情報を取得します AWS。IAM Identity Center アクセス許可セットに接続された IAM ロールである `sso_role_name`値は、アプリケーションで AWS のサービス 使用されている へのアクセスを許可する必要があります。

次のサンプル`config`ファイルは、 SSO トークンプロバイダーで設定された`[default]`プロファイルを示しています。プロファイルの `sso_session` 設定は、指定された `sso-session` セクションを参照します。`sso-session` セクションには、 AWS アクセスポータルセッションを開始するための設定が含まれています。

```
[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

SSO 解決が機能するには、 PowerShell セッションに次のモジュールをインストールしてインポートする必要があります。

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

古いバージョンの Tools for を使用していて PowerShell 、これらのモジュールがない場合は、「Assembly AWSSDK.SSOOIDC could not found...」というエラーが表示されます。

AWS アクセスポータルセッションを開始する

にアクセスするコマンドを実行する前に AWS のサービス、Tools for Windows が IAM Identity Center 認証を使用して認証情報を解決できるように PowerShell 、アクティブな AWS アクセスポー

タルセッションが必要です。設定したセッションの長さによっては、アクセスが最終的に期限切れになり、Tools for Windows で認証エラー PowerShell が発生します。 AWS アクセスポータルにサインインするには、 で次のコマンドを実行します AWS CLI。

```
aws sso login
```

[default] プロファイルを使用しているため、 --profileオプションを指定して コマンドを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルが使用されている場合、コマンドは aws sso login --profile *named-profile* になります。名前付きプロファイルの詳細については、「 SDK およびツールリファレンスガイド」の「プロファイルAWS SDKs」セクションを参照してください。

既にアクティブなセッションがあるかどうかをテストするには、次の AWS CLI コマンドを実行します (名前付きプロファイルについても同じ考慮事項があります) 。

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

このコマンドへの応答により、共有 config ファイルに設定されている IAM Identity Center アカウントとアクセス許可のセットが報告されます。

> **Note**
>
> 既にアクティブな AWS アクセスポータルセッションがあり、 を実行している場合はaws sso login、 認証情報を提供する必要はありません。
> サインインプロセスでは、データ AWS CLI へのアクセスを許可するように求められる場合があります。 AWS CLI は SDK for Python 上に構築されているため、アクセス許可メッセージにはbotocore名前のバリエーションが含まれている場合があります。

例

以下は、Tools for で IAM Identity Center を使用する方法の例です PowerShell。以下を想定しています。

- IAM Identity Center を有効にし、このトピックで前述したように構成している。SSO プロパティが [default] プロファイルにある。
- AWS CLI を使用して からログインするとaws sso login、そのユーザーには Amazon S3 に対する少なくとも読み取り専用のアクセス許可が付与されます。

- そのユーザーは一部の S3 バケットを閲覧できる。

S3 バケットのリストを表示するには、次の PowerShell コマンドを使用します。

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

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

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

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

上記のように、 [default]プロファイルを使用しているため、 Get-S3Bucket -ProfileNameオプションを使用して コマンドレットを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルを使用している場合、コマンドは Get-S3Bucket -ProfileName *named-profile* です。名前付きプロファイルの詳細については、「 SDK およびツールリファレンスガイド」の「プロファイルAWS SDKs」セクションを参照してください。

追加情報

- プロファイルや環境変数の使用など PowerShell、Tools for の認証に関するその他のオプションについては、「 SDK とツールのリファレンスガイド」の「設定」の章を参照してください。 AWS SDKs
- 一部のコマンドでは、 AWS リージョンを指定する必要があります。そのためには、 コマンドレットオプション、 -Region [default]プロファイル、AWS_REGION環境変数など、さまざまな方法があります。詳細については、このガイドAWS リージョンを指定するの「」と AWS SDKsAWS 「リージョン」を参照してください。
- ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」を参照してください。

- 短期 AWS 認証情報を作成するには、「IAM ユーザーガイド」の「一時的なセキュリティ認証情報」を参照してください。
- その他の認証情報プロバイダーについては、AWS SDK とツールのリファレンスガイドの「標準化された認証情報プロバイダー」を参照してください。

AWS リージョンを指定する

AWS Tools for PowerShell コマンドを実行するときに使用する AWS リージョンを指定するには、2 つの方法があります。

- 個々のコマンドで `-Region` 共通パラメータを使用します。
- `Set-DefaultAWSRegion` コマンドを使用して、すべてのコマンドのデフォルトのリージョンを設定します。

Tools for Windows が使用するリージョンを特定できない場合、多くの AWS コマンドレット PowerShell は失敗します。例外には、Amazon S3、Amazon SES、および のコマンドレットがあり AWS Identity and Access Management、自動的にグローバルエンドポイントに設定されます。

1 つの AWS コマンドにリージョンを指定するには

次のような `-Region` パラメータをコマンドに追加します。

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

現在のセッションのすべての AWS CLI コマンドにデフォルトのリージョンを設定するには

PowerShell コマンドプロンプトから、次のコマンドを入力します。

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

> **Note**
>
> この設定は、現在のセッション中にのみ維持されます。設定をすべての PowerShellセッションに適用するには、コマンドの場合と同様に、このImport-Moduleコマンドを PowerShell プロファイルに追加します。

すべての AWS CLI コマンドの現在のデフォルトリージョンを表示するには

PowerShell コマンドプロンプトから、次のコマンドを入力します。

```
PS > Get-DefaultAWSRegion

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

すべての AWS CLI コマンドの現在のデフォルトリージョンをクリアするには

PowerShell コマンドプロンプトから、次のコマンドを入力します。

```
PS > Clear-DefaultAWSRegion
```

利用可能なすべての AWS リージョンのリストを表示するには

PowerShell コマンドプロンプトから、次のコマンドを入力します。サンプルの 3 番目の列で、現在のセッションのデフォルトのリージョンがわかります。

```
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**
>
> 一部のリージョンはサポートされていても、Get-AWSRegion コマンドレットの出力に含まれていない場合があります。例えば、これはまだグローバルではないリージョンにも当てはまる場合があります。-Region パラメータをコマンドに追加してもリージョンを指定できない場合は、次のセクションに示すように、代わりにカスタムエンドポイントでリージョンを指定してみてください。

カスタムエンドポイントまたは非標準エンドポイントの指定

次のサンプル形式で Tools for Windows PowerShell コマンドに`-EndpointUrl`共通パラメータを追加して、カスタムエンドポイントを URL として指定します。

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

以下は `Get-EC2Instance` コマンドレットを使用した例です。この例では、カスタムエンドポイントは `us-west-2` または米国西部 (オレゴン) にありますが、他のサポートされている任意の AWS リージョン (`Get-AWSRegion` で列挙されないリージョンも含む) を使用できます。

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

追加情報

AWS リージョンの詳細については、「 SDK およびツールリファレンスガイド」のAWS 「リージョン」を参照してください。 AWS SDKs

AWS Tools for PowerShell を使用したフェデレーティッド ID の設定

組織のユーザーが AWS リソースにアクセスできるようにするには、セキュリティ、監査性、コンプライアンス、ロールとアカウントの分離のサポートという目的で、標準の繰り返し可能な認証方法を設定する必要があります。ユーザーに AWS API へのアクセス機能を提供するのが一般的ですが、フェデレーティッド API アクセスがない場合、AWS Identity and Access Management (IAM) ユーザーも作成する必要があり、フェデレーションを使用する意味がなくなります。このトピックでは、AWS Tools for PowerShell での SAML (Security Assertion Markup Language) のサポートについて説明します。このサポートにより、フェデレーティッドアクセスソリューションが簡単になります。

AWS Tools for PowerShell の SAML サポートを使用すると、ユーザーに AWS サービスへのフェデレーティッドアクセスを提供できます。SAML は、サービス間、特に ID プロバイダ (Active Directory フェデレーションサービスなど) とサービスプロバイダ (AWS など) の間でユーザーの認証と許可のデータを転送するための XML ベースのオープンスタンダード形式です。SAML とそのしくみの詳細については、Wikipedia の SAML、または Organization for the Advancement of Structured

Information Standards (OASIS) ウェブサイトの SAML 技術仕様を参照してください。AWS Tools for PowerShell での SAML サポートは SAML 2.0 と互換性があります。

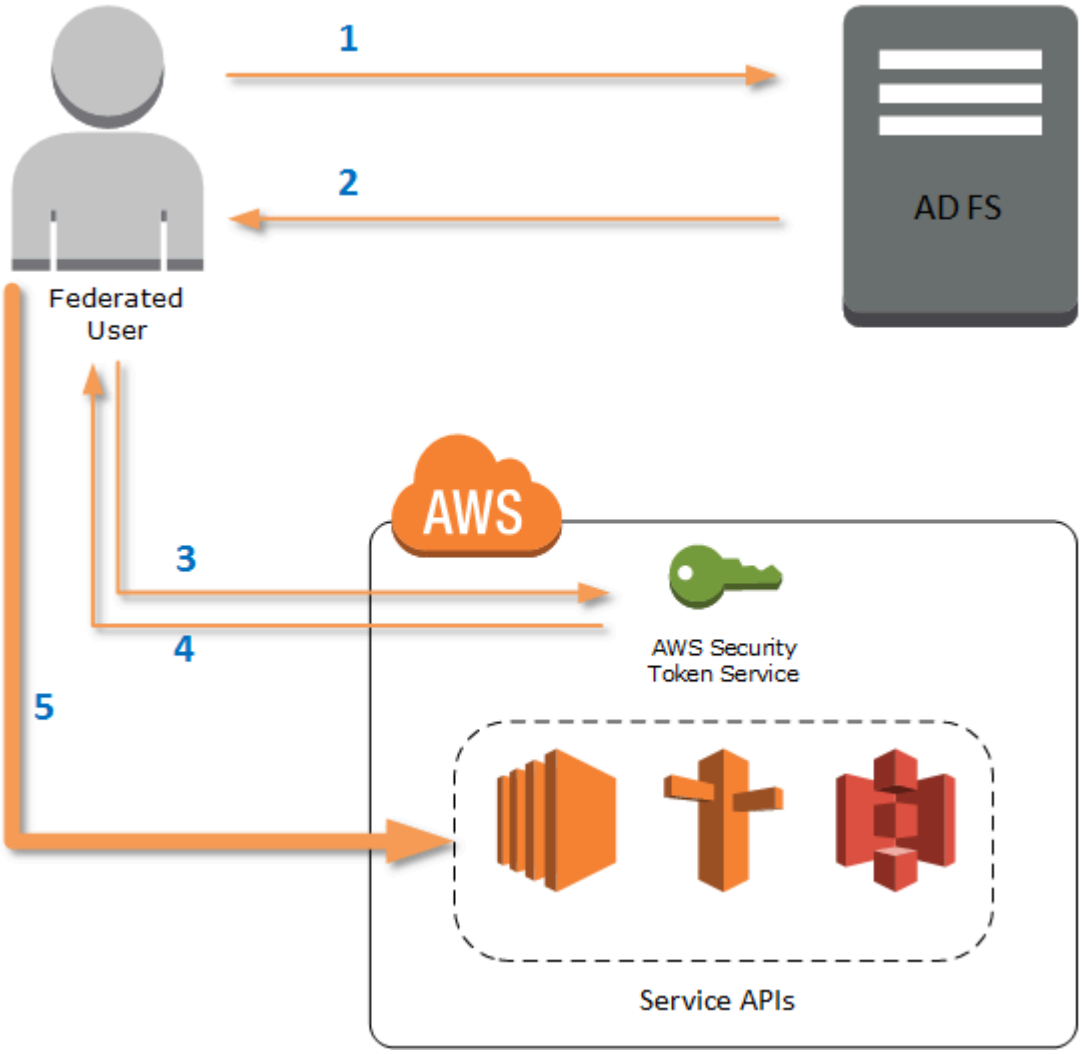
前提条件

初めて SAML サポートを使用する前に、次の項目を用意しておく必要があります。

- 組織の認証情報のみを使用して、コンソールにアクセスするために AWS アカウントと正常に統合されたフェデレーティッド ID ソリューション。Active Directory フェデレーションサービスに対してこれを行う方法については、IAM ユーザーガイドの「SAML 2.0 フェデレーションについて」、およびブログの投稿「Enabling Federation to AWS Using Windows Active Directory, AD FS, and SAML 2.0」を参照してください。このブログの投稿では AD FS 2.0 について説明していますが、AD FS 3.0 を実行している場合でも手順は似ています。
- ローカルワークステーションにインストールされた AWS Tools for PowerShell のバージョン 3.1.31.0 以降。

ID フェデレーティッドユーザーが AWS のサービス API にフェデレーティッドアクセスするしくみ

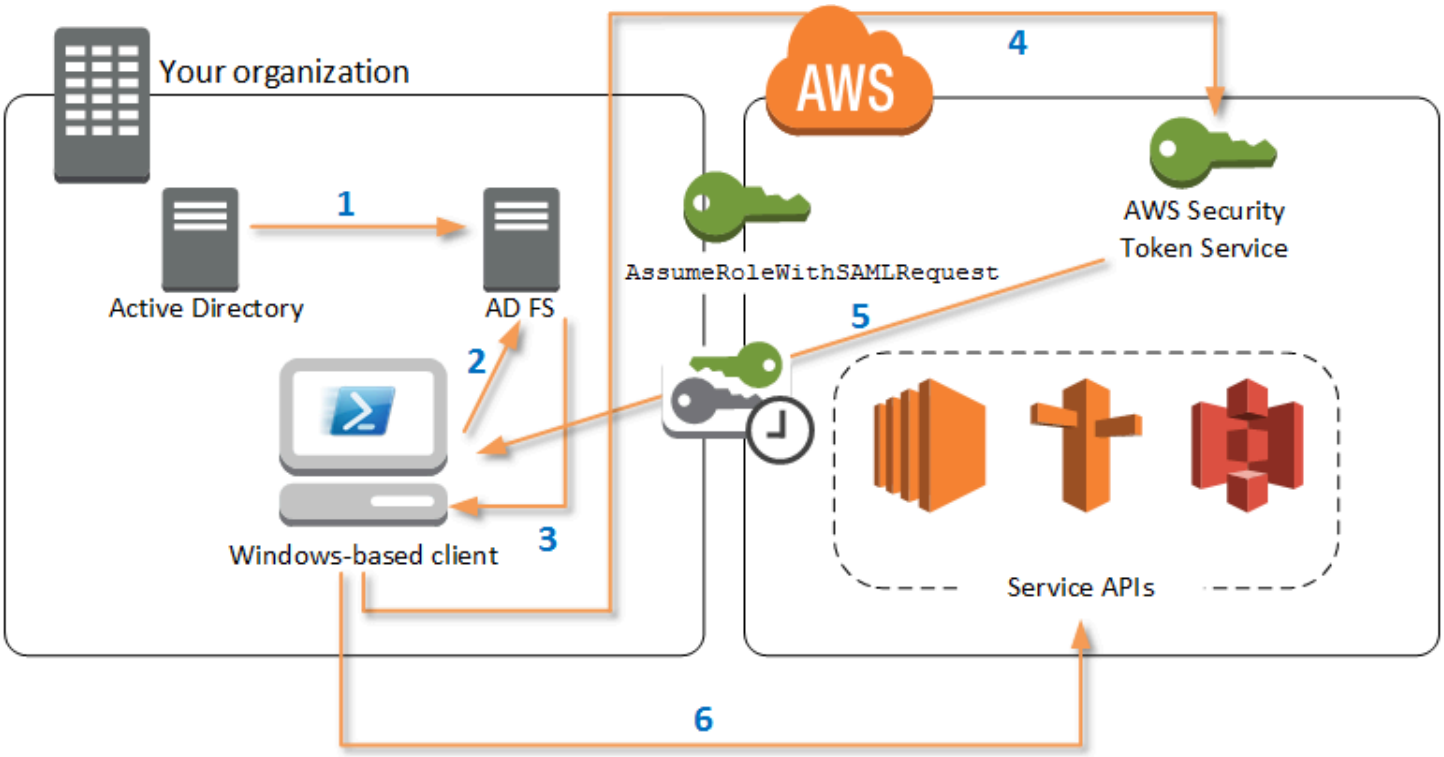
次のプロセスは、AWS リソースにアクセスするために Active Directory (AD) ユーザーが AD FS によってフェデレーションされるプロセスの概要を示しています。



1. フェデレーティッドユーザーのコンピュータ上のクライアントは、AD FS に対して認証されます。
2. 認証が成功すると、AD FS はユーザーに SAML アサーションを送信します。
3. ユーザーのクライアントは、SAML フェデレーション要求の一部として SAML アサーションを AWS Security Token Service (STS) に送信します。
4. STS は、ユーザーが継承できるロールの一時的な AWS 認証情報を含む SAML レスポンスを返します。
5. ユーザーは、AWS Tools for PowerShell によるリクエストに一時的な認証情報を含めることによって、AWS サービス API にアクセスします。

AWS Tools for PowerShell での SAML サポートのしくみ

このセクションでは、AWS Tools for PowerShell コマンドレットを使用してユーザーの SAML ベースの ID フェデレーションの設定を有効にする方法を説明します。



1. AWS Tools for PowerShell は、Windows ユーザーの現在の認証情報を使用して、または、AWS に呼び出すために認証情報を必要とするコマンドレットが実行されたときはインタラクティブに、AD FS に対して認証を行います。
2. AD FS がユーザーを認証します。
3. AD FS は、アサーションを含む SAML 2.0 認証応答を生成します。アサーションの目的は、ユーザーに関する情報を識別して提供することです。AWS Tools for PowerShell は、SAML アサーションからユーザーの許可されたロールのリストを抽出します。
4. AWS Tools for PowerShell は、`AssumeRoleWithSAMLRequest` API コールを行って、リクエストされたロールの Amazon リソースネーム (ARN) などの SAML リクエストを STS に転送します。
5. SAML リクエストが有効な場合、STS は、AWS `AccessKeyId`、`SecretAccessKey`、`SessionToken` を含んだレスポンスを返します。これらの認証情報は 3,600 秒 (1 時間) 有効です。

6. これにより、ユーザーに、ユーザーのロールでアクセスを許可されている AWS サービス API を操作するための有効な認証情報が付与されます。AWS Tools for PowerShell は、これらの認証情報を後続の AWS API 呼び出しに自動的に適用し、有効期限が切れると自動的に更新します。

> **Note**
>
> 認証情報の有効期限が切れ、新しい認証情報が必要になると、AWS Tools for PowerShell は自動的に AD FS と再認証を行い、次の 1 時間用の新しい認証情報を取得します。ドメイン結合されたアカウントのユーザーの場合、このプロセスはサイレントに行われます。ドメイン結合されていないアカウントの場合、ユーザーは、再認証のために認証情報を入力するように AWS Tools for PowerShell によって指示されます。

PowerShell SAML 設定コマンドレットを使用する方法

AWS Tools for PowerShell には、SAML サポートを提供する新しい 2 つのコマンドレットが含まれます。

- `Set-AWSSamlEndpoint` は、AD FS エンドポイントを設定し、エンドポイントにわかりやすい名前を割り当て、必要に応じて、エンドポイントの認証タイプを説明します。
- `Set-AWSSamlRoleProfile` は、AD FS エンドポイントと関連付ける必要があるユーザーアカウントのプロファイルを作成または編集します。このエンドポイントは、`Set-AWSSamlEndpoint` コマンドレットにわかりやすい名前を指定することで識別されます。各ロールプロファイルは、ユーザーが実行を許可されている 1 つのロールにマップします。

  AWS 認証情報プロファイルと同様、ロールプロファイルにわかりやすい名前を割り当てます。`Set-AWSCredential` コマンドレットと同じわかりやすい名前を使用するか、AWS のサービス API を呼び出すコマンドレットの `-ProfileName` パラメータの値として使用できます。

新しい AWS Tools for PowerShell セッションを開きます。PowerShell 3.0 以降を実行している場合、そのコマンドレットのいずれかを実行すると、AWS Tools for PowerShell モジュールが自動的にインポートされます。PowerShell 2.0 を実行している場合は、次の例に示すように、「Import-Module」コマンドレットを実行して、モジュールを手動でインポートする必要があります。

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

Set-AWSSamlEndpoint と Set-AWSSamlRoleProfile コマンドレットを実行する方法

1. まず、AD FS システムのエンドポイント設定を行います。最も簡単な方法は、次の手順に示すように、変数にエンドポイントを保存する方法です。必ず、プレースホルダーアカウント ID と AD FS ホスト名を自分のアカウント ID と AD FS ホスト名に置き換えます。Endpoint パラメータに AD FS ホスト名を指定します。

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

2. エンドポイント設定を作成するには、Set-AWSSamlEndpoint パラメータに正しい値を指定して、AuthenticationType コマンドレットを実行します。有効な値には、Basic、Digest、Kerberos、Negotiate、および NTLM があります。このパラメータを指定しない場合、デフォルト値は Kerberos になります。

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

   コマンドレットは、-StoreAs パラメータを使用して割り当てたわかりやすい名前を返すため、次の行で Set-AWSSamlRoleProfile を実行するときにその名前を使用できます。

3. 次に、Set-AWSSamlRoleProfile コマンドレットを実行して、AD FS ID プロバイダとの認証を行い、ユーザーに実行が許可されている一連のロールを (SAML アサーションで) 取得します。

   Set-AWSSamlRoleProfile コマンドレットは、返された一連のロールを使用して、指定のプロファイルに関連付けるロールを選択するようにユーザーに指示するか、パラメータに指定されたロールのデータが存在することを確認します (存在しない場合、ユーザーに選択するように指示します)。ユーザーに許可されたロールが 1 つのみの場合、コマンドレットは、ユーザーに指示することなく、ロールをプロファイルに自動的に割り当てます。ドメイン結合用途にプロファイルを設定するために認証情報を提供する必要はありません。

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

   または、ドメイン結合されていないアカウントの場合、次の行に示すように、Active Directory 認証情報を提供し、ユーザーにアクセス権がある AWS ロールを選択できます。これは、別の Active Directory ユーザーアカウントが存在する場合に、組織内のロールを区別するときに役立ちます (たとえば、管理機能)。

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

4. どちらの場合も、Set-AWSSamlRoleProfile コマンドレットはプロファイルに保存するロールを選択するように指示します。次の例は、2 つの使用可能なロール ADFS-Dev および ADFS-Production を示しています。IAM ロールは、AD FS 管理者によって AD ログイン認証情報に関連付けられます。

```
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"):
```

   または、RoleARN、PrincipalARN、オプションの NetworkCredential パラメータを入力して、プロンプトを表示せずにロールを指定することもできます。指定されたロールが認証によって返されたアサーションにリストされていない場合、ユーザーは使用可能なロールから選択するよう求められます。

```
PS > $params = @{ "NetworkCredential"=$credential,
 "PrincipalARN"="{arn:aws:iam::012345678912:saml-provider/ADFS}",
 "RoleARN"="{arn:aws:iam::012345678912:role/ADFS-Dev}"
}
PS > $epName | Set-AWSSamlRoleProfile @params -StoreAs SAMLDemoProfile1 -Verbose
```

5. 次のコードに示すように、StoreAllRoles パラメータを追加することで、1 つのコマンドですべてのロールのプロファイルを作成できます。ロール名はプロファイル名として使用されます。

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

ロールプロファイルを使用して、AWS 認証情報を必要とするコマンドレットを実行する方法

AWS 認証情報を必要とするコマンドレットを実行するには、AWS 共有認証情報ファイルで定義されたロールプロファイルを使用できます。ロールプロファイルの名前を Set-AWSCredential に

(または、AWS Tools for PowerShell の任意の ProfileName パラメータの値として) 指定して、プロファイルに記述されているロールの一時的な AWS 認証情報を自動的に取得します。

使用するのは一度に 1 つのロールプロファイルですが、シェルセッション内でプロファイルを切り替えることができます。Set-AWSCredential コマンドレットは、単独で実行した場合、認証を行わず、認証情報を取得しません。コマンドレットは、指定したロールプロファイルの使用をユーザーが希望していることを記録します。AWS 認証情報を必要とするコマンドレットを実行するまで、認証または認証情報のリクエストは行われません。

これで、SAMLDemoProfile プロファイルを使用して取得した一時的な AWS 認証情報を使用して、AWS のサービス API を使用できるようになります。次のセクションでは、ロールプロファイルの使用方法の例を示します。

例 1: **Set-AWSCredential** でデフォルトロールを設定する

この例では、Set-AWSCredential を使用して、AWS Tools for PowerShell セッションのデフォルトロールを設定します。次に、認証情報を必要とするコマンドレットを実行できます。これにより、指定されたロールによって権限が付与されます。この例では、Set-AWSCredential コマンドレットで指定されたプロファイルに関連付けられている米国西部 (オレゴン) リージョンにあるすべての Amazon Elastic Compute Cloud インスタンスを一覧表示します。

```
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}
```

例 2: PowerShell セッション中にロールプロファイルを変更する

この例では、SAMLDemoProfile プロファイルに関連付けられているロールの AWS アカウントにある使用可能なすべての Amazon S3 バケットを一覧表示します。この例では、AWS Tools for PowerShell セッションで別のプロファイルを使用していた場合でも、それをサポートするコマンドレットで -ProfileName パラメータに別の値を指定することで、プロファイルを変更できることを

示しています。これは、PowerShell コマンドラインから Amazon S3 を管理している管理者の一般的なタスクです。

```
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
```

Get-S3Bucket コマンドレットでは、Set-AWSSamlRoleProfile コマンドレットを実行して作成されたプロファイルの名前を指定します。このコマンドは、セッションで以前に (たとえば、Set-AWSCredential コマンドレットを実行して) ロールプロファイルを設定し、Get-S3Bucket コマンドレットに別のロールプロファイルを使用する場合に役立ちます。プロファイルマネージャーは、一時的な認証情報を Get-S3Bucket コマンドレットに使用できるようにします。

認証情報は 1 時間で有効期限が切れますが (STS によって実施される制限)、AWS Tools for PowerShell が現在の認証情報の有効期限が切れたことを検出すると、ツールは、新しい SAML アサーションをリクエストすることで、認証情報を自動的に更新します。

ドメイン結合されたユーザーの場合、現在のユーザーの Windows ID が認証時に使用されるため、このプロセスは中断なしで行われます。ドメイン結合されていないユーザーアカウントの場合、AWS Tools for PowerShell は、PowerShell 認証情報のプロンプトを表示して、ユーザーパスワードを要求します。ユーザーは、ユーザーの認証に使用する認証情報を指定し、新しいアサーションを取得します。

例 3: リージョンのインスタンスを取得する

次の例では、ADFS-Production プロファイルで使用されたアカウントに関連付けられているアジアパシフィック (シドニー) リージョンにあるすべての Amazon EC2 インスタンスを一覧表示します。これは、リージョンにあるすべての Amazon EC2 インスタンスを返す便利なコマンドです。

```
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
```

その他の参考資料

フェデレーティッド API アクセスの実装方法に関する一般的な情報については、「How to Implement a General Solution for Federated API/CLI Access Using SAML 2.0」を参照してください。

ご質問やご意見がございましたら、PowerShell Scripting または .NET Development の AWS 開発者フォーラムにアクセスしてください。

コマンドレットの検出とエイリアス

このセクションでは、AWS Tools for PowerShell でサポートされているサービスを一覧表示する方法、これらのサービスをサポートするために AWS Tools for PowerShell から提供されているコマンドレットのセットを表示する方法、これらのサービスにアクセスするための代替のコマンドレット名 (エイリアスとも呼ばれます) を見つける方法について説明します。

コマンドレットの検出

すべての AWS サービスオペレーション (または API) は、各サービスの API リファレンスガイドに記載されています。例えば、「IAM API リファレンス」を参照してください。ほとんどの場合、AWS サービス API と AWS PowerShell コマンドレット間には 1 対 1 の対応があります。AWS サービスの API 名に対応するコマンドレット名を取得するには`-ApiOperation`、パラメータと AWS サービスの API 名を指定して AWS `Get-AWSCmdletName` コマンドレットを呼び出します。例えば、利用可能なすべての `DescribeInstances` AWS サービスの API に基づくすべてのコマンドレット名を取得するには、次のコマンドを実行します。

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

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

-ApiOperation パラメータはデフォルトのパラメータなので、パラメータ名は省略できます。次の例は、前の例と同等です。

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

API とサービスの両方の名前がわかっている場合は、コマンドレット名プレフィックスまたは AWS サービス名の一部と -Service パラメータを含めることができます たとえば、Amazon EC2 のコマンドレット名プレフィックスは EC2 です。Amazon EC2 サービスの DescribeInstances API に対応するコマンドレット名を取得するには、以下のいずれかのコマンドを実行します。どのコマンドでも同じ出力になります。

```
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
```

これらのコマンドのパラメータ値では、大文字と小文字が区別されません。

使用する AWS サービス API または AWS サービスの名前がわからない場合は、マッチングパターンおよび -MatchWithRegex パラメータとともに -ApiOperation パラメータを使用できます。たとえば、SecurityGroup を含むすべてのコマンドレット名を取得するには、次のコマンドを実行します。

```
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
```

AWS サービスの名前はわかるが、AWS サービス API の名前がわからない場合は、-MatchWithRegex パラメータと -Service パラメータを追加して、検索範囲を単一のサービスに絞り込みます。たとえば、Amazon EC2 サービス内のみで、SecurityGroup を含むすべてのコマンドレット名を取得するには、次のコマンドを実行します。

```
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
```

AWS Command Line Interface (AWS CLI) コマンドの名前がわかっている場合は、`-AwsCliCommand` パラメータと希望する AWS CLI コマンド名を使用して、同じ API に基づくコマンドレット名を取得します。例えば、Amazon EC2 サービスの `authorize-security-group-ingress` AWS CLI コマンド呼び出しに対応するコマンドレット名を取得するには、次のコマンドを実行します。

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

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

`Get-AWSCmdletName` コマンドレットには、サービスと AWS API を特定できる AWS CLI コマンド名の一部を指定すれば十分です。

Tools for PowerShell Core のすべてのコマンドレットを一覧表示するには、次の例に示すように PowerShell `Get-Command` コマンドレットを実行します。

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

`-Module AWSPowerShell` で同じコマンドを実行すると、AWS Tools for Windows PowerShell でコマンドレットを確認できます。

`Get-Command` コマンドレットは、コマンドレットをアルファベット順に一覧表示します。デフォルトでは、リストは PowerShell 名詞ではなく、PowerShell 動詞によってソートされます。

代わりに、サービスによって結果をソートするには、次のコマンドを実行します。

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

Get-Command コマンドレットから返されたコマンドレットをフィルタリングするには、PowerShell の Select-String コマンドレットを実行します。たとえば、AWS リージョンを使用するコマンドレットのセットを表示するには、次のコマンドを実行します。

```
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
```

コマンドレット名詞のサービスプレフィックスでフィルタすることで特定のサービスのコマンドレットを見つけることもできます。使用可能なサービスプレフィックスのリストを表示するには、Get-AWSPowerShellVersion -ListServiceVersionInfo を実行します。次の例では、Amazon CloudWatch Events サービスをサポートするコマンドレットが返されます。

```
PS > Get-Command -Module AWSPowerShell -Noun CWE*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-CWEResourceTag                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWEEventSource                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWERule                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWEEventSource                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBus                                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBusList                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSource                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSourceList                             3.3.563.1
 AWSPowerShell.NetCore
```

```
Cmdlet          Get-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceAccountList            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceList                   3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleDetail                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleNamesByTarget                        3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWETargetsByRule                            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEEventBus                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEEventBus                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPartnerEventSource                    3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPermission                            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEResourceTag                           3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWERule                                  3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWETarget                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Test-CWEEventPattern                            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEEvent                                  3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPartnerEvent                           3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPermission                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWERule                                   3.3.563.1
 AWSPowerShell.NetCore
```

```
Cmdlet          Write-CWETarget                                    3.3.563.1
 AWSPowerShell.NetCore
```

コマンドレットの名前付けとエイリアス

各サービスの AWS Tools for PowerShell のコマンドレットは、そのサービスの AWS SDK で提供されているメソッドに基づいています。ただし、PowerShell の必須の命名規則のため、コマンドレットの名前は、それが基づく API コールやメソッドの名前とは異なる場合があります。たとえば、`Get-EC2Instance` コマンドレットは Amazon EC2 `DescribeInstances` メソッドに基づいています。

コマンドレット名とメソッド名が同じ場合でも、実際に実行される機能は異なることがあります。たとえば、Amazon S3 `GetObject` メソッドは Amazon S3 オブジェクトを取得します。一方、`Get-S3Object` コマンドレットは、オブジェクト自体ではなく、Amazon S3 オブジェクトに関する情報を返します。

```
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
```

AWS Tools for PowerShell を使用して S3 オブジェクトを取得するには、`Read-S3Object` コマンドレットを実行します。

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

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

> **Note**
>
> AWS コマンドレットのコマンドレットヘルプには、コマンドレットが基づく AWS SDK API の名前が記載されています。

標準の PowerShell 動詞とその意味の詳細については、「Approved Verbs for PowerShell Commands」を参照してください。

Remove 動詞を使用するすべての AWS コマンドレット (および -Terminate パラメータの追加時の Stop-EC2Instance コマンドレット) では、処理を続行する前に確認プロンプトが表示されます。確認を省略するには、コマンドに -Force パラメータを追加します。

> **Important**
>
> AWS コマンドレットでは、-WhatIf スイッチはサポートされていません。

エイリアス

AWS Tools for PowerShell のセットアップでは、多くの AWS コマンドレットのエイリアスが格納されたエイリアスファイルがインストールされます。これらのエイリアスは、コマンドレット名よりも直感的で理解しやすい場合があります。たとえば、一部のエイリアスでは、サービス名と AWS SDK メソッド名によって PowerShell の動詞と名詞が置き換えられます。例として EC2-DescribeInstances エイリアスがあります。

また、標準の PowerShell 命名規則には従わないものの、実際のオペレーションよりもわかりやすい動詞を使用するエイリアスもあります。たとえば、エイリアスファイルでは、エイリアス Get-S3Content がコマンドレット Read-S3Object に対応付けられています。

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

エイリアスファイルは、AWS Tools for PowerShell インストールディレクトリにあります。使用環境にエイリアスをロードするには、ドットソース形式でファイルを読み込みます。以下は、Windows ベースの例です。

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

Linux または macOS シェルの場合、次のようになります。

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

すべての AWS Tools for PowerShell エイリアスを表示するには、次のコマンドを実行します。このコマンドは、PowerShell `Where-Object` コマンドレットの ? エイリアスと `Source` プロパティを使用して、`AWSPowerShell.NetCore` モジュールからのエイリアスのみをフィルタリングします。

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

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           Add-ASInstances                                    3.3.343.0
 AWSPowerShell
Alias           Add-CTTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-DPTags                                         3.3.343.0
 AWSPowerShell
Alias           Add-DSIpRoutes                                     3.3.343.0
 AWSPowerShell
Alias           Add-ELBTags                                        3.3.343.0
 AWSPowerShell
Alias           Add-EMRTag                                         3.3.343.0
 AWSPowerShell
Alias           Add-ESTag                                          3.3.343.0
 AWSPowerShell
Alias           Add-MLTag                                          3.3.343.0
 AWSPowerShell
Alias           Clear-AWSCredentials                               3.3.343.0
 AWSPowerShell
Alias           Clear-AWSDefaults                                  3.3.343.0
 AWSPowerShell
Alias           Dismount-ASInstances                               3.3.343.0
 AWSPowerShell
Alias           Edit-EC2Hosts                                      3.3.343.0
 AWSPowerShell
Alias           Edit-RSClusterIamRoles                             3.3.343.0
 AWSPowerShell
Alias           Enable-ORGAllFeatures                              3.3.343.0
 AWSPowerShell
Alias           Find-CTEvents                                      3.3.343.0
 AWSPowerShell
Alias           Get-ASACases                                       3.3.343.0
 AWSPowerShell
Alias           Get-ASAccountLimits                                3.3.343.0
 AWSPowerShell
```

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

...
```

このファイルに独自のエイリアスを追加するには、必要に応じて PowerShell の `$MaximumAliasCount` ユーザー設定変数の値を 5500 より大きい値に引き上げます。デフォルト値

は 4096 であり、これを最大 32768 まで引き上げることができます。これを行うには、次のコマンドを実行します。

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

変更が正常に行われたことを確認するには、変数名を入力して現在の値を表示します。

```
PS > $MaximumAliasCount
32768
```

パイプライン処理と $AWSHistory

コレクションを返す AWS サービス呼び出しで、コレクション内のオブジェクトはパイプラインに列挙されます。コレクションに含まれておらず、ページング制御フィールドではない追加フィールドを含む結果オブジェクトでは、これらのフィールドが呼出しの Note プロパティとして追加されます。これらの Note プロパティは、新しい `$AWSHistory` セッション変数に記録されるため、必要に応じて参照できます。`$AWSHistory` 変数については、次のセクションで説明します。

> **Note**
>
> v1.1 より前のバージョンの Tools for Windows PowerShell では、コレクションオブジェクト自体を出力していました。そのため、パイプライン処理を続行するには、foreach {$_.getenumerator()} を使用する必要がありました。

例

次の例では、各リージョンの AWS リージョンと Amazon EC2 マシンイメージ (AMI) のリストが返されます。

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

次の例では、現在のデフォルトのリージョン内のすべての Amazon EC2 インスタンスが停止されます。

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

コレクションはパイプラインに列挙されるため、指定されたコマンドレットの出力は、$null、単一のオブジェクト、コレクションのいずれかである可能性があります。出力がコレクションの場合は、.Count プロパティを使用してコレクションのサイズを決定できます。ただし、単一のオブジェクトのみが送信される場合、.Count プロパティは存在しません。スクリプトが、送信されたオブジェクトの数を一貫した方法で判断する必要がある場合は、$AWSHistory で最後のコマンド値の EmittedObjectsCount プロパティを確認します。

$AWSHistory

パイプライン処理のサポートを改善するために、送信されたコレクションオブジェクトにサービスの応答および結果インスタンスを Note プロパティとして含めるための、AWS コマンドレットからの出力はリシェイプされません。代わりに、出力として単一のコレクションを送信する呼び出しについては、コレクションが PowerShell パイプラインに列挙されるようになりました。つまり、AWS SDK の応答および結果データをアタッチできるコレクションオブジェクトを格納できないため、これらのデータはパイプ内に存在できなくなります。

ほとんどのユーザーはこれらのデータを必要としませんが、コマンドレットによって呼び出された下層の AWS サービスとの間で送受信されたデータを詳細に確認できるため、診断目的には有用です。

バージョン 1.1 より、これらのデータを $AWSHistory という名前の新しいシェル変数として参照できるようになりました。この変数には、AWS コマンドレット呼び出しと、呼び出しごとに受信したサービス応答の記録が格納されます。必要に応じて、この履歴を、各コマンドレットによって行われたサービスリクエストを記録するように設定することもできます。コマンドレット全体の実行時間といった、その他の有益なデータを各エントリから取得することもできます。セキュリティ上の理由から、機密データを含むリクエストとレスポンスはデフォルトでは記録されません。ただし、必要に応じてこの動作をオーバーライドするように履歴を設定できます。詳細については、以下の Set-AWSHistoryConfiguration コマンドレットを参照してください。

$AWSHistory.Commands リスト内の各エントリの型は AWSCmdletHistory です。この型には、以下の有用なメンバーが含まれています。

CmdletName

コマンドレットの名前。

CmdletStart

コマンドレットが実行された日時。

CmdletEnd

コマンドレットがすべての処理を終了した日時。

リクエスト

リクエストの記録が有効になっている場合の最後のサービスリクエストのリスト。

レスポンス

受信した最後のサービス応答のリスト。

LastServiceResponse

最新のサービス応答を返すためのヘルパー。

LastServiceRequest

最新のサービスリクエストを返すためのヘルパー (使用可能な場合)。

サービス呼び出しを実行する AWS コマンドレットが使用されるまで、$AWSHistory 変数は作成されません。それまでは $null に評価されます。

> **Note**
>
> 旧バージョンの Tools for Windows PowerShell では、サービス応答に関連するデータを、返されたオブジェクトの Note プロパティとして出力していました。これらのデータは、リスト内の呼び出しごとに記録される応答エントリとして残されるようになりました。

Set-AWSHistoryConfiguration

コマンドレットの呼び出しは、ゼロ個以上のサービスリクエストとサービス応答のエントリを保持できます。メモリの消費量を抑えるため、$AWSHistory リストには、デフォルトで、最新の 5 つのコマンドレットの実行記録のみが保持され、それぞれについて、最新の 5 つのサービス応答 (および、有効化されている場合は、最新の 5つのサービスリクエスト) のみが保持されます。これらのデフォルトの制限値は、Set-AWSHistoryConfiguration コマンドレットを実行することで変更できます。このコマンドレットを使用すると、リストのサイズ、およびサービスリクエストを記録するかどうかの両方を制御できます。

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

すべてのパラメータは省略可能です。

MaxCmdletHistory パラメータには、いつでも追跡可能なコマンドレットの最大数を設定します。0 を設定すると、AWS コマンドレットアクティビティの記録がオフになります。MaxServiceCallHistory パラメータには、各コマンドレットについて追跡されるサービス応答 (またはサービスリクエスト) の最大数を設定します。RecordServiceRequests パラメータを指定すると、各コマンドレットのサービスリクエストの追跡が有効になります。IncludeSensitiveData パラメータを指定すると、各コマンドレットで機密データを含むサービスレスポンスとリクエストの追跡 (追跡されている場合) が有効になります。

パラメータを一切指定せずに実行すると、Set-AWSHistoryConfiguration は単純に以前のリクエストの記録はすべてオフにして、現在のリストサイズを維持します。

現在の履歴内のすべてのエントリをクリアするには、Clear-AWSHistory コマンドレットを実行します。

$AWSHistory の例

リストに保持されている AWS コマンドレットの詳細をパイプラインに列挙します。

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

実行された最後の AWS コマンドレットの詳細にアクセスします。

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

最後に実行された AWS コマンドレットによって受信された最後のサービス応答の詳細にアクセスします。出力をページングしている AWS コマンドレットでは、複数のサービス呼び出しを起動して、すべてのデータまたは最大量のデータ (どちらになるかはコマンドレットに与えられたパラメーターによって決まる) を取得します。

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

最後に行われたリクエストの詳細にアクセスします (この場合も、コマンドレットがユーザーのためにページングを行っている場合、コマンドレットは複数のリクエストを起動することがあります)。サービスリクエストのトレースが有効になっていない場合は、$null を出力します。

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

複数ページを返す操作のページ自動完成

任意の呼び出しに対してデフォルトで最大オブジェクト返却数を課すサービス API、またはページング可能な結果セットをサポートしているサービス API では、すべてのコマンドレットがデフォルトでページを最後まで完成させます。各コマンドレットは、ユーザーに代わってパイプラインに完全なデータセットを返すために、必要な数だけ呼び出しを実行します。

次の例では、Get-S3Object を使用しています。$c 変数には、test バケット内のすべてのキーの S3Object インスタンスが格納されます (したがって、非常に大きなデータセットになる可能性があります)。

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

返されるデータ量をコントロールする必要がある場合は、個々のコマンドレットにパラメータ (Get-S3Object の MaxKey など) を指定できます。または、自身でページングを明示的に処理することもできます。この処理には、コマンドレットの各種ページングパラメータを組み合わせて使用し、$AWSHistory 変数に格納されたデータを使用してサービスの次のトークンデータを取得します。次の例では、MaxKeys パラメータを使用して、返される S3Object インスタンスの数を、バケット内に見つかった最初の 500 個以下に制限しています。

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

まだ返されていない使用可能なデータが残っているかどうかを確認するには、コマンドレットによって行われたサービス呼び出しを記録した $AWSHistory セッション変数エントリを使用します。

次の式が $true に評価される場合は、$AWSHistory.LastServiceResponse.NextMarker を使用して、次の結果セットの next マーカーを見つけることができます。

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

Get-S3Object を使用して手動でページングを制御するには、コマンドレットの MaxKey パラメータと Marker パラメータを組み合わせて指定し、最後に記録された応答の IsTruncated/NextMarker メモを使用します。次の例では、変数 $c に、バケットの指定されたキープレフィックスマーカーの開始後に見つかった次の 500 個のオブジェクトについて、最大 500 個の S3Object インスタンスが格納されます。

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

認証情報とプロファイルの解決

認証情報の検索順序

コマンドを実行すると、AWS Tools for PowerShell は、次の順序で認証情報を検索します。使用可能な認証情報が見つかると停止します。

1. コマンドラインにパラメータとして埋め込まれているリテラル認証情報。

   コマンドラインにリテラル認証情報を入力するのではなく、プロファイルを使用することを強くお勧めします。
2. 指定されたプロファイル名またはプロファイルの場所。
   - プロファイル名のみを指定した場合、AWS SDK ストアにある指定のプロファイルが検索されますが、そのようなプロファイルが存在しない場合は、デフォルトの場所にある AWS 共有認証情報ファイルの指定プロファイルが使用されます。
   - プロファイルの場所のみを指定した場合、コマンドはその認証情報ファイルから `default` プロファイルを検索します。
   - 名前と場所の両方を指定した場合、コマンドはその認証情報ファイルで指定したプロファイルを検索します。

   指定されたプロファイルまたは場所が見つからない場合、このコマンドは例外をスローします。プロファイルとロケーションを両方とも指定しなかった場合のみ、以下の手順で検索が行われます。
3. `-Credential` パラメータで指定された認証情報。
4. セッションプロファイル (存在する場合)。
5. 次の順序で、デフォルトのプロファイルを使用します。
   a. AWS SDK ストアの `default` プロファイル。
   b. AWS 共有認証情報ファイル内の `default` プロファイル。
   c. AWS SDK ストアの `AWS PS Default` プロファイル。
6. IAM ロールを使用するように設定された Amazon EC2 インスタンスでコマンドが実行されている場合、EC2 インスタンスの一時的な認証情報は、インスタンスプロファイルからアクセスされます。

   Amazon EC2 インスタンスでの IAM ロールの使用の詳細については、「AWS SDK for .NET」を参照してください。

この検索により指定された認証情報が検索できなかった場合、このコマンドは例外をスローします。

ユーザーとロールに関する追加情報

AWS で Tools for PowerShell コマンドを実行するには、タスクに適したユーザー、アクセス許可セット、およびサービスロールの組み合わせが必要です。

作成する特定のユーザー、アクセス許可セットとサービスロール、およびそれらの使用方法は、要件によって異なります。それらが使用される理由とその作成方法について、以下でさらに説明します。

ユーザーとアクセス許可セット

長期的な認証情報を持つ IAM ユーザーアカウントを使用して AWS のサービスにアクセスすることは可能ですが、これはもはやベストプラクティスではなく、避ける必要があります。開発中であっても、AWS IAM Identity Center でユーザーとアクセス許可セットを作成し、ID ソースから提供される一時的な認証情報を使用するのがベストプラクティスです。

開発には、自分で作成したユーザー、または ツール認証を設定する で付与されたユーザーを使用できます。適切な AWS Management Console アクセス許可があれば、そのユーザー用の最小特権のアクセス許可で別のアクセス許可セットを作成するか、開発プロジェクト専用の新しいユーザーを作成して、最小特権のアクセス許可セットを提供できます。選択する行動方針 (ある場合) は、状況によって異なります。

これらのユーザーとアクセス許可セット、および作成方法の詳細については、「AWS SDK とツールリファレンスガイド」の「認証とアクセス」と、「AWS IAM Identity Centerユーザーガイド」の「はじめに」を参照してください。

サービスロール

AWS サービスロールを作成すると、ユーザーに代わって AWS サービスにアクセスできます。このタイプのアクセスは、複数のユーザーがアプリケーションをリモートで実行する場合 (たとえば、この目的のために作成した Amazon EC2 インスタンス上で実行する場合など) に適しています。

サービスロールを作成するプロセスは状況によって異なりますが、基本的に次のプロセスを実行します。

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

2. [ロール]、[ロールの作成] の順に選択します。
3. [AWS service] (AWS サービス) を選択して、[EC2] (この例の場合) を見つけて選択し、[EC2] ユースケース (この例の場合) を選択します。
4. [次へ] を選択し、アプリケーションが使用する AWS サービスに適切なポリシーを選択します。

> **Warning**
>
> AdministratorAccess ポリシーは選択しないでください。このポリシーを選択すると、お使いのアカウント内のほぼすべての情報の読み取りと書き込みのアクセス許可が有効になります。

5. [Next] (次へ) をクリックします。ロール名、説明、および必要なタグを入力します。

   タグに関する情報については、IAM ユーザーガイドの「AWS リソースタグ使用したアクセスコントロール」を参照してください。
6. [Create role] (ロールの作成) を選択します。

IAM ロールに関する概要については、IAM ユーザーガイドの「IAM ID (ユーザー、グループ、ロール)」を参照してください。ロールの詳細については、同ガイドの「IAM ロール」トピックを参照してください。

レガシー認証情報の使用

このセクションのトピックでは、AWS IAM Identity Center を使用せずに長期または短期認証情報を使用する方法について説明します。

> **Warning**
>
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

> **Note**
>
> これらのトピックの情報は、短期または長期認証情報を手動で取得および管理する必要がある場合を対象としています。短期認証情報と長期認証情報に関する追加情報については、

「AWS SDK およびツールリファレンスガイド」の「その他の認証方法」を参照してください。
セキュリティのベストプラクティスについては、ツール認証を設定する の説明に従って AWS IAM Identity Center を使用してください。

認証情報に関する重要な警告とガイダンス

認証情報に関する警告

- お使いのアカウントのルート認証情報を使用して AWS リソースにアクセスしないでください。これらの認証情報は無制限のアカウントアクセスを提供し、取り消すのが困難です。
- コマンドやスクリプトには、リテラルアクセスキーや認証情報を入れないでください。これを行うと、認証情報が誤って公開されるリスクがあります。
- 共有 AWS `credentials` ファイル中に格納された認証情報は平文で格納される点に注意してください。

認証情報を安全に管理するための追加のガイダンス

AWS 認証情報を安全に管理する方法の全般的な説明については、「AWS 全般のリファレンス」の「AWS セキュリティ認証情報」、および「IAM ユーザーガイド」の「セキュリティのベストプラクティスとユースケース」を参照してください。これらの説明に加えて、以下の点を考慮してください。

- IAM Identity Center のユーザーなど、追加ユーザーを作成し、AWS ルートユーザー認証情報を使用する代わりにそのユーザー認証情報を使用します。他のユーザーの認証情報は、必要に応じて取り消すこともできますが、本質的に一時的なものです。さらに、各ユーザーに対して、特定のリソースとアクションにアクセスするためのポリシーを適用できます。これにより、最小特権のアクセス許可になります。

- Amazon Elastic Container Service (Amazon ECS) タスクで、タスク用の IAM ロールを使用します。

- Amazon EC2 インスタンスで実行中のアプリケーションに対して、IAM ロールを使用します。

トピック

- AWS 認証情報の使用
- AWS Tools for PowerShell での認証情報の共有

AWS 認証情報の使用

各 AWS Tools for PowerShell コマンドには AWS 認証情報のセットが含まれている必要があり、これを使用して対応するウェブサービスリクエストが暗号で署名されます。コマンド個別、セッション個別、すべてのセッションに対して認証情報が指定できます。

> **Warning**
>
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

> **Note**
>
> このトピックの情報は、短期または長期認証情報を手動で取得および管理する必要がある場合を対象としています。短期認証情報と長期認証情報に関する追加情報については、「AWS SDK およびツールリファレンスガイド」の「その他の認証方法」を参照してください。セキュリティのベストプラクティスについては、ツール認証を設定する の説明に従って AWS IAM Identity Center を使用してください。

ベストプラクティスとして、認証情報を公開しないように、コマンドにリテラル認証情報を配置しないでください。代わりに、使用する認証情報セットごとにプロファイルを作成し、このプロファイルを 2 つの認証情報ストアのどちらかに保存します。コマンドで正しいプロファイル名を指定すると、AWS Tools for PowerShell が関連付けられている認証情報を取得します。AWS 認証情報を安全に管理する方法の概要については、「Amazon Web Services 全般のリファレンス」の「AWS アクセスキーを管理するためのベストプラクティス」を参照してください。

> **Note**
>
> 認証情報を取得し、AWS Tools for PowerShell を使用するには、AWS アカウントが必要です。AWS アカウントを作成するには、「AWS Account Management リファレンスガイド」の「使用開始:初めての AWS ユーザーですか?」を参照してください。

トピック

認証情報ストアの場所

AWS Tools for PowerShell では認証情報の保存場所として、2 つのストアのどちらかを使用できます。

- AWS SDK ストアは、認証情報を暗号化してユーザーのホームフォルダに保存します。Windows では、このストアは `C:\Users\`*`username`*`\AppData\Local\AWSToolkit\RegisteredAccounts.json` にあります。

  AWS SDK for .NET および Toolkit for Visual Studio でも AWS SDK ストアを使用できます。
- 共有認証情報ファイルもホームフォルダに配置されますが、認証情報はプレーンテキストとして保存されます。

  デフォルトでは次の場所に認証情報ファイルが保存されます。

  - Windows の場合: `C:\Users\`*`username`*`\.aws\credentials`
  - Mac/Linuxの場合: `~/.aws/credentials`

  AWS SDK と AWS Command Line Interface でも認証情報ファイルが使用できます。AWS ユーザーのコンテキスト外でスクリプトを実行している場合は、認証情報を含むファイルを必ず、すべてのユーザーアカウント (ローカルシステムおよびユーザー) からアクセスできる場所にコピーしてください。

プロファイルの管理

プロファイルを使用すると、AWS Tools for PowerShell でさまざまな認証情報セットを参照できます。AWS Tools for PowerShell コマンドレットを使用して、AWS SDK ストア内のプロファイルを管理できます。AWS SDK ストアのプロファイルは、Toolkit for Visual Studio を使用して管理できます。または、AWS SDK for .NET を使用してプログラムで管理することもできます。認証情報ファイ

ルのプロファイルを管理する方法については、「AWS アクセスキーを管理するためのベストプラクティス」を参照してください。

新しいプロファイルの追加

AWS SDK ストアに新しいプロファイルを追加するには、コマンド Set-AWSCredential を実行します。アクセスキーとシークレットキーは、指定したプロファイル名の下のデフォルトの認証情報ファイルに保存されます。

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

- -AccessKey– アクセスキー ID。
- -SecretKey– シークレットキー。
- -StoreAs– プロファイル名。これは一意である必要があります。デフォルトのプロファイルを指定するには、名前 default を使用します。

プロファイルの更新

AWS SDK ストアは手動で管理する必要があります。たとえば、サービスの認証情報を後から変更する場合、ローカルに保存された認証情報を使用して IAM コンソールからコマンドを実行すると、次のエラーメッセージが表示されて失敗します。

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

プロファイルを更新するには、プロファイルに対して Set-AWSCredential コマンドを繰り返し使用して、新しいアクセスキーとシークレットキーをプロファイルに渡します。

プロファイルのリスト表示

現在の名前のリストを確認するには、次のコマンドを使用します。この例では、Shirley という名前のユーザーは、共有認証情報ファイル (~/.aws/credentials) に保存されている 3 つのプロファイルにアクセスできます。

```
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
```

プロファイルの削除

不要になったプロファイルを削除するには、次のコマンドを使用します。

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

`-ProfileName` パラメータは、削除するプロファイルを指定します。

廃止されたコマンド Clear-AWSCredential は、下位互換性のために引き続き使用できますが、`Remove-AWSCredentialProfile` を使用することをお勧めします。

認証情報の指定

認証情報を指定する方法は複数あります。推奨される方法は、リテラル認証情報をコマンドラインに組み込むのではなく、プロファイルを識別する方法です。AWS Tools for PowerShell は、「認証情報の検索順序」で説明されている検索順序を使用してプロファイルを検索します。

Windows では、AWS SDK ストアに保存された AWS 認証情報は、ログインした Windows ユーザー ID で暗号化されます。別のアカウントを使用して復号化したり、最初に作成されたデバイスとは異なるデバイスで使用したりすることはできません。たとえば、スケジュールされたタスクを実行する場合など、別のユーザーの認証情報でタスクを実行するには、前のセクションで説明した方法により、そのユーザーとしてコンピュータにログオンする場合に使用する認証情報プロファイルをセットアップします。タスクを実行するユーザーとしてログインして認証情報のセットアップ手順を完了し、そのユーザーに適したプロファイルを作成します。その後、ログアウトし、独自の認証情報を使用して再度ログインし、スケジュールされたタスクを設定します。

> **Note**
>
> 共通パラメータ `-ProfileName` を使用してプロファイルを指定します。このパラメータは、以前の AWS Tools for PowerShell リリースでの `-StoredCredentials` パラメータに相当します。下位互換性のために、`-StoredCredentials` も引き続きサポートされています。

デフォルトのプロファイル (推奨)

認証情報は、default という名前のプロファイルに保存すると、ローカルコンピュータのすべての AWS SDK と管理ツールで自動的に検索できます。たとえば、ローカルコンピュータに default という名前のプロファイルがある場合、Initialize-AWSDefaultConfiguration コマンドレットまたは Set-AWSCredential コマンドレットを実行する必要はありません。ツールでは、そのプロファイルに保存されているアクセスとシークレットキーのデータが自動的に使用されます。デフォルトリージョン (Get-DefaultAWSRegion の結果) 以外の AWS リージョンを使用する場合は、Set-DefaultAWSRegion を実行してリージョンを指定できます。

プロファイルが default という名前でなくても、現在のセッションでデフォルトプロファイルとして使用したい場合は、Set-AWSCredential を実行してデフォルトプロファイルとして設定します。

Initialize-AWSDefaultConfiguration を実行すると、すべての PowerShell セッションのデフォルトプロファイルを指定できます。この場合、認証情報はカスタム名のプロファイルから読み込まれ、default プロファイルは指定したプロファイルで上書きされます。

Initialize-AWSDefaultConfiguration は実行しないようお勧めします。ただし、PowerShell セッションを実行している Amazon EC2 インスタンスの起動にインスタンスプロファイルを使用していなくて、認証情報プロファイルを手動で設定する場合は除きます。この場合、認証情報プロファイルに認証情報は含まれません。EC2 インスタンスで Initialize-AWSDefaultConfiguration を実行した結果生じる認証情報プロファイルは、認証情報を直接保存するのではなく、インスタンスメタデータ (自動的に更新される一時的な認証情報を提供する) を指します。ただし、インスタンスのリージョンは保存されます。Initialize-AWSDefaultConfiguration を実行する別のシナリオとして、インスタンスが実行されているリージョン以外のリージョンに対して呼び出しを実行する場合があります。そのコマンドを実行すると、インスタンスメタデータに保存されているリージョンは永続的に上書きされます。

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

> **Note**
>
> デフォルトの認証情報は default プロファイル名の下の AWS SDK ストアに含まれています。このコマンドを使用すると、その名前の既存のプロファイルが上書きされます。

EC2 インスタンスの起動にインスタンスプロファイルを使用している場合、PowerShell は AWS 認証情報とリージョン情報をインスタンスプロファイルから自動的に取得します。Initialize-

AWSDefaultConfiguration を実行する必要はありません。EC2 インスタンスの起動にインスタンスプロファイルを使用している場合、Initialize-AWSDefaultConfiguration コマンドレットの実行は不要です。PowerShell がデフォルトで使用するのと同じインスタンスプロファイルが使用されます。

セッションのプロファイル

特定セッションにデフォルトのプロファイルを指定するには、Set-AWSCredential を使用します。このプロファイルはセッション期間中、デフォルトのプロファイルを上書きします。現在の default プロファイルの代わりにカスタム名のプロファイルをセッションで使用する場合は、この方法をお勧めします。

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

> **Note**
>
> 1.1 より前の Tools for Windows PowerShell バージョンでは、Set-AWSCredential コマンドレットが正しく機能せず、「MyProfileName」で指定したプロファイルが上書きされていました。最新バージョンの Tools for Windows PowerShell を使用することをお勧めします。

コマンドのプロファイル

個々のコマンドでは、-ProfileName パラメータを追加して、その 1 つのコマンドだけに適用されるプロファイルを指定できます。このプロファイルは、次の例に示すように、デフォルトプロファイルまたはセッションプロファイルを上書きします。

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

> **Note**
>
> デフォルトまたはセッションのプロファイルを指定する場合には、-Region パラメータを追加してデフォルトまたはセッションのリージョンを上書きすることもできます。詳細については、「AWS リージョンを指定する」を参照してください。次の例では、デフォルトのプロファイルとリージョンを指定しています。
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-
> west-2
> ```

デフォルトでは、AWS 共有認証情報ファイルはユーザーのホームフォルダ (Windows の場合は `C:\Users\username\.aws`、Linux の場合は `~/.aws`) に配置されます。別の場所に認証情報ファイルを指定するには、`-ProfileLocation` パラメータを含んで、認証情報ファイルのパスを指定します。次の例では、特定のコマンドに対してデフォルト以外の認証情報ファイルを指定しています。

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

> **Note**
>
> 通常の作業時間以外で、スケジュールされたタスクとして PowerShell スクリプトを実行している場合など、AWS に通常サインインしていない時間中に PowerShell スクリプトを実行している場合、使用するプロファイルを指定するには、`-ProfileLocation` パラメータを追加して、その値に認証情報が保存されているファイルのパスを設定します。AWS Tools for PowerShell スクリプトが正しいアカウント認証情報を使用して確実に実行されるようにするには、AWS アカウントを使用しないコンテキストまたはプロセスでスクリプトを実行する場合に必ず `-ProfileLocation` パラメータを追加する必要があります。スクリプトがタスクを実行するために使用するローカルシステムまたは他のアカウントからアクセス可能な場所に、認証情報ファイルをコピーすることもできます。

認証情報の検索順序

コマンドを実行すると、AWS Tools for PowerShell は、次の順序で認証情報を検索します。使用可能な認証情報が見つかると停止します。

1. コマンドラインにパラメータとして埋め込まれているリテラル認証情報。

   コマンドラインにリテラル認証情報を入力するのではなく、プロファイルを使用することを強くお勧めします。

2. 指定されたプロファイル名またはプロファイルの場所。
   - プロファイル名のみを指定した場合、AWS SDK ストアにある指定のプロファイルが検索されますが、そのようなプロファイルが存在しない場合は、デフォルトの場所にある AWS 共有認証情報ファイルの指定プロファイルが使用されます。
   - プロファイルの場所のみを指定した場合、コマンドはその認証情報ファイルから `default` プロファイルを検索します。

- 名前と場所の両方を指定した場合、コマンドはその認証情報ファイルで指定したプロファイルを検索します。

  指定されたプロファイルまたは場所が見つからない場合、このコマンドは例外をスローします。プロファイルとロケーションを両方とも指定しなかった場合のみ、以下の手順で検索が行われます。

3. `-Credential` パラメータで指定された認証情報。
4. セッションプロファイル (存在する場合)。
5. 次の順序で、デフォルトのプロファイルを使用します。
   a. AWS SDK ストアの `default` プロファイル。
   b. AWS 共有認証情報ファイル内の `default` プロファイル。
   c. AWS SDK ストアの `AWS PS Default` プロファイル。
6. IAM ロールを使用するように設定された Amazon EC2 インスタンスでコマンドが実行されている場合、EC2 インスタンスの一時的な認証情報は、インスタンスプロファイルからアクセスされます。

   Amazon EC2 インスタンスでの IAM ロールの使用の詳細については、「AWS SDK for .NET」を参照してください。

この検索により指定された認証情報が検索できなかった場合、このコマンドは例外をスローします。

AWS Tools for PowerShell Core での認証情報の処理

AWS Tools for PowerShell Core のコマンドレットは、AWS Tools for Windows PowerShell と同様に、実行時に AWS アクセスキーとシークレットキー、または認証情報プロファイルの名前を受け入れます。Windows で実行すると、どちらのモジュールも AWS SDK for .NET 認証情報ストアファイル (各ユーザーの `AppData\Local\AWSToolkit\RegisteredAccounts.json` ファイルに保存されています) にアクセスできます。

このファイルにはユーザーのキーが暗号化された形式で保存されていて、別のコンピュータで使用することはできません。これは、AWS Tools for PowerShell が認証情報プロファイルを検索する最初のファイルであり、AWS Tools for PowerShell が認証情報プロファイルを保存するファイルでもあります。AWS SDK for .NET 認証情報ストアファイルの詳細については、「AWS 認証情報の設定」を参照してください。Tools for Windows PowerShell モジュールでは、他のファイルまたは場所への認証情報の書き込みを現在サポートしていません。

どちらのモジュールも、他の AWS SDK および AWS CLI で使用される AWS 共有認証情報ファイルのプロファイルを読み取ることができます。Windows では、このファイルのデフォルトの場所は C:\Users\<userid>\.aws\credentials です。Windows 以外のプラットフォームでは、このファイルは ~/.aws/credentials に保存されています。-ProfileLocation パラメータを使用して、デフォルト以外のファイル名またはファイルの場所を指定することができます。

SDK 認証情報ストアには、Windows 暗号化 API を使用して暗号化された形式の認証情報が保持されています。これらの API は他のプラットフォームでは使用できないため、AWS Tools for PowerShell Core モジュールは、AWS 共有認証情報ファイルを排他的に使用して、共有認証情報ファイルへの新しい認証情報プロファイルの書き込みをサポートします。

以下のスクリプトの例は、Set-AWSCredential コマンドレットを使用して、[AWSPowerShell] または [AWSPowerShell.NetCore] モジュールのいずれかにより Windows で認証情報プロファイルを処理するためのオプションを示しています。

```
# Writes a new (or updates existing) profile with name "myProfileName"
# in the encrypted SDK store file

Set-AWSCredential -AccessKey akey -SecretKey skey -StoreAs myProfileName

# Checks the encrypted SDK credential store for the profile and then
# falls back to the shared credentials file in the default location

Set-AWSCredential -ProfileName myProfileName

# Bypasses the encrypted SDK credential store and attempts to load the
# profile from the ini-format credentials file "mycredentials" in the
# folder C:\MyCustomPath

Set-AWSCredential -ProfileName myProfileName -ProfileLocation C:\MyCustomPath
\mycredentials
```

以下の例は、Linux または Mac OS オペレーティングシステムでの [AWSPowerShell.NetCore] モジュールの動作を示しています。

```
# 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
```

AWS Tools for PowerShell での認証情報の共有

Tools for Windows PowerShell は、AWS CLI および他の AWS SDK と同様に、AWS 共有認証情報ファイルの使用をサポートします。Tools for Windows PowerShell で、.NET 認証情報ファイルおよび AWS 共有認証情報ファイルに対して、`basic`、`session`、および `assume role` 認証情報プロファイルの読み書きをサポートするようになりました。この機能は、新しい `Amazon.Runtime.CredentialManagement` 名前空間で有効になります。

> **Warning**
>
> セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center などの ID プロバイダーとのフェデレーションを使用してください。

> **Note**
>
> このトピックの情報は、短期または長期認証情報を手動で取得および管理する必要がある場合を対象としています。短期認証情報と長期認証情報に関する追加情報については、「AWS SDK およびツールリファレンスガイド」の「その他の認証方法」を参照してください。セキュリティのベストプラクティスについては、ツール認証を設定する の説明に従って AWS IAM Identity Center を使用してください。

新しいプロファイルタイプと AWS 共有認証情報ファイルへのアクセスは、認証情報に関連するコマンドレットに追加された各パラメータ、Initialize-AWSDefaultConfiguration、New-AWSCredential、Set-AWSCredential でサポートされます。サービスコマンドレットで共通パラメータ `-ProfileName` を追加すると、新しいプロファイルを参照できます。

AWS Tools for PowerShell での IAM ロールの使用

AWS 共有認証情報ファイルを使用すると、追加の種類のアクセスが可能になります。例えば、IAM ユーザーの長期認証情報の代わりに IAM ロールを使用して AWS リソースにアクセスできます。これを行うには、ロールを継承するアクセス許可を持つ標準プロファイルが必要です。ロールを指定したプロファイルを使用するように AWS Tools for PowerShell に指示すると、AWS Tools for PowerShell は `SourceProfile` パラメータで識別されるプロファイルを検索します。これらの認証情報は、`RoleArn` パラメータで指定されたロールの一時的な認証情報を要求するために使用されます。ロールが第三者によって継承される場合は、オプションで、多要素認証 (MFA) デバイスまたは `ExternalId` コードの使用を要求できます。

Parameter Name	説明
ExternalId	ロールを引き受ける際に使用するユーザー定義の外部 ID (ロールで必要とされる場合)。これは通常、アカウントへのアクセス権を第三者に委任する場合にのみ必要です。第三者は、割り当てられたロールを継承するときに、パラメータとして ExternalId を含める必要があります。詳細については、IAM ユーザーガイドの「AWS リソースへのアクセス権を第三者に付与するときに外部 ID を使用する方法」を参照してください。
MfaSerial	ロールを引き受ける際に使用する MFA シリアル番号 (ロールで必要とされる場合)。詳細については、IAM ユーザーガイドの「AWS での多要素認証 (MFA) の使用」を参照してください。
RoleArn	ロールの継承認証情報を引き受けるロールの ARN。ロールの作成と使用の詳細について

Parameter Name	説明
	は、IAM ユーザーガイドの IAM ロールを参照してください。
SourceProfile	ロールの継承認証情報によって使用されるソースプロファイルの名前。このプロファイルで見つかった認証情報は、RoleArn パラメータで指定されたロールを継承するために使用されます。

ロールを継承するためのプロファイルの設定

次に、IAM ロールを直接継承することができるソースプロファイルを設定する方法の例を示します。

最初のコマンドは、ロールプロファイルが参照するソースプロファイルを作成します。2 番目のコマンドは、継承するロールプロファイルを作成します。3 番目のコマンドは、ロールプロファイルの認証情報を表示します。

```
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
```

このロールプロファイルを Tools for Windows PowerShell サービスコマンドレットで使用するには、ロールプロファイルを参照するコマンドに -ProfileName 共通パラメータを追加します。次の例では、前の例で定義されたロールプロファイルを使用して Get-S3Bucket コマンドレットにアクセスします。AWS Tools for PowerShell は my_source_profile で認証情報を検索し、それらの認証情報を使用してユーザーの代わりに AssumeRole を呼び出し、それらの一時的なロールの認証情報を使用して Get-S3Bucket を呼び出します。

```
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
```

認証情報プロファイルタイプの使用

認証情報プロファイルタイプを設定するには、プロファイルのタイプで必要とされる情報を提供するパラメーターを把握します。

認証情報タイプ	使用する必要があるパラメータ

Basic (ベーシック)
 	
これらは、IAM ユーザーの長期的な認証情報です。
	`-AccessKey`
	
	`-SecretKey`

セッション:
 	
これらは、Use-STSRole コマンドレットを直接呼び出すなど、手動で取得する IAM ロールの短期的な認証情報です。
	`-AccessKey`
	
	`-SecretKey`
	
	`-SessionToken`

ロール:
 	
これらは、AWS Tools for PowerShell が取得する IAM ロールの短期認証情報です。
	`-SourceProfile`
	
	`-RoleArn`
	
	オプション: `-ExternalId`
	
	オプション: `-MfaSerial`

`ProfilesLocation` 共通パラメータ

`-ProfileLocation` を使用して共有認証情報ファイルに書き出すことができると同時に、コマンドレットに認証情報ファイルから読み取るよう指示を出すこともできます。`-ProfileLocation` パラメータを追加することで、共有認証情報ファイルと .NET 認証情報ファイルのどちらを Tools for

Windows PowerShell で使用するかを制御できます。次の表では、Tools for Windows PowerShell でのこのパラメータの動作方法について説明します。

プロファイルの場所の値	プロファイルの解決動作
null (未設定) または空	最初に、.NET 認証情報ファイル内で指定された名前のプロファイルを検索します。プロファイルが見つからない場合は、*(user's home directory)* \.aws\credentials で AWS 共有認証情報ファイルを検索します。
AWS 共有認証情報ファイル形式のファイルへのパス	指定されたファイルのみを対象に、指定された名前のプロファイルを検索します。

認証情報の認証情報ファイルへの保存

2 つの認証情報ファイルのどちらか一方に、認証情報を書き込み、保存するには、Set-AWSCredential コマンドレットを実行します。次の例は、その方法を示しています。最初のコマンドは、Set-AWSCredential を -ProfileLocation と共に使用して、-ProfileName パラメータで指定されたプロファイルにアクセスキーとシークレットキーを追加します。2 行目では、Get-Content コマンドレットを実行して認証情報ファイルの内容を表示しています。

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

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

認証情報プロファイルの表示

Get-AWSCredential コマンドレットを実行して、-ListProfileDetail パラメータを追加すると、認証情報ファイルのタイプと場所、プロファイル名のリストが返されます。

```
PS > Get-AWSCredential -ListProfileDetail

ProfileName                   StoreTypeName         ProfileLocation
-----------                   -------------         ---------------
source_profile                NetSDKCredentialsFile
```

```
assume_role_profile             NetSDKCredentialsFile
basic_profile                   SharedCredentialsFile C:\Users\auser\.aws\credentials
```

認証情報プロファイルの削除

認証情報プロファイルを削除するには、新しい Remove-AWSCredentialProfile コマンドレットを実行します。Clear-AWSCredential は廃止されましたが、下位互換性のために引き続き使用できます。

重要な注意点

ロールプロファイルのパラメータがサポートされるのは、Initialize-AWSDefaultConfiguration、New-AWSCredential、Set-AWSCredential のみです。`Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name` などのコマンドでは、ロールパラメータを直接指定することはできません。サービスコマンドレットは `SourceProfile` または `RoleArn` パラメータを直接サポートしていないため、これは機能しません。代わりに、これらのパラメータをプロファイルに保存し、`-ProfileName` パラメータを指定してコマンドを呼び出す必要があります。

AWS Tools for PowerShell での AWS サービスの操作

このセクションでは、AWS Tools for PowerShell を使用して AWS のサービスにアクセスする例を示します。以下の例では、コマンドレットを使用して実際の AWS タスクを実行する方法を示します。これらの例は、Tools for PowerShell が提供するコマンドレットに依存しています。使用できるコマンドレットについては、「AWS Tools for PowerShell コマンドレットリファレンス」を参照してください。

PowerShell ファイルの連結エンコード

AWS Tools for PowerShell の一部のコマンドレットは、AWS にある既存のファイルまたはレコードを編集します。一例は `Edit-R53ResourceRecordSet` です。これは Amazon Route 53 の ChangeResourceRecordSets API を呼び出します。

PowerShell 5.1 以前のリリースでファイルを編集または連結すると、PowerShell は UTF-8 ではなく UTF-16 で出力をエンコードします。これにより、不要な文字が追加されたり、有効でない結果が作成されたりする場合があります。16 進数エディタを使用すると、不要な文字を表示できます。

ファイル出力が UTF-16 に変換されないようにするには、次の例に示すように、PowerShell の `Out-File` コマンドレットにコマンドをパイプ処理し、UTF-8 エンコードを指定します。

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

PowerShell コンソールで AWS CLI コマンドを実行している場合、同じ動作が適用されます。AWS CLI コマンドの出力を PowerShell コンソールの `Out-File` にパイプ処理できます。`Export-Csv` や `Export-Clixml` など、その他のコマンドレットにも `Encoding` パラメータがあります。`Encoding` パラメータを持つコマンドレット、および連結されたファイル出力のエンコードを修正できるコマンドレットの詳細なリストについては、次のコマンドを実行します。

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

> **Note**
>
> PowerShell Core を含む PowerShell 6.0 以降では、連結されたファイル出力の UTF-8 エンコードが自動的に保持されます。

PowerShell ツールに対して返されるオブジェクト

ネイティブの PowerShell 環境で AWS Tools for PowerShell をより便利にするために、AWS Tools for PowerShell コマンドレットによって返されるオブジェクトは .NET オブジェクトであり、AWS SDK の対応する API から通常返される JSON テキストオブジェクトではありません。例えば、`Get-S3Bucket` は、Amazon S3 JSON 応答オブジェクトではなく、`Buckets` コレクションを返します。`Buckets` コレクションは PowerShell パイプラインに配置し、適切な方法で操作できます。同様に、`Get-EC2Instance` は `DescribeEC2Instances` JSON 結果オブジェクトではなく、`Reservation` .NET オブジェクトコレクションを出力します。この動作は設計によるものであり、AWS Tools for PowerShell と慣用的な PowerShell との一貫性を向上させるためのものです。

実際のサービス応答は、返されたオブジェクトの `note` プロパティとして保存され、必要であれば利用できます。`NextToken` フィールドを使用してページングをサポートする API アクションの場合、`note` プロパティとしても添付されます。

Amazon EC2

このセクションでは、以下の方法を含む、Amazon EC2 インスタンスを起動するために必要な手順について説明します。

- Amazon Machine Image (AMI) のリストを取得する
- SSH 認証のキーペアを作成する
- Amazon EC2 セキュリティグループを作成して設定する。
- インスタンスの起動、インスタンスに関する情報を取得する

Amazon S3

このセクションでは、Amazon S3 でホストされる静的ウェブサイトを作成するために必要な手順について説明します。以下の方法について説明します。

- Amazon S3 バケットの作成と削除を行う
- Amazon S3 バケットへのオブジェクトとしてファイルをアップロードする
- Amazon S3 バケットからオブジェクトを削除する
- Amazon S3 バケットをウェブサイトとして指定する

AWS Lambda および AWS Tools for PowerShell

このセクションでは、AWS Lambda Tools for PowerShell モジュールを簡単に紹介し、このモジュールの設定に必要なステップについて説明します。

Amazon SNS と Amazon SQS

このセクションでは、Amazon SQS キューを Amazon SNS トピックにサブスクライブするために必要なステップについて説明します。以下の方法について説明します。

- Amazon SNS トピックを作成します。
- Amazon SQS キューを作成します。
- キューをトピックにサブスクライブする。
- メッセージをトピックに送信する。
- キューからメッセージを受信する。

CloudWatch

このセクションでは、カスタムデータを CloudWatch に発行する方法を例を挙げて説明します。

- カスタムメトリクスを CloudWatch ダッシュボードに発行する。

以下も参照してください。

- AWS Tools for Windows PowerShell の開始方法

トピック

- Amazon S3 と Tools for Windows PowerShell
- Amazon EC2 と Tools for Windows PowerShell
- AWS Lambda および AWS Tools for PowerShell
- Amazon SQS、Amazon SNS、および Tools for Windows PowerShell
- AWS Tools for Windows PowerShell からのCloudWatch

- コマンドレットでの ClientConfig パラメータの使用

Amazon S3 と Tools for Windows PowerShell

このセクションでは、Amazon S3 と CloudFront で AWS Tools for Windows PowerShell を使用して静的ウェブサイトを作成します。このプロセスでは、これらのサービスに共通な多くのタスクを扱います。このウォークスルーは、「静的ウェブサイトをホスティングする」の入門ガイドです。このガイドでは、AWS マネジメントコンソールを使用した同様なプロセスを説明しています。

ここで示すコマンドは、ユーザーの PowerShell セッションのデフォルトの認証情報とデフォルトのリージョンが設定済みであることを前提としています。したがって、認証情報とリージョンはコマンドレットの呼び出しには含まれません。

> **Note**
>
> 現時点では、バケットまたはオブジェクトの名前を変更する Amazon S3 API はないため、このタスクを実行する単一の Tools for Windows PowerShell コマンドレットはありません。S3 のオブジェクトの名前を変更するには、Copy-S3Object コマンドレットを実行して、新しい名前でオブジェクトをコピーしてから、Remove-S3Object コマンドレットを実行して元のオブジェクトを削除することをお勧めします。

以下も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- Amazon S3 で静的ウェブサイトをホスティングする
- Amazon S3 コンソール

トピック

- Amazon S3 バケットの作成、そのリージョンの確認、および必要に応じたバケットの削除
- Amazon S3 バケットをウェブサイトとして設定し、ログを有効にする
- オブジェクトの Amazon S3 バケットへのアップロード
- Amazon S3 オブジェクトとバケットの削除
- インラインテキストコンテンツの Amazon S3 へのアップロード

Amazon S3 バケットの作成、そのリージョンの確認、および必要に応じたバケットの削除

新しい Amazon S3 バケットを作成するには、New-S3Bucket コマンドレットを使用します。次の例では、website-example という名前のバケットを作成します。バケットの名前はすべてのリージョン間で一意である必要があります。この例では、us-west-1 リージョンにバケットを作成します。

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

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

バケットがあるリージョンを確認するには、Get-S3BucketLocation コマンドレットを使用します。

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

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

このチュートリアルを終了したら、次の行を使用してこのバケットを削除できます。このバケットは以降の例で使用するため、そのままにしておくことをお勧めします。

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

バケットの削除プロセスは完了するまでに時間がかかる場合があります。同じ名前のバケットをすぐに再作成しようとすると、古いバケットが完全に削除されるまで New-S3Bucket コマンドレットが失敗することがあります。

以下の資料も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- PUT Bucket (Amazon S3 サービスリファレンス)
- Amazon S3 の AWS PowerShell リージョン

Amazon S3 バケットをウェブサイトとして設定し、ログを有効にする

`Write-S3BucketWebsite` コマンドレットを使用して、Amazon S3 バケットを静的ウェブサイトとして設定します。次の例では、デフォルトのコンテンツウェブページの `index.html` の名前とデフォルトのエラーウェブページの `error.html` の名前を指定します。このコマンドレットは、これらのページを作成しません。Amazon S3 オブジェクトとしてアップロードする必要があります。

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

以下の資料も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- PUT Bucket website (Amazon S3 API リファレンス)
- PUT Bucket acl (Amazon S3 API リファレンス)

オブジェクトの Amazon S3 バケットへのアップロード

`Write-S3Object` コマンドレットでは、ローカルファイルシステムのファイルをオブジェクトとして Amazon S3 バケットにアップロードします。以下の例では、2 つの簡単な HTML ファイルを作成して、Amazon S3 バケットにアップロードし、アップロードされたオブジェクトが存在するかどうかを確認します。`-File` への `Write-S3Object` パラメータは、ローカルファイルシステム内のファイルの名前を指定します。`-Key` パラメータは、Amazon S3 での対応するオブジェクトの名前を指定します。

Amazon は、ファイルの拡張子、この場合、「.html」からオブジェクトのコンテンツタイプを推論します。

```
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
```

既定 ACL オプション

Tools for Windows PowerShell で既定 ACL を指定するための値は、AWS SDK for .NET によって使用される値と同じです。ただし、Amazon S3 `Put Object` アクションによって使用される値とは異なります。Tools for Windows PowerShellでは、次の既定 ACL がサポートされています。

- NoACL
- プライベート
- public-read
- public-read-write

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

この既定 ACL 設定の詳細については、「アクセスコントロールリスト (ACL) の概要」を参照してください。

マルチパートアップロードに関する注意事項

Amazon S3 API を使用して、5 GB よりも大きいファイルをアップロードする場合、マルチパートアップロードを使用する必要があります。ただし、Tools for Windows PowerShell が提供する `Write-S3Object` コマンドレットでは、5 GB よりも大きいファイルのアップロードを透過的に処理します。

ウェブサイトをテストする

この時点で、ブラウザを使用して移動することで、ウェブサイトをテストできます。Amazon S3 でホストされる静的ウェブサイトの URL は、標準形式に従います。

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

例:

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

以下の資料も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- PUT オブジェクト (Amazon S3 API リファレンス)
- 既定 ACL (Amazon S3 API リファレンス)

Amazon S3 オブジェクトとバケットの削除

このセクションでは、以前のセクションで作成したウェブサイトを削除する方法について説明します。HTML ファイルのオブジェクトを単純に削除し、その後にサイトの Amazon S3 バケットを削除します。

まず、Amazon S3 バケットから HTML ファイルのオブジェクトを削除するには、`Remove-S3Object` コマンドレットを実行します。

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

`False` レスポンスは、Amazon S3 のリクエスト処理で予期されるアーティファクトです。この場合、このレスポンスは問題を示しているわけではありません。

この時点で、`Remove-S3Bucket` コマンドレットを実行して、サイトの空になった Amazon S3 バケットを削除することができます。

```
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    :
```

バージョン 1.1 以降の AWS Tools for PowerShell では、`-DeleteBucketContent` パラメータを `Remove-S3Bucket` に追加できます。このパラメータにより、指定したバケット内のすべてのオブジェクトとオブジェクトバージョンが削除された後でバケット自体が削除されます。バケット内のオブジェクトまたはオブジェクトのバージョンの数によっては、この操作にかなりの時間がかかることがあります。バージョン 1.1 より前の Tools for Windows PowerShell では、`Remove-S3Bucket` を使用してバケットを削除する前に、バケットを空にする必要がありました。

> **Note**
>
> `-Force` パラメータを追加しない限り、コマンドレットが実行される前に AWS Tools for PowerShell により確認プロンプトが表示されます。

以下の資料も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- DELETE オブジェクト (Amazon S3 API リファレンス)
- DeleteBucket (Amazon S3 API リファレンス)

インラインテキストコンテンツの Amazon S3 へのアップロード

`Write-S3Object` コマンドレットでは、インラインテキストコンテンツを Amazon S3 にアップロードする機能をサポートしています。`-Content` パラメータ (別名 `-Text`) を使用して、Amazon S3 にアップロードするテキストベースのコンテンツを指定することができます。この場合、事前にコンテンツをファイルに格納する必要はありません。このパラメーターでは、簡単な 1 行の文字列や、次のような複数の行が含まれている文字列が指定できます。

```
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 と Tools for Windows PowerShell

AWS Tools for PowerShell を使用して Amazon EC2 に関連する一般的なタスクを実行できます。

以下に示すコマンド例では、PowerShell セッションのデフォルトの認証情報とデフォルトのリージョンは設定済みであるとものとします。したがって、コマンドレットを呼び出す際に認証情報やリージョンは指定していません。詳細については、「AWS Tools for Windows PowerShell の開始方法」を参照してください。

トピック

- キーペアを作成する
- Windows を使用してセキュリティグループを作成する PowerShell
- Windows PowerShell を使用した Amazon Machine Image の検索
- Windows を使用して Amazon EC2 インスタンスを起動する PowerShell

キーペアを作成する

次の New-EC2KeyPair 例では、キーペアを作成し、PowerShell 変数 $myPSKeyPair に保存します。

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

キーペアオブジェクトを Get-Member コマンドレットにパイプ処理すると、オブジェクトの構造が表示されます。

```
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;}
```

キーペアオブジェクトを Format-List コマンドレットにパイプ処理すると、KeyName、KeyFingerprint、および KeyMaterial の各メンバーの値が表示されます (出力は読みやすくするために切り詰められます)。

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

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

KeyMaterial メンバーにはキーペアのプライベートキーが保存されます。パブリックキーは AWS に保存されます。AWS からパブリックキーを取得することはできませんが、プライベートキーの KeyFingerprint を、AWS から返されたパブリックキーのフィンガープリントと比較することで、パブリックキーを確認することができます。

キーペアのフィンガープリントの表示

Get-EC2KeyPair コマンドレットを使用して、キーペアのフィンガープリントを表示できます。

```
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
```

プライベートキーの保存

ファイルにプライベートキーを保存するには、KeyFingerMaterial メンバーを Out-File コマンドレットにパイプ処理します。

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

プライベートキーをファイルに書き込む際には、-Encoding ascii を指定する必要があります。指定しない場合、openssl などのツールがファイルを正しく読み取ることができなくなる可能性があります。作成したファイルの形式が正しいことを確認するには、次のようなコマンドを使用します。

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

(openssl ツールは AWS Tools for PowerShell または AWS SDK for .NET には含まれていません。)

キーペアの削除

インスタンスを起動して接続するにはキーペアが必要です。キーペアを使用し終わったら、削除できます。AWS からパブリックキーを削除するには、Remove-EC2KeyPair コマンドレットを使用します。プロンプトが表示されたら、Enter キーを押して、キーペアを削除します。

```
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"):
```

現行の PowerShell セッションにはまだ $myPSKeyPair 変数が存在しており、キーペア情報が含まれています。また、myPSKeyPair.pem ファイルも存在します。ただし、キーペアのパブリックキーが AWS に保存されていないため、プライベートキーは使用できません。

Windows を使用してセキュリティグループを作成する PowerShell

を使用して AWS Tools for PowerShell 、セキュリティグループを作成および設定できます。セキュリティグループを作成するときは、EC2-Classic 用のセキュリティグループなのか、EC2-VPC 用のセキュリティグループなのかを指定します。応答はセキュリティグループの ID です。

インスタンスに接続する必要がある場合、SSH トラフィック (Linux) または RDP トラフィック (Windows) を許可するようにセキュリティグループを設定する必要があります。

トピック

前提条件

コンピュータの CIDR 表記のパブリック IP アドレスが必要です。サービスを使用して、ローカルコンピュータのパブリック IP アドレスを取得できます。たとえば、Amazon では、http://checkip.amazonaws.com/ または https://checkip.amazonaws.com/ のサービスを提供しています。IP アドレスを提供する別のサービスを検索するには、検索フレーズ「what is my IP address」を使用します。ISP 経由で、またはファイアウォールの内側から静的な IP アドレスなしで接続している場合は、クライアントコンピュータで使用できる IP アドレスの範囲を見つける必要があります。

> **Warning**
>
> `0.0.0.0/0` を指定すると、世界中の任意の IP アドレスからのトラフィックが有効になります。SSH と RDP プロトコルの場合、これはテスト環境で短時間なら許容できますが、実稼働環境で行うのは安全ではありません。実稼働環境では、適切な個別の IP アドレスまたはアドレス範囲からのアクセスのみを許可するようにしてください。

EC2-Classic 用セキュリティグループの作成

> **Warning**
>
> 2022 年 8 月 15 日に、EC2-Classic の提供を終了し h ます。EC2-Classic は、VPC への移行をお勧めします。詳細については、Amazon EC2 EC2-Classic から VPC への移行Amazon

EC2」を参照してください。ブログ記事「EC2-Classic Networking は販売終了になります — 準備方法はこちら」も参照してください。

次の例では、New-EC2SecurityGroup コマンドレットを使用して EC2-Classic のセキュリティグループを作成します。

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

sg-0a346530123456789
```

セキュリティグループの初期設定を表示するには、Get-EC2SecurityGroup コマンドレットを使用します。

```
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
```

TCP ポート 22 (SSH) と TCP ポート 3389 でインバウンドトラフィックを許可するようにセキュリティグループを設定するには、Grant-EC2SecurityGroupIngress コマンドレットを使用します。たとえば、次のスクリプト例では、単一の IP アドレス 203.0.113.25/32 からの SSH トラフィックを有効にする方法を示します。

```
$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
```

セキュリティグループが更新されているかどうかを確認するには、再度、Get-EC2SecurityGroup コマンドレットを実行します。EC2-Classic のアウトバウンドルールは指定できません。

```
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                : {}
```

セキュリティグループのルールを表示するには、IpPermissions プロパティを使用します。

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

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

EC2-VPC 用セキュリティグループの作成

次の New-EC2SecurityGroup 例では、指定された VPC のセキュリティグループを作成するために -VpcId パラメータを追加します。

```
PS > $groupid = New-EC2SecurityGroup `
    -VpcId "vpc-da0013b3" `
    -GroupName "myPSSecurityGroup" `
    -GroupDescription "EC2-VPC from PowerShell"
```

セキュリティグループの初期設定を表示するには、Get-EC2SecurityGroup コマンドレットを使用します。デフォルトでは、VPC 用のセキュリティグループにはすべてのアウトバウンドトラフィックを許可するルールが含まれています。EC2-VPC 用セキュリティグループは名前では参照できないことに注意してください。

```
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                : {}
```

TCP ポート 22 (SSH) および TCP ポート 3389 のインバウンドトラフィックのアクセス許可を定義するには、New-Object コマンドレットを使用します。次のスクリプト例では、単一の IP アドレス、203.0.113.25/32 から TCP ポート 22 および 3389 のアクセス許可を定義します。

```
$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 )
```

セキュリティグループが更新されているかどうかを確認するには、再度、Get-EC2SecurityGroup コマンドレットを使用します。

```
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                : {}
```

インバウンドルールを表示するには、前のコマンドで返されたコレクションオブジェクトから IpPermissions プロパティを取得します。

```
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}
```

Windows PowerShell を使用した Amazon Machine Image の検索

Amazon EC2 インスタンスを起動する際には、インスタンスのテンプレートとなる Amazon Machine Image (AMI) を指定します。ただし、AWS では、最新の更新とセキュリティ強化が適用された新しい AMI を提供するため、AWS Windows AMI の ID は頻繁に変更されます。Get-EC2Image コマンドレットおよび Get-EC2ImageByName コマンドレットを使用し、最新の Windows AMI を検索して ID を取得できます。

トピック

- Get-EC2Image
- Get-EC2ImageByName

Get-EC2Image

`Get-EC2Image` コマンドレットは、使用できる AMI のリストを取得します。

`-Owner` パラメータに配列値 `amazon, self` を指定して、`Get-EC2Image` が、Amazon またはユーザー自身に属する AMI のみを取得するようにします。このコンテキストでは、ユーザー とは、コマンドレットの呼び出しに使用した認証情報を持つユーザーを指します。

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

`-Filter` パラメータを使用して結果を絞り込むことができます。フィルタを指定するには、`Amazon.EC2.Model.Filter` 型のオブジェクトを作成します。たとえば、次のフィルタを使用すると、Windows AMI のみが表示されます

```
$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
```

次の例は、コマンドレットによって返される AMI の 1 つです。上記のコマンドの実際の出力では、多くの AMI の情報が提供されます。

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

Get-EC2ImageByName

`Get-EC2ImageByName` コマンドレットを使用すると、ユーザーが関心のあるサーバー設定のタイプに基づいて AWS Windows AMI のリストをフィルタリングできます。

次のようにパラメータを指定せずに実行すると、コマンドレットは、現在のすべてのフィルター名を出力します。

```
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
```

返されるイメージのセットを絞り込むには、Names パラメータに 1 つ以上のフィルタ名を指定します。

```
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
```

Windows を使用して Amazon EC2 インスタンスを起動する PowerShell

Amazon EC2 インスタンスの起動には、前のセクションで作成したキーペアとセキュリティグループが必要です。また、Amazon Machine Image (AMI) の ID も必要です。詳細については、次のドキュメントを参照してください。

- キーペアを作成する
- Windows を使用してセキュリティグループを作成する PowerShell
- Windows を使用して Amazon マシンイメージを検索する PowerShell

> **Important**
>
> 無料利用枠に含まれないインスタンスを起動する場合は、インスタンスを起動すると料金が発生し、そのインスタンスの実行中はアイドル状態であっても料金がかかります。

トピック

- EC2-Classic でのインスタンスの起動
- VPC でのインスタンスの起動
- VPC でのスポットインスタンスの起動

EC2-Classic でのインスタンスの起動

> **Warning**
>
> 2022 年 8 月 15 日に、EC2-Classic の提供を終了し h ます。EC2-Classic は、VPC への移行をお勧めします。詳細については、Amazon EC2 EC2-Classic から VPC への移行Amazon EC2」を参照してください。ブログ記事「EC2-Classic Networking は販売終了になります — 準備方法はこちら」も参照してください。

次のコマンドは、単一の `t1.micro` インスタンスを作成して起動します。

```
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       : {}
```

最初、インスタンスは pending 状態ですが、数分後には running 状態になります。インスタンスに関する情報を表示するには、Get-EC2Instance コマンドレットを使用します。複数のインスタンスがある場合は、Filter パラメータを使用して予約 ID の結果をフィルタします。まず、Amazon.EC2.Model.Filter 型のオブジェクトを作成します。次に、フィルターを使用する Get-EC2Instance を呼び出し、Instances プロパティを表示します。

```
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                 :
```

VPC でのインスタンスの起動

次のコマンドでは、指定したプライベートサブネットで、単一の m1.small インスタンスを作成しています。セキュリティグループは、指定したサブネットに対して有効である必要があります。

```
PS > New-EC2Instance `
    -ImageId ami-c49c0dac `
    -MinCount 1 -MaxCount 1 `
    -KeyName myPSKeyPair `
    -SecurityGroupId sg-5d293231 `
    -InstanceType m1.small `
    -SubnetId subnet-d60013bf

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

最初、インスタンスは pending 状態ですが、数分後には running 状態になります。インスタンスに関する情報を表示するには、Get-EC2Instance コマンドレットを使用します。複数のインスタンスがある場合は、Filter パラメータを使用して予約 ID の結果をフィルタします。まず、Amazon.EC2.Model.Filter 型のオブジェクトを作成します。次に、フィルターを使用する Get-EC2Instance を呼び出し、Instances プロパティを表示します。

```
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
```

VPC でのスポットインスタンスの起動

次のスクリプト例は、指定されたサブネットのスポットインスタンスをリクエストします。セキュリティグループは、指定したサブネットが含まれている VPC 用に作成したものである必要があります。

```
$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 および AWS Tools for PowerShell

AWSLambdaPSCore モジュールを使用すると、.NET Core 2.1 ランタイムを使用して PowerShell Core 6.0 の AWS Lambda 関数を開発できます。PowerShell 開発者は、Lambda を使用して PowerShell 環境でAWS リソースを管理し、自動化スクリプトを記述できます。Lambda の PowerShell サポートにより、任意の Lambda イベント (Amazon S3 イベントやスケジュールされた Amazon CloudWatch イベントなど) に応じて PowerShell スクリプトまたは関数を実行できます。AWSLambdaPSCore モジュールは PowerShell の独立した AWS モジュールです。AWS Tools for PowerShell の一部ではありません。また、AWSLambdaPSCore モジュールをインストールしても、AWS Tools for PowerShell はインストールされません。

AWSLambdapScore モジュールのインストール後、使用可能な PowerShell コマンドレットを使用するか、独自のコマンドレットを開発して、サーバーレス関数を作成することができます。AWS Lambda Tools for PowerShell モジュールには、PowerShell ベースのサーバーレスアプリケーション用のプロジェクトテンプレートと、プロジェクトを AWS に発行するためのツールが含まれています。

AWSLambdaPSCore モジュールのサポートは、Lambda をサポートするすべてのリージョンで使用できます。サポートされるリージョンの詳細については、AWS リージョン表を参照してください。

前提条件

AWSLambdaPSCore モジュールをインストールして使用する前に、以下のステップを実行する必要があります。これらの手順の詳細については、AWS Lambda デベロッパーガイドの「PowerShell 開発環境のセットアップ」を参照してください。

- PowerShell の適切なリリースのインストール – Lambda での PowerShell のサポートは、クロスプラットフォームの PowerShell Core 6.0 リリースに基づいています。PowerShell Lambda 関数

は、Windows、Linux、または Mac で作成できます。少なくともこのリリースの PowerShell をインストールしていない場合は、Microsoft PowerShell ドキュメントのウェブサイトでインストール手順を参照してください。

- .NET Core 2.1 SDK のインストール – PowerShell Core は .NET Core に基づいているため、Lambda での PowerShell のサポートでは同じ .NET Core 2.1 Lambda ランタイムを .NET Core と PowerShell Lambda 関数の両方に使用します。Lambda PowerShell でコマンドレットを発行する際は、.NET Core 2.1 SDK を使用して Lambda デプロイパッケージを作成します。.NET Core 2.1 SDK は Microsoft ダウンロードセンターから入手できます。ランタイムではなく、SDK を必ずインストールしてください。

AWSLambdaPSCore モジュールのインストール

前提条件を満たすと、AWSLambdaPSCore モジュールをインストールする準備が整います。PowerShell Core セッションで次のコマンドを実行します。

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

これで PowerShell での Lambda 関数の作成を開始できます。開始方法の詳細については、AWS Lambda デベロッパーガイドの「PowerShell での Lambda 関数の作成用プログラミングモデル」を参照してください。

以下も参照してください。

- AWS デベロッパーブログの「Lambda での PowerShell Core のサポートの発表」
- PowerShell Gallery ウェブサイトの AWSLambdaPSCore モジュール
- PowerShell 開発環境の設定
- GitHub の AWS Lambda Tools for PowerShell
- AWS Lambda コンソール

Amazon SQS、Amazon SNS、および Tools for Windows PowerShell

このセクションでは、次の方法の例を示します。

- Amazon SQS キューを作成し、キュー ARN (Amazon リソースネーム) を取得します。

- Amazon SNS トピックを作成する。
- SNS トピックにアクセス許可を付与して、キューにメッセージを送信できるようにします。
- キューの SNS トピックへのサブスクライブを行います。
- IAM ユーザーまたは AWS アカウントに、SNS トピックに発行し、SQS キューからのメッセージを読むためのアクセス許可を与えます。
- トピックにメッセージを発行し、キューからのメッセージを読むことで結果を確認します。

Amazon SQS キューの作成およびキュー ARN の取得

次のコマンドは、デフォルトのリージョンに SQS キューを作成します。出力には、新しいキューの URL が表示されます。

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

次のコマンドは、キューの ARN を取得します。

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

Amazon SNS トピックを作成する

次のコマンドは、デフォルトリージョンに SNS トピックを作成し、新しいトピックの ARN を返します。

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

SNS トピックへのアクセス許可の付与

次のスクリプト例は、SQS キューと SNS トピックの両方を作成し、SQS キューにメッセージを送信できるように SNS トピックにアクセス許可を付与します。

```
# 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 }
```

キューの SNS トピックへのサブスクライブを行います。

次のコマンドは、SNS トピック myQueue にキュー myTopic をサブスクライブし、サブスクリプション ID を返します。

```
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
```

アクセス許可の付与

次のコマンドでは、トピック sns:Publish に対して myTopic アクションを実行するためのアクセス許可を付与します。

```
PS > Add-SNSPermission `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
```

```
    -Label ps-cmdlet-topic `
    -AWSAccountIds 123456789012 `
    -ActionNames publish
```

次のコマンドでは、キュー sqs:ReceiveMessage の sqs:DeleteMessage および myQueue アクションを実行するためのアクセス許可を付与します。

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

結果の確認

次のコマンドは、SNS トピック myTopic にメッセージを発行して、新しいキューとトピックをテストし、MessageId を返します。

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

次のコマンドは、SQS キュー myQueue からメッセージを取得し、表示します。

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

Attributes             : {}
Body                   : {
                             "Type" : "Notification",
                             "MessageId" : "491c687d-b78d-5c48-b7a0-3d8d769ee91b",
                             "TopicArn" : "arn:aws:sns:us-west-2:123456789012:myTopic",
                             "Message" : "Have A Nice Day!",
                             "Timestamp" : "2019-09-09T21:06:27.201Z",
                             "SignatureVersion" : "1",
                             "Signature" :
 "llE17A2+XOuJZnw3TlgcXz4C4KPLXZxbxoEMIirelhl3u/oxkWmz5+9tJKFMns1ZOqQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1bO3hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
                         vyuGkJReV7lOJWPHnCN
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yo
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
```

```
                          kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvr3WbaSvg==",
                            "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-6aad65c2f9911b05cd53efda11f913f9.pem",
                            "UnsubscribeURL" :
                          "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:myTopic:22b77de7-
a216-4000-9a23-bf465744ca84"
                          }
MD5OfBody              : 5b5ee4f073e9c618eda3718b594fa257
MD5OfMessageAttributes :
MessageAttributes      : {}
MessageId              : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
ReceiptHandle          :
 AQEB2vvk1e5cOKFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJbl7F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHnl8SlbQa0R/kD+Saqa4OIvfj8x3M4Oh1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
                          +HmXdkax2Wd+9AxrHlQZV5ur1MoByKWWbDbsqoYJTJquCclOgWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
                          sHN12776axknhg3j9K/Xwj54DixdsegnrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2MseyOeM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==
```

AWS Tools for Windows PowerShell からのCloudWatch

このセクションでは、Tools for Windows PowerShell を使用してカスタムメトリクスデータを CloudWatch に発行する方法を例を挙げて示します。

この例では、ユーザーの PowerShell セッションのデフォルトの認証情報とデフォルトのリージョンが設定済みであることを前提としています。

カスタムメトリクスの CloudWatch ダッシュボードへの発行

次の PowerShell のコードでは、CloudWatch `MetricDatum` オブジェクトを初期化して、サービスに送信しています。このオペレーションの結果は、CloudWatch コンソールで確認できます。

```
$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
```

次の点に注意してください。

- $dat.Timestamp を初期化するために使用する日時情報は、協定世界時 (UTC) 形式で指定する必要があります。
- $dat.Value を初期化するために使用する値は、引用符で囲まれた文字列値または数値 (引用符なし) のどちらかで指定する必要があります。この例は、文字列値を示しています。

以下の資料も参照してください。

- AWS Tools for PowerShell での AWS サービスの操作
- AmazonCloudWatchClient.PutMetricData (.NET SDK リファレンス)
- MetricDatum (サービス API リファレンス)
- Amazon CloudWatch コンソール

コマンドレットでの ClientConfig パラメータの使用

ClientConfig パラメータを使用すると、サービスに接続するときに、特定の構成設定を指定できます。このパラメータに指定できるプロパティのほとんどは、AWS のサービスの API に継承された Amazon.Runtime.ClientConfig クラスで定義されています。単純な継承の例については、Amazon.Keyspaces.AmazonKeyspacesConfig クラスを参照してください。さらに、一部のサービスでは、そのサービスにのみ適切な追加プロパティが定義されています。定義されているその他のプロパティの例については、Amazon.S3.AmazonS3Config クラス、特に ForcePathStyle プロパティを参照してください。

ClientConfig パラメータの使用

ClientConfig パラメータを使用するには、コマンドラインで ClientConfig オブジェクトとして指定するか、PowerShell スプラッティングを使用してパラメータ値のコレクションをコマンドに単位として渡します。次の例に、これらの方法を示します。この例では、AWS.Tools.S3 モジュールがインストールおよびインポートされ、適切なアクセス許可を持つ [default] 認証情報プロファイルがあることを前提としています。

ClientConfig オブジェクトの定義

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

PowerShell スプラッティングの使用による **ClientConfig** プロパティの追加

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

Get-S3Object @params
```

未定義プロパティの使用

PowerShell スプラッティングを使用するときに、存在しない ClientConfig プロパティを指定した場合、AWS Tools for PowerShell は実行時まで エラーを検出せず、実行時には例外を返します。上記の例を修正すると、次のようになります。

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

Get-S3Object @params
```

この例では、以下のような例外が生成されます。

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

AWS リージョン の指定

ClientConfig パラメータを使用して、コマンドに対する AWS リージョンを設定します。[リージョン] は RegionEndpoint プロパティを通じて設定されます。AWS Tools for PowerShell は、使用する [リージョン] を次の優先順位に従って計算します。

1. `-Region` パラメータ
2. `ClientConfig` パラメータで渡された [リージョン]
3. PowerShell のセッション状態
4. 共有 AWS `config` ファイル
5. 環境変数
6. Amazon EC2 インスタンスメタデータ (有効になっている場合)。

PowerShell コード例のツール

このトピックのコード例は、 AWS Tools for PowerShell で を使用する方法を示しています AWS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

例

- Tools for を使用したアクションとシナリオ PowerShell

Tools for を使用したアクションとシナリオ PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS のサービス。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

サービス

- Tools for を使用した ACM の例 PowerShell
- AppStream Tools for を使用した 2.0 の例 PowerShell
- Tools for を使用した Aurora の例 PowerShell
- Tools for を使用した Auto Scaling の例 PowerShell
- AWS Budgets Tools for を使用した の例 PowerShell
- AWS Cloud9 Tools for を使用した の例 PowerShell

- AWS CloudFormation Tools for を使用した の例 PowerShell
- CloudFront Tools for を使用した の例 PowerShell
- CloudTrail Tools for を使用した の例 PowerShell
- CloudWatch Tools for を使用した の例 PowerShell
- CodeCommit Tools for を使用した の例 PowerShell
- CodeDeploy Tools for を使用した の例 PowerShell
- CodePipeline Tools for を使用した の例 PowerShell
- Tools for を使用した Amazon Cognito ID の例 PowerShell
- AWS Config Tools for を使用した の例 PowerShell
- Tools for を使用した Device Farm の例 PowerShell
- AWS Directory Service Tools for を使用した の例 PowerShell
- AWS DMS Tools for を使用した の例 PowerShell
- Tools for を使用した DynamoDB の例 PowerShell
- Tools for を使用した Amazon EC2 の例 PowerShell
- Tools for を使用した Amazon ECR の例 PowerShell
- Tools for を使用した Amazon ECS の例 PowerShell
- Tools for を使用した Amazon EFS の例 PowerShell
- Tools for を使用した Amazon EKS の例 PowerShell
- Tools for を使用した Elastic Load Balancing - バージョン 1 の例 PowerShell
- Elastic Load Balancing - Tools for を使用したバージョン 2 の例 PowerShell
- Tools for を使用した Amazon FSx の例 PowerShell
- AWS Glue Tools for を使用した の例 PowerShell
- AWS Health Tools for を使用した の例 PowerShell
- Tools for を使用した IAM の例 PowerShell
- Tools for を使用した Kinesis の例 PowerShell
- Tools for を使用した Lambda の例 PowerShell
- Tools for を使用した Amazon ML の例 PowerShell
- Tools for を使用した Macie の例 PowerShell

- AWS OpsWorks Tools for を使用した の例 PowerShell
- AWS の料金表 Tools for を使用した の例 PowerShell
- Tools for を使用した Resource Groups の例 PowerShell
- Tools for を使用した Resource Groups Tagging API の例 PowerShell
- Tools for を使用した Route 53 の例 PowerShell
- Tools for を使用した Amazon S3 の例 PowerShell
- Tools for を使用した S3 Glacier の例 PowerShell
- Tools for を使用した Amazon SES の例 PowerShell
- Tools for を使用した Amazon SNS の例 PowerShell
- Tools for を使用した Amazon SQS の例 PowerShell
- AWS STS Tools for を使用した の例 PowerShell
- AWS Support Tools for を使用した の例 PowerShell
- Tools for を使用した Systems Manager の例 PowerShell
- Tools for を使用した Amazon Translate の例 PowerShell
- AWS WAFV2 Tools for を使用した の例 PowerShell
- WorkSpaces Tools for を使用した の例 PowerShell

Tools for を使用した ACM の例 PowerShell

次のコード例は、ACM AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Get-ACMCertificate

次の例は、Get-ACMCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、証明書の ARN を使用して証明書とそのチェーンを返す方法を示します。

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

- API の詳細については、「 コマンドレットリファレンスGetCertificate」の「」を参照してください。 AWS Tools for PowerShell

Get-ACMCertificateDetail

次の例は、Get-ACMCertificateDetail を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された証明書の詳細を返します。

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

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスDescribeCertificate」の「」を参照してください。 AWS Tools for PowerShell

Get-ACMCertificateList

次の例は、Get-ACMCertificateList を使用する方法を説明しています。

のツール PowerShell

例 1: すべての証明書 ARNsとそれぞれのドメイン名のリストを取得します。コマンドレットは自動的にページ分割され、すべての ARNs を取得します。ページ分割を手動で制御するには、MaxItem パラメータを使用して各サービスコールに対して返される証明書 ARNs の数を制御し、NextToken パラメータを使用して各コールの開始点を示します。

```
Get-ACMCertificateList
```

出力:

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

例 2: 指定された状態で証明書ステータスが一致するすべての証明書 ARNs のリストを取得します。

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

例 3: この例では、キータイプが RSA_2048、拡張キーの使用法または目的が CODE_SIGNING である us-east-1 リージョン内のすべての証明書のリストを返します。これらのフィルタリングパラメータの値は、「フィルター API ListCertificates リファレンス」トピックで確認できます。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
```

出力:

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

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

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

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

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

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

- API の詳細については、「 コマンドレットリファレンスListCertificates」の「」を参照してください。 AWS Tools for PowerShell

New-ACMCertificate

次の例は、New-ACMCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい証明書を作成します。サービスは新しい証明書の ARN を返します。

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

出力:

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

例 2: 新しい証明書を作成します。サービスは新しい証明書の ARN を返します。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスRequestCertificate」の「」を参照してください。 AWS Tools for PowerShell

Remove-ACMCertificate

次の例は、Remove-ACMCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された ARN および関連するプライベートキーによって識別される証明書を削除します。コマンドレットは続行する前に確認を求めます。確認を抑制するには -Force スイッチを追加します。

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

- API の詳細については、「 コマンドレットリファレンスDeleteCertificate」の「」を参照してください。 AWS Tools for PowerShell

Send-ACMValidationEmail

次の例は、Send-ACMValidationEmail を使用する方法を説明しています。

のツール PowerShell

例 1: www.example.com」のドメインの所有権を検証する E メールの送信をリクエストします。シェルの $ ConfirmPreference が「Medium」以下に設定されている場合、コマンドレットは先に進む前に確認を求めます。-Force スイッチを追加して、確認プロンプトを非表示にします。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスResendValidationEmail」の「」を参照してください。 AWS Tools for PowerShell

AppStream Tools for を使用した 2.0 の例 PowerShell

次のコード例は、 AppStream 2.0 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-APSResourceTag`

次の例は、Add-APSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、リソースタグを AppStream リソースに追加します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスTagResource」の「」を参照してください。 AWS Tools for PowerShell

Copy-APSImage

次の例は、Copy-APSImage を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、イメージを他のリージョンにコピーします。

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

出力:

```
TestImageCopy
```

- API の詳細については、「 コマンドレットリファレンスCopyImage」の「」を参照してください。 AWS Tools for PowerShell

Disable-APSUSer

次の例は、Disable-APSUSer を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは USERPOOL でユーザーを無効にします

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

- API の詳細については、「 コマンドレットリファレンスDisableUser」の「」を参照してください。 AWS Tools for PowerShell

Enable-APSUser

次の例は、Enable-APSUser を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、USERPOOL で無効化されたユーザーを有効にします

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

- API の詳細については、「 コマンドレットリファレンスEnableUser」の「」を参照してください。 AWS Tools for PowerShell

Get-APSAssociatedFleetList

次の例は、Get-APSAssociatedFleetList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、スタックに関連付けられたフリートを表示します

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

出力:

```
PowershellFleet
```

- API の詳細については、「 コマンドレットリファレンスListAssociatedFleets」の「」を参照してください。 AWS Tools for PowerShell

Get-APSAssociatedStackList

次の例は、Get-APSAssociatedStackList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、フリートに関連付けられたスタックを表示します

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

出力:

```
PowershellStack
```

- API の詳細については、「 コマンドレットリファレンスListAssociatedStacks」の「」を参照してください。 AWS Tools for PowerShell

Get-APSDirectoryConfigList

次の例は、Get-APSDirectoryConfigList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 で作成されたディレクトリ設定を表示します。 AppStream

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeDirectoryConfigs」の「」を参照してください。 AWS Tools for PowerShell

Get-APSFleetList

次の例は、Get-APSFleetList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはフリートの詳細を表示します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeFleets」の「」を参照してください。 AWS Tools for PowerShell

Get-APSImageBuilderList

次の例は、Get-APSImageBuilderList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 の詳細を表示します。 ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeImageBuilders」の「」を参照してください。 AWS Tools for PowerShell

Get-APSImageList

次の例は、Get-APSImageList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはプライベート AppStream イメージを表示します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeImages」の「」を参照してください。 AWS Tools for PowerShell

Get-APSImagePermission

次の例は、Get-APSImagePermission を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、共有イメージに対する AppStream イメージのアクセス許可を表示します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeImagePermissions」の「」を参照してください。 AWS Tools for PowerShell

Get-APSSessionList

次の例は、Get-APSSessionList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、フリートへのセッションのリストを表示します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeSessions」の「」を参照してください。 AWS Tools for PowerShell

Get-APSStackList

次の例は、Get-APSStackList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは AppStream スタックのリストを表示します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeStacks」の「」を参照してください。 AWS Tools for PowerShell

Get-APSTagsForResourceList

次の例は、Get-APSTagsForResourceList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、 AppStream リソースのタグを表示します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

Get-APSUsageReportSubscription

次の例は、Get-APSUsageReportSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは AppStreamUsageReport 設定の詳細を表示します

```
Get-APSUsageReportSubscription
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeUsageReportSubscriptions」の「」を参照してください。 AWS Tools for PowerShell

Get-APSUser

次の例は、Get-APSUser を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、ステータスが有効なユーザーのリストを表示します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeUsers」の「」を参照してください。 AWS Tools for PowerShell

Get-APSUserStackAssociation

次の例は、Get-APSUserStackAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、スタックに割り当てられたユーザーのリストを表示します。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeUserStackAssociations」の「」を参照してください。 AWS Tools for PowerShell

New-APSDirectoryConfig

次の例は、New-APSDirectoryConfig を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 にディレクトリ設定を作成します。 AppStream

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateDirectoryConfig」の「」を参照してください。 AWS Tools for PowerShell

New-APSFleet

次の例は、New-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは新しい AppStream フリートを作成します

```
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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateFleet」の「」を参照してください。 AWS Tools for PowerShell

New-APSImageBuilder

次の例は、New-APSImageBuilder を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 で Image Builder を作成します。 AppStream

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateImageBuilder」の「」を参照してください。 AWS Tools for PowerShell

New-APSImageBuilderStreamingURL

次の例は、New-APSImageBuilderStreamingURL を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、有効期間が 2 時間の ImageBuilder ストリーミング URL を作成します。

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

出力:

```
Expires              StreamingURL
-------              ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- API の詳細については、「 コマンドレットリファレンス」のCreateImageBuilderStreaming「URL」を参照してください。 AWS Tools for PowerShell

New-APSStack

次の例は、New-APSStack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは新しい AppStream スタックを作成します

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

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスCreateStack」の「」を参照してください。 AWS Tools for PowerShell

New-APSStreamingURL

次の例は、New-APSStreamingURL を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、 スタックのストリーミング URL を作成します。

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

出力:

```
Expires                StreamingURL
-------                ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- API の詳細については、「 コマンドレットリファレンス」のCreateStreaming「URL」を参照してください。 AWS Tools for PowerShell

New-APSUsageReportSubscription

次の例は、New-APSUsageReportSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは AppStream 使用状況レポートを有効にします

```
New-APSUsageReportSubscription
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスCreateUsageReportSubscription」の「」を参照してください。 AWS Tools for PowerShell

New-APSUser

次の例は、New-APSUser を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは USERPOOL でユーザーを作成します

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

出力:

```
Test@lab.com
```

- API の詳細については、「 コマンドレットリファレンスCreateUser」の「」を参照してください。 AWS Tools for PowerShell

Register-APSFleet

次の例は、Register-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、フリートをスタックに登録します

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

- API の詳細については、「 コマンドレットリファレンスAssociateFleet」の「」を参照してください。 AWS Tools for PowerShell

Register-APSUserStackBatch

次の例は、Register-APSUserStackBatch を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、USERPOOL のユーザーにスタックを割り当てます。

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

- API の詳細については、「 コマンドレットリファレンスBatchAssociateUserStack」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSDirectoryConfig

次の例は、Remove-APSDirectoryConfig を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは AppStream ディレクトリ設定を削除します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteDirectoryConfig」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSFleet

次の例は、Remove-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはフリー AppStream トを削除します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteFleet」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSImage

次の例は、Remove-APSImage を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはイメージを削除します

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

出力:

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

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

- API の詳細については、「 コマンドレットリファレンスDeleteImage」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSImageBuilder

次の例は、Remove-APSImageBuilder を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは を削除します ImageBuilder

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

出力:

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

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

- API の詳細については、「 コマンドレットリファレンスDeleteImageBuilder」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSImagePermission

次の例は、Remove-APSImagePermission を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、イメージのアクセス許可を削除します

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDeleteImagePermissions」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSResourceTag

次の例は、Remove-APSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、リソースから AppStream リソースタグを削除します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUntagResource」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSStack

次の例は、Remove-APSStack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは スタックを削除します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDeleteStack」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSUsageReportSubscription

次の例は、Remove-APSUsageReportSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは AppStream 使用状況レポートのサブスクリプションを無効にします

```
Remove-APSUsageReportSubscription
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteUsageReportSubscription」の「」を参照してください。 AWS Tools for PowerShell

Remove-APSUser

次の例は、Remove-APSUser を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは USERPOOL からユーザーを削除します

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDeleteUser」の「」を参照してください。 AWS Tools for PowerShell

Revoke-APSSession

次の例は、Revoke-APSSession を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 AppStream フリートへのセッションを取り消します

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

- API の詳細については、「 コマンドレットリファレンスExpireSession」の「」を参照してください。 AWS Tools for PowerShell

Start-APSFleet

次の例は、Start-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはフリートを開始します

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

- API の詳細については、「 コマンドレットリファレンスStartFleet」の「」を参照してください。 AWS Tools for PowerShell

`Start-APSImageBuilder`

次の例は、`Start-APSImageBuilder` を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは を開始します ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスStartImageBuilder」の「」を参照してください。 AWS Tools for PowerShell

Stop-APSFleet

次の例は、Stop-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはフリートを停止する

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

- API の詳細については、「 コマンドレットリファレンスStopFleet」の「」を参照してください。 AWS Tools for PowerShell

Stop-APSImageBuilder

次の例は、Stop-APSImageBuilder を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは を停止する ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスStopImageBuilder」の「」を参照してください。 AWS Tools for PowerShell

Unregister-APSFleet

次の例は、Unregister-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、スタックからフリートを登録解除します

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

- API の詳細については、「 コマンドレットリファレンスDisassociateFleet」の「」を参照してください。 AWS Tools for PowerShell

Unregister-APSUserStackBatch

次の例は、Unregister-APSUserStackBatch を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、割り当てられた スタックからユーザーを削除します

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

- API の詳細については、「 コマンドレットリファレンスBatchDisassociateUserStack」の「」を参照してください。 AWS Tools for PowerShell

Update-APSDirectoryConfig

次の例は、Update-APSDirectoryConfig を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 で作成されたディレクトリ設定を更新します。 AppStream

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateDirectoryConfig」の「」を参照してください。 AWS Tools for PowerShell

Update-APSFleet

次の例は、Update-APSFleet を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルはフリートのプロパティを更新します

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

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateFleet」の「」を参照してください。 AWS Tools for PowerShell

Update-APSImagePermission

次の例は、Update-APSImagePermission を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 AppStream イメージを他のアカウントと共有します

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

- API の詳細については、「 コマンドレットリファレンスUpdateImagePermissions」の「」を参照してください。 AWS Tools for PowerShell

Update-APSStack

次の例は、Update-APSStack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、スタック上のアプリケーション設定の永続化とホームフォルダを更新 (有効化) します。

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

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスUpdateStack」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Aurora の例 PowerShell

次のコード例は、Aurora AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-RDSOrderableDBInstanceOption`

次の例は、Get-RDSOrderableDBInstanceOption を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、 AWS リージョン内で特定の DB インスタンスクラスをサポートする DB エンジンのバージョンを一覧表示します。

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

例 2: この例は、 AWS リージョン内で特定の DB エンジンのバージョンをサポートする DB インスタンスクラスを一覧表示します。

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

- API の詳細については、「 コマンドレットリファレンス」のDescribeOrderable「DBInstanceOptions」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Auto Scaling の例 PowerShell

次のコード例は、Auto Scaling AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-ASLoadBalancer`

次の例は、`Add-ASLoadBalancer` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーを指定された Auto Scaling グループにアタッチします。

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

- API の詳細については、「 コマンドレットリファレンスAttachLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

`Complete-ASLifecycleAction`

次の例は、`Complete-ASLifecycleAction` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたライフサイクルアクションを完了します。

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

- API の詳細については、「 コマンドレットリファレンスCompleteLifecycleAction」の「」を参照してください。 AWS Tools for PowerShell

Disable-ASMetricsCollection

次の例は、Disable-ASMetricsCollection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたメトリクスのモニタリングを無効にします。

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

例 2: この例では、指定された Auto Scaling グループのすべてのメトリクスのモニタリングを無効にします。

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

- API の詳細については、「 コマンドレットリファレンスDisableMetricsCollection」の「」を参照してください。 AWS Tools for PowerShell

Dismount-ASInstance

次の例は、Dismount-ASInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを指定された Auto Scaling グループからデタッチし、Auto Scaling が代替インスタンスを起動しないように、必要な容量を減らします。

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

出力:

```
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        :
```

例 2: この例では、希望する容量を減らすことなく、指定された Auto Scaling グループから指定されたインスタンスをデタッチします。Auto Scaling は代替インスタンスを起動します。

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

出力:

```
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        :
```

- API の詳細については、「 コマンドレットリファレンスDetachInstances」の「」を参照してください。 AWS Tools for PowerShell

`Dismount-ASLoadBalancer`

次の例は、`Dismount-ASLoadBalancer` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループから指定されたロードバランサーをデタッチします。

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

- API の詳細については、「 コマンドレットリファレンスDetachLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

Enable-ASMetricsCollection

次の例は、Enable-ASMetricsCollection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたメトリクスのモニタリングを有効にします。

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

例 2: この例では、指定された Auto Scaling グループのすべてのメトリクスのモニタリングを有効にします。

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

- API の詳細については、「 コマンドレットリファレンスEnableMetricsCollection」の「」を参照してください。 AWS Tools for PowerShell

Enter-ASStandby

次の例は、Enter-ASStandby を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスをスタンバイモードにして希望する容量を減らし、Auto Scaling が代替インスタンスを起動しないようにします。

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

出力:

```
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        :
```

例 2: この例では、希望する容量を減らすことなく、指定されたインスタンスをスタンバイモードにします。Auto Scaling は代替インスタンスを起動します。

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

出力:

```
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        :
```

- API の詳細については、「 コマンドレットリファレンスEnterStandby」の「」を参照してください。 AWS Tools for PowerShell

Exit-ASStandby

次の例は、Exit-ASStandby を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスをスタンバイモードから移行します。

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

出力:

```
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        :
```

- API の詳細については、「 コマンドレットリファレンスExitStandby」の「」を参照してください。 AWS Tools for PowerShell

Get-ASAccountLimit

次の例は、Get-ASAccountLimit を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウントの Auto Scaling リソース制限について説明します。

```
Get-ASAccountLimit
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeAccountLimits」の「」を参照してください。 AWS Tools for PowerShell

Get-ASAdjustmentType

次の例は、Get-ASAdjustmentType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている調整タイプについて説明します。

```
Get-ASAdjustmentType
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeAdjustmentTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-ASAutoScalingGroup

次の例は、Get-ASAutoScalingGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling グループの名前を一覧表示します。

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

出力:

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

例 2: この例では、指定された Auto Scaling グループについて説明します。

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

出力:

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

例 3: この例では、指定された 2 つの Auto Scaling グループについて説明します。

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

例 4: この例では、指定された Auto Scaling グループの Auto Scaling インスタンスについて説明します。

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

例 5: この例では、すべての Auto Scaling グループについて説明します。

```
Get-ASAutoScalingGroup
```

例 6: この例では、すべての Auto Scaling グループを 10 個のバッチで記述します。

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

例 7: この例では LaunchTemplate 、指定された Auto Scaling グループの について説明します。この例では、「インスタンスの購入オプション」が「起動テンプレートに準拠する」に設定されていることを前提としています。このオプションが「購入オプションとインスタンスタイプの組み合わせ」に設定されている場合、 LaunchTemplate 「.MixedInstancesPolicyLaunchTemplate」プロパティを使用してアクセスできます。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeAutoScalingGroups」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASAutoScalingInstance`

次の例は、`Get-ASAutoScalingInstance` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling インスタンスIDs を一覧表示します。

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

出力:

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

例 2: この例では、指定された Auto Scaling インスタンスについて説明します。

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

出力:

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

例 3: この例では、指定された 2 つの Auto Scaling インスタンスについて説明します。

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

例 4: この例では、指定された Auto Scaling グループの Auto Scaling インスタンスについて説明します。

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

例 5: この例では、すべての Auto Scaling インスタンスについて説明します。

```
Get-ASAutoScalingInstance
```

例 6: この例では、すべての Auto Scaling インスタンスを 10 のバッチで記述します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeAutoScalingInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-ASAutoScalingNotificationType

次の例は、Get-ASAutoScalingNotificationType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている通知タイプを一覧表示します。

```
Get-ASAutoScalingNotificationType
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeAutoScalingNotificationTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-ASLaunchConfiguration

次の例は、Get-ASLaunchConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、起動設定の名前を一覧表示します。

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

出力:

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

例 2: この例では、指定された起動設定について説明します。

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

出力:

```
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                    :
```

例 3: この例では、指定された 2 つの起動設定について説明します。

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

例 4: この例では、すべての起動設定について説明します。

```
Get-ASLaunchConfiguration
```

例 5: この例では、すべての起動設定を 10 個のバッチで説明します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeLaunchConfigurations」の「」を参照してください。 AWS Tools for PowerShell

Get-ASLifecycleHook

次の例は、Get-ASLifecycleHook を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたライフサイクルフックについて説明します。

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

出力:

```
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
```

例 2: この例では、指定された Auto Scaling グループのすべてのライフサイクルフックについて説明します。

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

例 3: この例では、すべての Auto Scaling グループのすべてのライフサイクルフックについて説明します。

```
Get-ASLifecycleHook
```

- API の詳細については、「 コマンドレットリファレンスDescribeLifecycleHooks」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASLifecycleHookType`

次の例は、`Get-ASLifecycleHookType` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされているライフサイクルフックタイプを一覧表示します。

```
Get-ASLifecycleHookType
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeLifecycleHookTypes」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASLoadBalancer`

次の例は、`Get-ASLoadBalancer` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループのロードバランサーについて説明します。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

Get-ASMetricCollectionType

次の例は、Get-ASMetricCollectionType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされているメトリクスコレクションタイプを一覧表示します。

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

出力:

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

例 2: この例では、対応する詳細度を一覧表示します。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeMetricCollectionTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-ASNotificationConfiguration

次の例は、Get-ASNotificationConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループに関連付けられた通知アクションについて説明します。

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

出力:

```
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
```

例 2: この例では、すべての Auto Scaling グループに関連付けられている通知アクションについて説明します。

```
Get-ASNotificationConfiguration
```

- API の詳細については、「 コマンドレットリファレンスDescribeNotificationConfigurations」の「」を参照してください。 AWS Tools for PowerShell

Get-ASPolicy

次の例は、Get-ASPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループのすべてのポリシーについて説明します。

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

出力:

```
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         : {}
```

例 2: この例では、指定された Auto Scaling グループの指定されたポリシーについて説明します。

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

例 3: この例では、すべての Auto Scaling グループのすべてのポリシーについて説明します。

```
Get-ASPolicy
```

- API の詳細については、「 コマンドレットリファレンスDescribePolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-ASScalingActivity

次の例は、Get-ASScalingActivity を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの過去 6 週間のスケーリングアクティビティについて説明します。

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

出力:

```
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        :
```

例 2: この例では、指定されたスケーリングアクティビティについて説明します。

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

例 3: この例では、すべての Auto Scaling グループの過去 6 週間のスケーリングアクティビティについて説明します。

```
Get-ASScalingActivity
```

- API の詳細については、「 コマンドレットリファレンスDescribeScalingActivities」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASScalingProcessType`

次の例は、`Get-ASScalingProcessType` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされているプロセスタイプを一覧表示します。

```
Get-ASScalingProcessType
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeScalingProcessTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-ASScheduledAction

次の例は、Get-ASScheduledAction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループに対してスケジュールされたスケーリングアクションについて説明します。

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

出力:

```
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
```

例 2: この例では、指定されたスケジュールされたスケーリングアクションについて説明します。

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

例 3: この例では、指定された時間までに開始されるスケジュールされたスケーリングアクションについて説明します。

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

例 4: この例では、指定した時刻までに終了するスケジュールされたスケーリングアクションについて説明します。

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

例 5: この例では、すべての Auto Scaling グループに対してスケジュールされたスケーリングアクションについて説明します。

```
Get-ASScheduledAction
```

- API の詳細については、「 コマンドレットリファレンスDescribeScheduledActions」の「」を参照してください。 AWS Tools for PowerShell

Get-ASTag

次の例は、Get-ASTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、キー値がmyTag」またはmyTag2」のタグについて説明します。フィルター名に使用できる値は、auto-scaling-group「」、「キー」、「値」、propagate-at-launch「」です。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

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

出力:

```
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
```

例 2: PowerShell バージョン 2 では、New-Object を使用して Filter パラメータのフィルターを作成する必要があります。

```
$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 )
```

例 3: この例では、すべての Auto Scaling グループのすべてのタグについて説明します。

```
Get-ASTag
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASTerminationPolicyType`

次の例は、`Get-ASTerminationPolicyType` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている終了ポリシーを一覧表示します。

```
Get-ASTerminationPolicyType
```

出力:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API の詳細については、「 コマンドレットリファレンスDescribeTerminationPolicyTypes」の「」を参照してください。 AWS Tools for PowerShell

`Mount-ASInstance`

次の例は、`Mount-ASInstance` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを指定された Auto Scaling グループにアタッチします。Auto Scaling は、Auto Scaling グループの希望する容量を自動的に増やします。

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

- API の詳細については、「 コマンドレットリファレンスAttachInstances」の「」を参照してください。 AWS Tools for PowerShell

New-ASAutoScalingGroup

次の例は、New-ASAutoScalingGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前と属性を持つ Auto Scaling グループを作成します。デフォルトの希望する容量は最小サイズです。したがって、この Auto Scaling グループは、指定された 2 つのアベイラビリティーゾーンのそれぞれに 1 つずつ、2 つのインスタンスを起動します。

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

- API の詳細については、「 コマンドレットリファレンスCreateAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

New-ASLaunchConfiguration

次の例は、New-ASLaunchConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「my-lc」という名前の起動設定を作成します。この起動設定を使用する Auto Scaling グループによって起動される EC2 インスタンスは、指定されたインスタンスタイプ、AMI、セキュリティグループ、および IAM ロールを使用します。

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

- API の詳細については、「 コマンドレットリファレンスCreateLaunchConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASAutoScalingGroup

次の例は、Remove-ASAutoScalingGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、実行中のインスタンスがない場合、指定された Auto Scaling グループを削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、オペレーションが続行する前に確認を求められません。

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

例 3: この例では、指定された Auto Scaling グループを削除し、そのグループに含まれる実行中のインスタンスをすべて終了します。

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

- API の詳細については、「 コマンドレットリファレンスDeleteAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASLaunchConfiguration

次の例は、Remove-ASLaunchConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Auto Scaling グループにアタッチされていない場合、指定された起動設定を削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、操作が進む前に確認を求められません。

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

- API の詳細については、「 コマンドレットリファレンスDeleteLaunchConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASLifecycleHook

次の例は、Remove-ASLifecycleHook を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたライフサイクルフックを削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、操作が進む前に確認を求められません。

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

- API の詳細については、「 コマンドレットリファレンスDeleteLifecycleHook」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASNotificationConfiguration

次の例は、Remove-ASNotificationConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された通知アクションを削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、オペレーションが続行する前に確認を求められません。

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

- API の詳細については、「 コマンドレットリファレンスDeleteNotificationConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASPolicy

次の例は、Remove-ASPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたポリシーを削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、オペレーションが続行する前に確認を求められません。

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

- API の詳細については、「 コマンドレットリファレンスDeletePolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASScheduledAction

次の例は、Remove-ASScheduledAction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたスケジュールされたアクションを削除します。操作を続行する前に確認画面が表示されます。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、操作が進む前に確認を求められません。

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

- API の詳細については、「 コマンドレットリファレンスDeleteScheduledAction」の「」を参照してください。 AWS Tools for PowerShell

Remove-ASTag

次の例は、Remove-ASTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループから指定されたタグを削除します。操作を続行する前に確認画面が表示されます。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

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

出力:

```
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"):
```

例 2: Force パラメータを指定した場合、オペレーションが進む前に確認を求められません。

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

例 3: Powershell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

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

- API の詳細については、「 コマンドレットリファレンスDeleteTags」の「」を参照してください。 AWS Tools for PowerShell

Resume-ASProcess

次の例は、Resume-ASProcess を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定された Auto Scaling プロセスを再開します。

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

例 2: この例では、指定された Auto Scaling グループの中断されたすべての Auto Scaling プロセスを再開します。

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

- API の詳細については、「 コマンドレットリファレンスResumeProcesses」の「」を参照してください。 AWS Tools for PowerShell

Set-ASDesiredCapacity

次の例は、Set-ASDesiredCapacity を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループのサイズを設定します。

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

例 2: この例では、指定された Auto Scaling グループのサイズを設定し、クールダウン期間が完了するのを待ってから、新しいサイズにスケーリングします。

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

- API の詳細については、「 コマンドレットリファレンスSetDesiredCapacity」の「」を参照してください。 AWS Tools for PowerShell

Set-ASInstanceHealth

次の例は、Set-ASInstanceHealth を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスのステータスを「異常」に設定し、サービスから外します。Auto Scaling はインスタンスを終了して置き換えます。

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

例 2: この例では、指定されたインスタンスのステータスを「正常」に設定し、稼働させ続けます。Auto Scaling グループのヘルスチェックの猶予期間は適用されません。

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

- API の詳細については、「 コマンドレットリファレンスSetInstanceHealth」の「」を参照してください。 AWS Tools for PowerShell

Set-ASInstanceProtection

次の例は、Set-ASInstanceProtection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスのインスタンス保護を有効にします。

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

例 2: この例では、指定したインスタンスのインスタンス保護を無効にします。

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

- API の詳細については、「 コマンドレットリファレンスSetInstanceProtection」の「」を参照してください。 AWS Tools for PowerShell

Set-ASTag

次の例は、Set-ASTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループに単一のタグを追加します。タグキーは「myTag」で、タグ値はmyTagValue「」です。Auto Scaling は、このタグを Auto Scaling グループによって起動された後続の EC2 インスタンスに伝播します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

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

例 2: PowerShell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateOrUpdateTags」の「」を参照してください。 AWS Tools for PowerShell

Start-ASPolicy

次の例は、Start-ASPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定されたポリシーを実行します。

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

例 2: この例では、クールダウン期間の完了を待ってから、指定された Auto Scaling グループの指定されたポリシーを実行します。

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

- API の詳細については、「 コマンドレットリファレンスExecutePolicy」の「」を参照してください。 AWS Tools for PowerShell

Stop-ASInstanceInAutoScalingGroup

次の例は、Stop-ASInstanceInAutoScalingGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを終了し、Auto Scaling グループに必要な容量を減らして、Auto Scaling が代替インスタンスを起動しないようにします。

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

出力:

```
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        :
```

例 2: この例では、Auto Scaling グループの希望する容量を減らすことなく、指定されたインスタンスを終了します。Auto Scaling は代替インスタンスを起動します。

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

出力:

```
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        :
```

- API の詳細については、「 コマンドレットリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Suspend-ASProcess

次の例は、Suspend-ASProcess を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの指定された Auto Scaling プロセスを一時停止します。

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

例 2: この例では、指定した Auto Scaling グループのすべての Auto Scaling プロセスを一時停止します。

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

- API の詳細については、「 コマンドレットリファレンスSuspendProcesses」の「」を参照してください。 AWS Tools for PowerShell

Update-ASAutoScalingGroup

次の例は、Update-ASAutoScalingGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Auto Scaling グループの最小サイズと最大サイズを更新します。

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

例 2: この例では、指定された Auto Scaling グループのデフォルトのクールダウン期間を更新します。

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

例 3: この例では、指定された Auto Scaling グループのアベイラビリティーゾーンを更新します。

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

例 4: この例では、指定された Auto Scaling グループを更新して Elastic Load Balancing ヘルスチェックを使用します。

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

- API の詳細については、「 コマンドレットリファレンスUpdateAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Write-ASLifecycleActionHeartbeat

次の例は、Write-ASLifecycleActionHeartbeat を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたライフサイクルアクションのハートビートを記録します。これにより、カスタムアクションが完了するまでインスタンスが保留状態になります。

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

- API の詳細については、「 コマンドレットリファレンスRecordLifecycleActionHeartbeat」の「」を参照してください。 AWS Tools for PowerShell

Write-ASLifecycleHook

次の例は、Write-ASLifecycleHook を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたライフサイクルフックを指定された Auto Scaling グループに追加します。

```
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"
```

- API の詳細については、「 コマンドレットリファレンスPutLifecycleHook」の「」を参照してください。 AWS Tools for PowerShell

Write-ASNotificationConfiguration

次の例は、Write-ASNotificationConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2 インスタンスの起動時に指定された SNS トピックに通知を送信するように、指定された Auto Scaling グループを設定します。

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

例 2: この例では、EC2 インスタンスを起動または終了するときに、指定された SNS トピックに通知を送信するように、指定された Auto Scaling グループを設定します。

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

- API の詳細については、「 コマンドレットリファレンスPutNotificationConfiguration」の「」を参照してください。 AWS Tools for PowerShell

`Write-ASScalingPolicy`

次の例は、`Write-ASScalingPolicy` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたポリシーを指定された Auto Scaling グループに追加します。指定された調整タイプによって、 ScalingAdjustment パラメータの解釈方法が決まります。ChangeInCapacity「」では、正の値は指定された数のインスタンスで容量を増やし、負の値は指定された数のインスタンスで容量を減らします。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスPutScalingPolicy」の「」を参照してください。 AWS Tools for PowerShell

Write-ASScheduledUpdateGroupAction

次の例は、Write-ASScheduledUpdateGroupAction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、1 回限りのスケジュールされたアクションを作成または更新して、指定した開始時刻に必要な容量を変更します。

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

- API の詳細については、「 コマンドレットリファレンスPutScheduledUpdateGroupAction」の「」を参照してください。 AWS Tools for PowerShell

AWS Budgets Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Budgets。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

New-BGTBudget

次の例は、New-BGTBudget を使用する方法を説明しています。

のツール PowerShell

例 1: E メール通知を使用して、指定された予算と時間の制約を持つ新しい予算を作成します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateBudget」の「」を参照してください。 AWS Tools for PowerShell

AWS Cloud9 Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Cloud9。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Get-C9EnvironmentData`

次の例は、Get-C9EnvironmentData を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境に関する情報を取得します。

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

出力:

```
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
```

例 2: この例では、指定された AWS Cloud9 開発環境のライフサイクルステータスに関する情報を取得します。

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

出力:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- API の詳細については、「 コマンドレットリファレンスDescribeEnvironments」の「」を参照してください。 AWS Tools for PowerShell

Get-C9EnvironmentList

次の例は、Get-C9EnvironmentList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能な AWS Cloud9 開発環境識別子のリストを取得します。

```
Get-C9EnvironmentList
```

出力:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- API の詳細については、「 コマンドレットリファレンスListEnvironments」の「」を参照してください。 AWS Tools for PowerShell

Get-C9EnvironmentMembershipList

次の例は、Get-C9EnvironmentMembershipList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境の環境メンバーに関する情報を取得します。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

出力:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

例 2: この例では、指定された AWS Cloud9 開発環境の所有者に関する情報を取得します。

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

出力:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

例 3: この例では、複数の AWS Cloud9 開発環境の指定された環境メンバーに関する情報を取得します。

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

出力:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
```

```
LastAccess     : 1/16/2018 11:21:24 PM
Permissions    : owner
UserArn        : arn:aws:iam::123456789012:user/MyDemoUser
UserId         : AIDAJ3LOROMOUXTBSU6EX
```

- API の詳細については、「 コマンドレットリファレンスDescribeEnvironmentMemberships」の「」を参照してください。 AWS Tools for PowerShell

Get-C9EnvironmentStatus

次の例は、Get-C9EnvironmentStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境のステータス情報を取得します。

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

出力:

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

- API の詳細については、「 コマンドレットリファレンスDescribeEnvironmentStatus」の「」を参照してください。 AWS Tools for PowerShell

New-C9EnvironmentEC2

次の例は、New-C9EnvironmentEC2 を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された設定で AWS Cloud9 開発環境を作成し、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起動して、インスタンスから環境に接続します。

```
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
```

出力:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API の詳細については、「 コマンドレットリファレンス」のCreateEnvironmentEc「2」を参照してください。 AWS Tools for PowerShell

New-C9EnvironmentMembership

次の例は、New-C9EnvironmentMembership を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された環境メンバーを指定された AWS Cloud9 開発環境に追加します。

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

出力:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API の詳細については、「 コマンドレットリファレンスCreateEnvironmentMembership」の「」を参照してください。 AWS Tools for PowerShell

Remove-C9Environment

次の例は、Remove-C9Environment を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境を削除します。Amazon EC2 インスタンスが環境に接続されている場合、 はインスタンスも終了します。

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API の詳細については、「 コマンドレットリファレンスDeleteEnvironment」の「」を参照してください。 AWS Tools for PowerShell

Remove-C9EnvironmentMembership

次の例は、Remove-C9EnvironmentMembership を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境から指定された環境メンバーを削除します。

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API の詳細については、「 コマンドレットリファレンスDeleteEnvironmentMembership」の「」を参照してください。 AWS Tools for PowerShell

Update-C9Environment

次の例は、Update-C9Environment を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された既存の AWS Cloud9 開発環境の指定された設定を変更します。

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- API の詳細については、「 コマンドレットリファレンスUpdateEnvironment」の「」を参照してください。 AWS Tools for PowerShell

Update-C9EnvironmentMembership

次の例は、Update-C9EnvironmentMembership を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Cloud9 開発環境の指定された既存の環境メンバーの設定を変更します。

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

出力:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API の詳細については、「 コマンドレットリファレンスUpdateEnvironmentMembership」の「」を参照してください。 AWS Tools for PowerShell

AWS CloudFormation Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS CloudFormation。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Get-CFNStack`

次の例は、`Get-CFNStack` を使用する方法を説明しています。

のツール PowerShell

例 1: ユーザーのスタックのすべてを記述するスタックインスタンスのコレクションを返します。

```
Get-CFNStack
```

例 2: 指定されたスタックを記述するスタックインスタンスを返します。

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

例 3: 手動ページングを使用して、ユーザーのスタックのすべてを記述するスタックインスタンスのコレクションを返します。次のページの開始トークンは、呼び出しごとに取得されます。$null は、取得する詳細情報が残っていないことを示します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeStacks」の「」を参照してください。 AWS Tools for PowerShell

Get-CFNStackEvent

次の例は、Get-CFNStackEvent を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックに関するすべてのスタック関連イベントを返します。

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

例 2: 指定されたトークンから始まる手動ページングを使用して、指定されたスタックに関するすべてのスタック関連イベントを返します。次のページの開始トークンは、呼び出しごとに取得されます。$null は、取得するイベントが残っていないことを示します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeStackEvents」の「」を参照してください。 AWS Tools for PowerShell

Get-CFNStackResource

次の例は、Get-CFNStackResource を使用する方法を説明しています。

のツール PowerShell

例 1: 論理 ID「MyDBInstance」を使用して、指定されたスタックに関連付けられているテンプレートで識別されたリソースの説明を返します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeStackResource」の「」を参照してください。 AWS Tools for PowerShell

Get-CFNStackResourceList

次の例は、Get-CFNStackResourceList を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックに関連付けられた最大 100 個の AWS リソースのリソースの説明を返します。スタックに関連付けられているすべてのリソースの詳細を取得するには StackResourceSummary、Get-CFN を使用します。これは、結果の手動ページングもサポートしています。

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

例 2: 論理 ID「Ec2Instance」を使用して、指定されたスタックに関連付けられているテンプレートで識別された Amazon EC2 インスタンスの説明を返します。

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

例 3: インスタンス ID「i-123456」で識別される Amazon EC2 インスタンスが含まれるスタックに関連付けられた、最大 100 個のリソースの説明を返します。スタックに関連付けられているすべてのリソースの詳細を取得するにはStackResourceSummary、Get-CFN を使用します。これは、結果の手動ページングもサポートしています。

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

例 4: スタックのテンプレートで論理 ID「Ec2Instance」によって識別される Amazon EC2 インスタンスの説明を返します。スタックは、スタックに含まれるリソースの物理リソース ID を使用して識別され、この場合はインスタンス ID が「i-123456」の Amazon EC2 インスタンスも含まれます。テンプレートのコンテンツによっては、スタックの識別に異なる物理リソース (Amazon S3 バケットなど) を使用することもできます。

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

- API の詳細については、「 コマンドレットリファレンスDescribeStackResources」の「」を参照してください。 AWS Tools for PowerShell

Get-CFNStackResourceSummary

次の例は、Get-CFNStackResourceSummary を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックに関連付けられているすべてのリソースの説明を返します。

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

例 2: 結果の手動ページングを使用して、指定されたスタックに関連付けられているすべてのリソースの説明を返します。次のページの開始トークンは、呼び出しごとに取得されます。$null は、取得する詳細情報が残っていないことを示します。

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

- API の詳細については、「 コマンドレットリファレンスListStackResources」の「」を参照してください。 AWS Tools for PowerShell

Get-CFNStackSummary

次の例は、Get-CFNStackSummary を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのスタックの概要情報を返します。

```
Get-CFNStackSummary
```

例 2: 現在作成中のすべてのスタックの概要情報を返します。

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

例 3: 現在作成中または更新中のすべてのスタックの概要情報を返します。

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

例 4: 結果の手動ページングを使用して、現在作成中または更新中のすべてのスタックの概要情報を返します。次のページの開始トークンは、呼び出しごとに取得されます。$null は、取得する詳細情報が残っていないことを示します。

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

- API の詳細については、「 コマンドレットリファレンスListStacks」の「」を参照してください。 AWS Tools for PowerShell

`Get-CFNTemplate`

次の例は、`Get-CFNTemplate` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックに関連付けられているテンプレートを返します。

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

- API の詳細については、「 コマンドレットリファレンスGetTemplate」の「」を参照してください。 AWS Tools for PowerShell

Measure-CFNTemplateCost

次の例は、Measure-CFNTemplateCost を使用する方法を説明しています。

のツール PowerShell

例 1: テンプレートの実行に必要なリソースを記述するクエリ文字列を含む AWS Simple Monthly Calculator URL を返します。テンプレートは指定された Amazon S3 URL から取得され、単一のカスタマイゼーションパラメータが適用されます。パラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKeyParameterValue」を使用して指定することもできます。

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

例 2: テンプレートの実行に必要なリソースを記述するクエリ文字列を含む AWS Simple Monthly Calculator URL を返します。テンプレートは、指定されたコンテンツと適用されたカスタマイズパラメータから解析されます (この例では、テンプレートコンテンツが 'KeyName' と '' の 2 つのパラメータを宣言したと仮定InstanceTypeします)。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。

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

例 3: New-Object を使用して一連のテンプレートパラメータを構築し、テンプレートの実行に必要なリソースを記述するクエリ文字列を含む AWS Simple Monthly Calculator URL を返します。テンプレートは、カスタマイズパラメータを使用して、指定されたコンテンツから解析されます (この例では、テンプレートコンテンツが 'KeyName' と '' の 2 つのパラメータを宣言したと仮定InstanceTypeします)。

```
$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 )
```

- API の詳細については、「 コマンドレットリファレンスEstimateTemplateCost」の「」を参照してください。 AWS Tools for PowerShell

New-CFNStack

次の例は、New-CFNStack を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。スタックの作成に失敗しても、スタックはロールバックされません。

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

例 2: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイゼーションパラメータを使用して、提供されたコンテンツから解析されます (「PK1」と「PK2」はテンプレートコンテンツで宣言されたパラメータの名前を表し、「PV1」と「PV2」はこれらのパラメータの値を表します)。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。スタックの作成に失敗すると、スタックがロールバックされます。

```
$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"
```

例 3: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイズパラメータ Amazon S3 ('PK1' はテンプレートコンテンツで宣言されたパラメータの名前を表し、'PV1' はパラメータの値を表します。 カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。 スタックの作成に失敗すると、ロールバックされます (-DisableRollback $false を指定するのと同じです)。

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

例 4: 指定された名前を使用して新しいスタックを作成します。テンプレートは、カスタマイズパラメータ Amazon S3 ('PK1' はテンプレートコンテンツで宣言されたパラメータの名前を表し、'PV1' はパラメータの値を表します。 カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。 スタックの作成が失敗すると、ロールバックされます (-DisableRollback $false を指定するのと同じです)。指定された通知 AEN は、公開されたスタック関連のイベントを受け取ります。

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

- API の詳細については、「 コマンドレットリファレンスCreateStack」の「」を参照してください。 AWS Tools for PowerShell

Remove-CFNStack

次の例は、Remove-CFNStack を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックを削除します。

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

- API の詳細については、「 コマンドレットリファレンスDeleteStack」の「」を参照してください。 AWS Tools for PowerShell

`Resume-CFNUpdateRollback`

次の例は、`Resume-CFNUpdateRollback` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックのロールバックを続行します。このスタックの状態は「UPDATE_ROLLBACK_FAILED」になっている必要があります。続行されたロールバックが成功すると、スタックの状態が「UPDATE_ROLLBACK_COMPLETE」になります。

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- API の詳細については、「 コマンドレットリファレンスContinueUpdateRollback」の「」を参照してください。 AWS Tools for PowerShell

`Stop-CFNUpdateStack`

次の例は、`Stop-CFNUpdateStack` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたスタックで行われている更新をキャンセルします。

```
Stop-CFNUpdateStack -StackName "myStack"
```

- API の詳細については、「 コマンドレットリファレンスCancelUpdateStack」の「」を参照してください。 AWS Tools for PowerShell

`Test-CFNStack`

次の例は、`Test-CFNStack` を使用する方法を説明しています。

のツール PowerShell

例 1: スタックが UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE、または UPDATE_COMPLETE のいずれかの状態に達したかどうかをテストします。

```
Test-CFNStack -StackName MyStack
```

出力:

```
False
```

例 2: スタックが UPDATE_COMPLETE または UPDATE_ROLLBACK_COMPLETE のいずれかのステータスに達したかどうかをテストします。

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

出力:

```
True
```

- API の詳細については、「 コマンドレットリファレンス」の「Test-CFNStack」を参照してください。 AWS Tools for PowerShell

`Test-CFNTemplate`

次の例は、`Test-CFNTemplate` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたテンプレートコンテンツを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

例 2: Amazon S3 URL 経由でアクセスされる、指定されたテンプレートを検証します。出力は、テンプレートの機能、説明、パラメータを詳しく説明します。

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template
```

- API の詳細については、「 コマンドレットリファレンスValidateTemplate」の「」を参照してください。 AWS Tools for PowerShell

Update-CFNStack

次の例は、Update-CFNStack を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV1」はその値を表します。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 2: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 3: 指定されたテンプレートとカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」はテンプレートで宣言されているパラメータの名前を表し、「PV2」はその値を表します。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

例 4: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

例 5: 指定されたテンプレート (Amazon S3 から取得されたもの) とカスタマイゼーションパラメータでスタック「MyStack」(この例では IAM リソースが含まれていることを想定) を更新します。「PK1」と「PK2」はテンプレートで宣言されているパラメータの名前を表し、「PV1」と「PV2」はそれらの要求された値を表します。カスタマイズパラメータは、「Key」と「Value」ではなく「Key」と「ValueParameterKey」を使用して指定することもできますParameterValue。IAM リソースを含むスタックでは、-Capabilities "CAPABILITY_IAM" パラメータを指定する必要があります。指定しないと、更新はInsufficientCapabilities「」エラーで失敗します。

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API の詳細については、「 コマンドレットリファレンスUpdateStack」の「」を参照してください。 AWS Tools for PowerShell

Wait-CFNStack

次の例は、Wait-CFNStack を使用する方法を説明しています。

のツール PowerShell

例 1: スタックが UPDATE_ROLLBACK_COMPLETE、CREATE_COMPLETE、ROLLBACK_COMPLETE、または UPDATE_COMPLETE のいずれかの状態に達したかどうかをテストします。スタックが 状態の 1

つにない場合、コマンドは 2 秒間スリープしてから、ステータスを再度テストします。これは、スタックがリクエストされた状態のいずれかに達するか、デフォルトのタイムアウト期間が 60 秒経過するまで繰り返されます。タイムアウト期間を超えると、例外がスローされます。スタックがタイムアウト期間内にリクエストされた状態のいずれかに達すると、パイプラインに返されます。

```
$stack = Wait-CFNStack -StackName MyStack
```

例 2: この例では、スタックが指定された状態のいずれかになるまで、合計 5 分 (300 秒) 待機します。この例では、タイムアウトの前に 状態に達したため、スタックオブジェクトはパイプラインに返されます。

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

出力:

```
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
```

例 3: この例は、スタックがタイムアウト期間 (この場合はデフォルト期間 60 秒) 内にリクエストされた状態のいずれかに到達しなかった場合のエラー出力を示しています。

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンス」の「Wait-CFNStack」を参照してください。 AWS Tools for PowerShell

CloudFront Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CloudFront。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-CFCloudFrontOriginAccessIdentity`

次の例は、Get-CFCloudFrontOriginAccessIdentity を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、-Id パラメータで指定された特定の Amazon CloudFront オリジンアクセスアイデンティティを返します。-Id パラメータは必須ではありませんが、指定しないと、結果は返されません。

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

出力:

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    ------------------------------------    --
----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- API の詳細については、「 コマンドレットリファレンスGetCloudFrontOriginAccessIdentity」の「」を参照してください。 AWS Tools for PowerShell

`Get-CFCloudFrontOriginAccessIdentityConfig`

次の例は、Get-CFCloudFrontOriginAccessIdentityConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、-Id パラメータで指定された単一の Amazon CloudFront オリジンアクセスアイデンティティに関する設定情報を返します。-Id パラメータを指定しないと、エラーが発生します。

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

出力:

```
     CallerReference                                                 Comment
     ---------------                                                 -------
     mycallerreference: 2/1/2011 1:16:32 PM                          Caller reference:
2/1/2011 1:16:32 PM
```

- API の詳細については、「 コマンドレットリファレンスGetCloudFrontOriginAccessIdentityConfig」の「」を参照してください。 AWS Tools for PowerShell

Get-CFCloudFrontOriginAccessIdentityList

次の例は、Get-CFCloudFrontOriginAccessIdentityList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Amazon CloudFront オリジンアクセスアイデンティティのリストを返します。MaxItem パラメータは 2 の値を指定するため、結果には 2 つの ID が含まれます。

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

出力:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- API の詳細については、「 コマンドレットリファレンスListCloudFrontOriginAccessIdentities」の「」を参照してください。 AWS Tools for PowerShell

Get-CFDistribution

次の例は、Get-CFDistribution を使用する方法を説明しています。

のツール PowerShell

例 1: 特定のディストリビューションに関する情報を取得します。

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- API の詳細については、「 コマンドレットリファレンスGetDistribution」の「」を参照してください。 AWS Tools for PowerShell

Get-CFDistributionConfig

次の例は、Get-CFDistributionConfig を使用する方法を説明しています。

のツール PowerShell

例 1: 特定のディストリビューションの設定を取得します。

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- API の詳細については、「 コマンドレットリファレンスGetDistributionConfig」の「」を参照してください。 AWS Tools for PowerShell

Get-CFDistributionList

次の例は、Get-CFDistributionList を使用する方法を説明しています。

のツール PowerShell

例 1: ディストリビューションを返します。

```
Get-CFDistributionList
```

- API の詳細については、「 コマンドレットリファレンスListDistributions」の「」を参照してください。 AWS Tools for PowerShell

New-CFDistribution

次の例は、New-CFDistribution を使用する方法を説明しています。

のツール PowerShell

例 1: ログ記録とキャッシュで設定された基本的な CloudFront ディストリビューションを作成します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateDistribution」の「」を参照してください。 AWS Tools for PowerShell

New-CFInvalidation

次の例は、New-CFInvalidation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ID が EXAMPLENSTXAXE であるディストリビューションで、新しいキャッシュ削除を作成します。 CallerReference はユーザーが選択した一意の ID です。この場合、2019 年 5 月 15 日午前 9 時を表すタイムスタンプが使用されます。$Paths 変数には、ユーザーがディストリビューションのキャッシュに含めたくない画像ファイルやメディアファイルへの 3 つのパスが格納されます。-Paths_Quantity パラメータ値は、-Paths_Item パラメータで指定したパスの総数です。

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

出力:

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- API の詳細については、「 コマンドレットリファレンスCreateInvalidation」の「」を参照してください。 AWS Tools for PowerShell

New-CFSignedCookie

次の例は、New-CFSignedCookie を使用する方法を説明しています。

のツール PowerShell

例 1: 既定ポリシーを使用して、指定されたリソースへの署名付き Cookie を作成します。Cookie は 1 年間有効です。

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
```

```
}
New-CFSignedCookie @params
```

出力:

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

例 2: カスタムポリシーを使用して、指定されたリソースへの署名付き Cookie を作成します。Cookie は 24 時間後に有効になり、1 週間後に期限切れになります。

```
$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
```

出力:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

例 3: カスタムポリシーを使用して、指定されたリソースへの署名付き Cookie を作成します。Cookie は 24 時間後に有効になり、1 週間後に期限切れになります。リソースへのアクセスは、指定された ip 範囲に制限されます。

```
$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
```

出力:

```
Policy
                                                                  ------


                                    [CloudFront-Policy, eyJTd...wIjo...
```

- API の詳細については、「 コマンドレットリファレンス」の「New-CFSignedCookie」を参照してください。 AWS Tools for PowerShell

New-CFSignedUrl

次の例は、New-CFSignedUrl を使用する方法を説明しています。

のツール PowerShell

例 1: 既定ポリシーを使用して、指定されたリソースへの署名付き URL を作成します。URL は 1 時間有効です。署名付き URL を含む System.Uri オブジェクトがパイプラインに出力されます。

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

例 2: カスタムポリシーを使用して、指定されたリソースへの署名付き URL を作成します。URL は 24 時間後に有効になり、1 週間後に期限切れになります。

```
$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
```

例 3: カスタムポリシーを使用して、指定されたリソースへの署名付き URL を作成します。URL は 24 時間後に有効になり、1 週間後に期限切れになります。リソースへのアクセスは、指定された ip 範囲に制限されます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンス」の「New-CFSignedUrl」を参照してください。 AWS Tools for PowerShell

CloudTrail Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CloudTrail。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Find-CTEvent

次の例は、Find-CTEvent を使用する方法を説明しています。

のツール PowerShell

例 1: 過去 7 日間に発生したすべてのイベントを返します。コマンドレットはデフォルトで、すべてのイベントを配信するために複数の呼び出しを自動的に実行し、サービスがそれ以上データがないことを示すと終了します。

```
Find-CTEvent
```

例 2: 現在のシェルのデフォルトではないリージョンを指定して、過去 7 日間に発生したすべてのイベントを返します。

```
Find-CTEvent -Region eu-central-1
```

例 3: RunInstances API コールに関連付けられているすべてのイベントを返します。

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

例 4: 使用可能な最初の 5 つのイベントを返します。追加のイベントを取得するために使用するトークンは、'NextToken' という名前のメモプロパティとして**$AWSHistory.LastServiceResponse**メンバーにアタッチされます。

```
Find-CTEvent -MaxResult 5
```

例 5: 前の呼び出しの「次のページ」トークンを使用して次の 10 個のイベントを返し、シーケンス内のイベントをどこから返すかを示します。

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

例 6: この例は、手動ページングを使用して使用可能なイベントをループスルーし、呼び出しごとに最大 5 つのイベントを取得する方法を示しています。

```
$nextToken = $null
do
```

```
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスLookupEvents」の「」を参照してください。 AWS Tools for PowerShell

Get-CTTrail

次の例は、Get-CTTrail を使用する方法を説明しています。

のツール PowerShell

例 1: アカウントの現在のリージョンに関連付けられているすべての証跡の設定を返します。

```
Get-CTTrail
```

例 2: 指定された証跡の設定を返します。

```
Get-CTTrail -TrailNameList trail1,trail2
```

例 3: 現在のシェルのデフォルト以外のリージョン (この場合はフランクフルト (eu-central-1) リージョン) で作成された指定された証跡の設定を返します。

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrails」の「」を参照してください。 AWS Tools for PowerShell

Get-CTTrailStatus

次の例は、Get-CTTrailStatus を使用する方法を説明しています。

のツール PowerShell

例 1: 証跡のステータス情報を「」という名前で返しますmyExampleTrail。返されるデータには、配信エラー、Amazon SNS および Amazon S3 エラー、証跡の口グ記録の開始時間と停止時間に関する情報が含まれます。この例では、証跡が現在のシェルのデフォルトと同じリージョンに作成されたことを前提としています。

```
Get-CTTrailStatus -Name myExampleTrail
```

例 2: 現在のシェルのデフォルト以外のリージョン (この場合はフランクフルト (eu-central-1) リージョン) で作成された証跡のステータス情報を返します。

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- API の詳細については、「 コマンドレットリファレンスGetTrailStatus」の「」を参照してください。 AWS Tools for PowerShell

New-CTTrail

次の例は、New-CTTrail を使用する方法を説明しています。

のツール PowerShell

例 1: ログファイルストレージにバケット「mycloudtrailbucket」を使用する証跡を作成します。

```
New-CTTrail -Name="awscloudtrail-example" -S3BucketName="mycloudtrailbucket"
```

例 2: ログファイルストレージにバケット「mycloudtrailbucket」を使用する証跡を作成します。ログを表す S3 オブジェクトには、「mylogs」という共通のキープレフィックスがあります。新しいログがバケットに配信されると、SNS トピック「mlog-deliverytopic」に通知が送信されます。この例では、スプラッティングを使用してパラメータ値を コマンドレットに指定します。

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="mycloudtrailbucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- API の詳細については、「 コマンドレットリファレンスCreateTrail」の「」を参照してください。 AWS Tools for PowerShell

Remove-CTTrail

次の例は、Remove-CTTrail を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された証跡を削除します。コマンドを実行する前に確認を求められます。確認を抑制するには、-Force スイッチパラメータを追加します。

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- API の詳細については、「 コマンドレットリファレンスDeleteTrail」の「」を参照してください。 AWS Tools for PowerShell

`Start-CTLogging`

次の例は、`Start-CTLogging` を使用する方法を説明しています。

のツール PowerShell

例 1: " という名前の証跡の AWS API コールとログファイルの配信の記録を開始します myExampleTrail。この例では、証跡が現在のシェルのデフォルトと同じリージョンに作成されたことを前提としています。

```
Start-CTLogging -Name myExampleTrail
```

例 2: 現在のシェルのデフォルト以外のリージョン (この場合はフランクフルト (eu-central-1) リージョン) で作成された証跡の AWS API コールとログファイル配信の記録を開始します。

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API の詳細については、「 コマンドレットリファレンスStartLogging」の「」を参照してください。 AWS Tools for PowerShell

`Stop-CTLogging`

次の例は、`Stop-CTLogging` を使用する方法を説明しています。

のツール PowerShell

例 1: 「」という名前の証跡の AWS API コールとログファイルの配信の記録を停止します myExampleTrail。この例では、証跡が現在のシェルのデフォルトと同じリージョンに作成されたことを前提としています。

```
Stop-CTLogging -Name myExampleTrail
```

例 2: 現在のシェルのデフォルト以外のリージョン (この場合はフランクフルト (eu-central-1) リージョン) で作成された証跡の AWS API コールとログファイル配信の記録を停止します。

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API の詳細については、「 コマンドレットリファレンスStopLogging」の「」を参照してください。 AWS Tools for PowerShell

`Update-CTTrail`

次の例は、`Update-CTTrail` を使用する方法を説明しています。

のツール PowerShell

例 1: グローバルサービスイベント (IAM からのイベントなど) が記録されるように指定された証跡を更新し、今後のログファイルの共通キープレフィックスを「globallogs」に変更します。

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

例 2: 指定された証跡を更新して、新しいログ配信に関する通知が指定された SNS トピックに送信されるようにします。

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

例 3: ログが別のバケットに配信されるように、指定された証跡を更新します。

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- API の詳細については、「 コマンドレットリファレンスUpdateTrail」の「」を参照してください。 AWS Tools for PowerShell

CloudWatch Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CloudWatch。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

アクション

`Get-CWDashboard`

次の例は、`Get-CWDashboard` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたダッシュボードの本体である arn を返します。

```
Get-CWDashboard -DashboardName Dashboard1
```

出力:

```
DashboardArn                                              DashboardBody
------------                                              -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- API の詳細については、「 コマンドレットリファレンスGetDashboard」の「」を参照してください。 AWS Tools for PowerShell

`Get-CWDashboardList`

次の例は、`Get-CWDashboardList` を使用する方法を説明しています。

のツール PowerShell

例 1: アカウントのダッシュボードのコレクションを返します。

```
Get-CWDashboardList
```

出力:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

例 2: 名前が 'dev' で始まるアカウントのダッシュボードのコレクションを返します。

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- API の詳細については、「 コマンドレットリファレンスListDashboards」の「」を参照してください。 AWS Tools for PowerShell

`Remove-CWDashboard`

次の例は、`Remove-CWDashboard` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定したダッシュボードを削除し、次に進む前に確認を促します。確認を省略するには、-Force スイッチをコマンドに追加します。

```
Remove-CWDashboard -DashboardName Dashboard1
```

- API の詳細については、「 コマンドレットリファレンスDeleteDashboards」の「」を参照してください。 AWS Tools for PowerShell

`Write-CWDashboard`

次の例は、`Write-CWDashboard` を使用する方法を説明しています。

のツール PowerShell

例 1: 'Dashboard1' という名前のダッシュボードを作成または更新して、2 つのメトリクスウィジェットを並べて表示します。

```
$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
```

例 2: ダッシュボードを作成または更新し、ダッシュボードを説明するコンテンツをコマンドレットにパイプします。

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- API の詳細については、「 コマンドレットリファレンスPutDashboard」の「」を参照してください。 AWS Tools for PowerShell

`Write-CWMetricData`

次の例は、`Write-CWMetricData` を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい MetricDatum オブジェクトを作成し、Amazon Web Services CloudWatch メトリクスに書き込みます。

```
### 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
```

- API の詳細については、「 コマンドレットリファレンスPutMetricData」の「」を参照してください。 AWS Tools for PowerShell

CodeCommit Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CodeCommit。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

アクション

`Get-CCBranch`

次の例は、`Get-CCBranch` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリの指定されたブランチに関する情報を取得します。

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

出力:

```
BranchName                               CommitId
----------                               --------
MyNewBranch                              7763222d...561fc9c9
```

- API の詳細については、「 コマンドレットリファレンスGetBranch」の「」を参照してください。 AWS Tools for PowerShell

Get-CCBranchList

次の例は、Get-CCBranchList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したリポジトリのブランチ名のリストを取得します。

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

出力:

```
master
MyNewBranch
```

- API の詳細については、「 コマンドレットリファレンスListBranches」の「」を参照してください。 AWS Tools for PowerShell

Get-CCRepository

次の例は、Get-CCRepository を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリの情報を取得します。

```
Get-CCRepository -RepositoryName MyDemoRepo
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetRepository」の「」を参照してください。 AWS Tools for PowerShell

Get-CCRepositoryBatch

次の例は、Get-CCRepositoryBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリのうち、見つかったものと見つからないものを確認します。

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

出力:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                 {AMissingRepo}
```

- API の詳細については、「 コマンドレットリファレンスBatchGetRepositories」の「」を参照してください。 AWS Tools for PowerShell

Get-CCRepositoryList

次の例は、Get-CCRepositoryList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのリポジトリをリポジトリ名で昇順に一覧表示します。

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

出力:

```
RepositoryId                            RepositoryName
------------                            --------------
```

```
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- API の詳細については、「 コマンドレットリファレンスListRepositories」の「」を参照してください。 AWS Tools for PowerShell

New-CCBranch

次の例は、New-CCBranch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリに指定された名前と指定されたコミット ID を持つ新しいブランチを作成します。

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- API の詳細については、「 コマンドレットリファレンスCreateBranch」の「」を参照してください。 AWS Tools for PowerShell

New-CCRepository

次の例は、New-CCRepository を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前と指定された説明を持つ新しいリポジトリを作成します。

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateRepository」の「」を参照してください。 AWS Tools for PowerShell

Remove-CCRepository

次の例は、Remove-CCRepository を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリを強制的に削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでリポジトリを削除します。

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

出力:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスDeleteRepository」の「」を参照してください。 AWS Tools for PowerShell

Update-CCDefaultBranch

次の例は、Update-CCDefaultBranch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリのデフォルトブランチを指定されたブランチに変更します。

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- API の詳細については、「 コマンドレットリファレンスUpdateDefaultBranch」の「」を参照してください。 AWS Tools for PowerShell

Update-CCRepositoryDescription

次の例は、Update-CCRepositoryDescription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリの説明を変更します。

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- API の詳細については、「 コマンドレットリファレンスUpdateRepositoryDescription」の「」を参照してください。 AWS Tools for PowerShell

Update-CCRepositoryName

次の例は、Update-CCRepositoryName を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリポジトリの名前を変更します。

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- API の詳細については、「 コマンドレットリファレンスUpdateRepositoryName」の「」を参照してください。 AWS Tools for PowerShell

CodeDeploy Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CodeDeploy。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

アクション

Add-CDOnPremiseInstanceTag

次の例は、Add-CDOnPremiseInstanceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたオンプレミスインスタンスの指定されたキーと値を持つオンプレミスインスタンスタグを追加します。

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- API の詳細については、「 コマンドレットリファレンスAddTagsToOnPremisesInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-CDApplication

次の例は、Get-CDApplication を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションに関する情報を取得します。

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

出力:

```
ApplicationId                         ApplicationName            CreateTime
         LinkedToGitHub
```

```
-------------                           ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- API の詳細については、「 コマンドレットリファレンスGetApplication」の「」を参照してください。 AWS Tools for PowerShell

Get-CDApplicationBatch

次の例は、Get-CDApplicationBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションに関する情報を取得します。

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスBatchGetApplications」の「」を参照してください。 AWS Tools for PowerShell

Get-CDApplicationList

次の例は、Get-CDApplicationList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能なアプリケーションのリストを取得します。

```
Get-CDApplicationList
```

出力:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- API の詳細については、「 コマンドレットリファレンスListApplications」の「」を参照してください。 AWS Tools for PowerShell

Get-CDApplicationRevision

次の例は、Get-CDApplicationRevision を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションリビジョンに関する情報を取得します。

```
$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)
```

出力:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- API の詳細については、「 コマンドレットリファレンスGetApplicationRevision」の「」を参照してください。 AWS Tools for PowerShell

Get-CDApplicationRevisionList

次の例は、Get-CDApplicationRevisionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションで使用可能なリビジョンに関する情報を取得します。

```
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)
>>   }
>> }
>>
```

出力:

```
Type = S3, Bucket = MyBucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- API の詳細については、「 コマンドレットリファレンスListApplicationRevisions」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeployment

次の例は、Get-CDDeployment を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイに関する概要情報を取得します。

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

出力:

```
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
```

例 2: この例では、指定したデプロイに参加しているインスタンスのステータスに関する情報を取得します。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

出力:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

例 3: この例では、指定されたデプロイのアプリケーションリビジョンに関する情報を取得します。

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

出力:

```
Bucket     : MyBucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- API の詳細については、「 コマンドレットリファレンスGetDeployment」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentBatch

次の例は、Get-CDDeploymentBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイに関する情報を取得します。

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスBatchGetDeployments」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentConfig

次の例は、Get-CDDeploymentConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイ設定に関する概要情報を取得します。

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

出力:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    --------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

例 2: この例では、指定されたデプロイ設定の定義に関する情報を取得します。

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

出力:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- API の詳細については、「 コマンドレットリファレンスGetDeploymentConfig」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentConfigList

次の例は、Get-CDDeploymentConfigList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能なデプロイ設定のリストを取得します。

```
Get-CDDeploymentConfigList
```

出力:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- API の詳細については、「 コマンドレットリファレンスListDeploymentConfigs」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentGroup

次の例は、Get-CDDeploymentGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイグループに関する情報を取得します。

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

出力:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- API の詳細については、「 コマンドレットリファレンスGetDeploymentGroup」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentGroupList

次の例は、Get-CDDeploymentGroupList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したアプリケーションのデプロイグループのリストを取得します。

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

出力:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- API の詳細については、「 コマンドレットリファレンスListDeploymentGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentInstance

次の例は、Get-CDDeploymentInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイの指定されたインスタンスに関する情報を取得します。

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetDeploymentInstance」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentInstanceList

次の例は、Get-CDDeploymentInstanceList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したデプロイのインスタンス IDsのリストを取得します。

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

出力:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- API の詳細については、「 コマンドレットリファレンスListDeploymentInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-CDDeploymentList

次の例は、Get-CDDeploymentList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションとデプロイグループのデプロイ IDsのリストを取得します。

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

出力:

```
d-QZMRGSTEX
```

```
d-RR0T5KTEX
```

- API の詳細については、「 コマンドレットリファレンスListDeployments」の「」を参照してください。 AWS Tools for PowerShell

Get-CDOnPremiseInstance

次の例は、Get-CDOnPremiseInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたオンプレミスインスタンスに関する情報を取得します。

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスGetOnPremisesInstance」の「」を参照してください。 AWS Tools for PowerShell

Get-CDOnPremiseInstanceBatch

次の例は、Get-CDOnPremiseInstanceBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたオンプレミスインスタンスに関する情報を取得します。

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスBatchGetOnPremisesInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-CDOnPremiseInstanceList

次の例は、Get-CDOnPremiseInstanceList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能なオンプレミスインスタンス名のリストを取得します。

```
Get-CDOnPremiseInstanceList
```

出力:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- API の詳細については、「 コマンドレットリファレンスListOnPremisesInstances」の「」を参照してください。 AWS Tools for PowerShell

New-CDApplication

次の例は、New-CDApplication を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前の新しいアプリケーションを作成します。

```
New-CDApplication -ApplicationName MyNewApplication
```

出力:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスCreateApplication」の「」を参照してください。 AWS Tools for PowerShell

New-CDDeployment

次の例は、New-CDDeployment を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイ設定とアプリケーションリビジョンを使用して、指定されたアプリケーションとデプロイグループの新しいデプロイを作成します。

```
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
```

出力:

```
d-ZHROG7UEX
```

例 2: この例では、ブルー/グリーンデプロイの置換環境に含めるためにインスタンスを識別する必要がある EC2 インスタンスタグのグループを指定する方法を示します。

```
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=
```

出力:

```
d-ZHROG7UEX
```

- API の詳細については、「 コマンドレットリファレンスCreateDeployment」の「」を参照してください。 AWS Tools for PowerShell

New-CDDeploymentConfig

次の例は、New-CDDeploymentConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前と動作で新しいデプロイ設定を作成します。

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

出力:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスCreateDeploymentConfig」の「」を参照してください。 AWS Tools for PowerShell

New-CDDeploymentGroup

次の例は、New-CDDeploymentGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションの指定された名前、Auto Scaling グループ、デプロイ設定、タグ、およびサービスロールを持つデプロイグループを作成します。

```
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
```

出力:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

例 2: この例では、ブルー/グリーンデプロイの置換環境に含めるためにインスタンスを識別する必要がある EC2 インスタンスタグのグループを指定する方法を示します。

```
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=
```

出力:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスCreateDeploymentGroup」の「」を参照してください。 AWS Tools for PowerShell

Register-CDApplicationRevision

次の例は、Register-CDApplicationRevision を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したアプリケーションの指定した Amazon S3 ロケーションにアプリケーションリビジョンを登録します。

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 MyBucket -S3Location_BundleType zip -S3Location_Key aws-codedeploy_linux-master.zip
 -Revision_RevisionType S3
```

- API の詳細については、「 コマンドレットリファレンスRegisterApplicationRevision」の「」を参照してください。 AWS Tools for PowerShell

Register-CDOnPremiseInstance

次の例は、Register-CDOnPremiseInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オンプレミスインスタンスを指定された名前と IAM ユーザーに登録します。

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- API の詳細については、「 コマンドレットリファレンスRegisterOnPremisesInstance」の「」を参照してください。 AWS Tools for PowerShell

Remove-CDApplication

次の例は、Remove-CDApplication を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した名前のアプリケーションを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでアプリケーションを削除します。

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- API の詳細については、「 コマンドレットリファレンスDeleteApplication」の「」を参照してください。 AWS Tools for PowerShell

Remove-CDDeploymentConfig

次の例は、Remove-CDDeploymentConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した名前のデプロイ設定を削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでデプロイ設定を削除します。

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- API の詳細については、「 コマンドレットリファレンスDeleteDeploymentConfig」の「」を参照してください。 AWS Tools for PowerShell

Remove-CDDeploymentGroup

次の例は、Remove-CDDeploymentGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションの指定された名前のデプロイグループを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでデプロイグループを削除します。

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- API の詳細については、「 コマンドレットリファレンスDeleteDeploymentGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-CDOnPremiseInstanceTag

次の例は、Remove-CDOnPremiseInstanceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前のオンプレミスインスタンスの指定されたタグを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでタグを削除します。

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- API の詳細については、「 コマンドレットリファレンスRemoveTagsFromOnPremisesInstances」の「」を参照してください。 AWS Tools for PowerShell

Stop-CDDeployment

次の例は、Stop-CDDeployment を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたデプロイ ID でデプロイを停止しようとします。

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

出力:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- API の詳細については、「 コマンドレットリファレンスStopDeployment」の「」を参照してください。 AWS Tools for PowerShell

Unregister-CDOnPremiseInstance

次の例は、Unregister-CDOnPremiseInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オンプレミスインスタンスを指定された名前で登録解除します。

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- API の詳細については、「 コマンドレットリファレンスDeregisterOnPremisesInstance」の「」を参照してください。 AWS Tools for PowerShell

Update-CDApplication

次の例は、Update-CDApplication を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したアプリケーションの名前を変更します。

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- API の詳細については、「 コマンドレットリファレンスUpdateApplication」の「」を参照してください。 AWS Tools for PowerShell

Update-CDDeploymentGroup

次の例は、Update-CDDeploymentGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーションの指定されたデプロイグループの名前を変更します。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

例 2: この例は、ブルー/グリーンデプロイの置換環境に含めるためにインスタンスを識別する必要がある EC2 インスタンスタグのグループを指定する方法を示しています。

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- API の詳細については、「 コマンドレットリファレンスUpdateDeploymentGroup」の「」を参照してください。 AWS Tools for PowerShell

CodePipeline Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています CodePipeline。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Confirm-CPJob`

次の例は、`Confirm-CPJob` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたジョブのステータスを取得します。

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

出力:

```
Value
-----
InProgress
```

- API の詳細については、「 コマンドレットリファレンスAcknowledgeJob」の「」を参照してください。 AWS Tools for PowerShell

`Disable-CPStageTransition`

次の例は、`Disable-CPStageTransition` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプライン内の指定されたステージのインバウンド移行を無効にします。

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- API の詳細については、「 コマンドレットリファレンスDisableStageTransition」の「」を参照してください。 AWS Tools for PowerShell

`Enable-CPStageTransition`

次の例は、`Enable-CPStageTransition` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプライン内の指定されたステージのインバウンド移行を有効にします。

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- API の詳細については、「 コマンドレットリファレンスEnableStageTransition」の「」を参照してください。 AWS Tools for PowerShell

Get-CPActionType

次の例は、Get-CPActionType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された所有者が使用できるすべてのアクションに関する情報を取得します。

```
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)
}
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスListActionTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-CPActionableJobList

次の例は、Get-CPActionableJobList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアクションカテゴリ、所有者、プロバイダー、バージョン、およびクエリパラメータのすべての実用的なジョブに関する情報を取得します。

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

出力:

```
AccountId       Data                                Id
         Nonce
---------       ----                                --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- API の詳細については、「 コマンドレットリファレンスPollForJobs」の「」を参照してください。 AWS Tools for PowerShell

Get-CPJobDetail

次の例は、Get-CPJobDetail を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたジョブに関する一般的な情報を取得します。

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

出力:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

例 2: この例では、指定されたジョブに関する詳細情報を取得します。

```
$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)
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetJobDetails」の「」を参照してください。 AWS Tools for PowerShell

Get-CPPipeline

次の例は、Get-CPPipeline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプラインに関する一般的な情報を取得します。

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

出力:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

例 2: この例では、指定されたパイプラインに関する詳細情報を取得します。

```
$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)
  }
}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetPipeline」の「」を参照してください。 AWS Tools for PowerShell

Get-CPPipelineList

次の例は、Get-CPPipelineList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能なパイプラインのリストを取得します。

```
Get-CPPipelineList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListPipelines」の「」を参照してください。 AWS Tools for PowerShell

Get-CPPipelineState

次の例は、Get-CPPipelineState を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプラインのステージに関する一般的な情報を取得します。

```
Get-CPPipelineState -Name CodePipelineDemo
```

出力:

```
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
```

例 2: この例では、指定されたパイプラインの状態に関する詳細情報を取得します。

```
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)
  }
}
```

出力:

```
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 =
```

- API の詳細については、「 コマンドレットリファレンスGetPipelineState」の「」を参照してください。 AWS Tools for PowerShell

New-CPCustomActionType

次の例は、New-CPCustomActionType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたプロパティを使用して新しいカスタムアクションを作成します。

```
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
```

出力:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- API の詳細については、「 コマンドレットリファレンスCreateCustomActionType」の「」を参照してください。 AWS Tools for PowerShell

New-CPPipeline

次の例は、New-CPPipeline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された設定で新しいパイプラインを作成します。

```
$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
```

出力:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
```

```
Name           : CodePipelineDemo
RoleArn        : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages         : {Source, Beta}
Version        : 1
```

- API の詳細については、「 コマンドレットリファレンスCreatePipeline」の「」を参照してください。 AWS Tools for PowerShell

Remove-CPCustomActionType

次の例は、Remove-CPCustomActionType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたカスタムアクションを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでカスタムアクションを削除します。

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- API の詳細については、「 コマンドレットリファレンスDeleteCustomActionType」の「」を参照してください。 AWS Tools for PowerShell

Remove-CPPipeline

次の例は、Remove-CPPipeline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプラインを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force パラメータを追加して、プロンプトなしでパイプラインを削除します。

```
Remove-CPPipeline -Name CodePipelineDemo
```

- API の詳細については、「 コマンドレットリファレンスDeletePipeline」の「」を参照してください。 AWS Tools for PowerShell

Start-CPPipelineExecution

次の例は、Start-CPPipelineExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパイプラインの実行を開始します。

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- API の詳細については、「 コマンドレットリファレンスStartPipelineExecution」の「」を参照してください。 AWS Tools for PowerShell

Update-CPPipeline

次の例は、Update-CPPipeline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された設定で指定された既存のパイプラインを更新します。

```
$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
```

出力:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- API の詳細については、「 コマンドレットリファレンスUpdatePipeline」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon Cognito ID の例 PowerShell

次のコード例は、Amazon Cognito Identity AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-CGIIdentityPool`

次の例は、`Get-CGIIdentityPool` を使用する方法を説明しています。

のツール PowerShell

例 1: 特定の ID プールに関する情報を ID で取得します。

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

Get-CGIIdentityPoolList

次の例は、Get-CGIIdentityPoolList を使用する方法を説明しています。

のツール PowerShell

例 1: 既存の ID プールのリストを取得します。

```
Get-CGIIdentityPoolList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListIdentityPools」の「」を参照してください。 AWS Tools for PowerShell

Get-CGIIdentityPoolRole

次の例は、Get-CGIIdentityPoolRole を使用する方法を説明しています。

のツール PowerShell

例 1: 特定の ID プールのロールに関する情報を取得します。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetIdentityPoolRoles」の「」を参照してください。 AWS Tools for PowerShell

New-CGIIdentityPool

次の例は、New-CGIIdentityPool を使用する方法を説明しています。

のツール PowerShell

例 1: 認証されていない ID を許可する新しい ID プールを作成します。

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

Remove-CGIIdentityPool

次の例は、Remove-CGIIdentityPool を使用する方法を説明しています。

のツール PowerShell

例 1: 特定の ID プールを削除します。

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API の詳細については、「 コマンドレットリファレンスDeleteIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

Set-CGIIdentityPoolRole

次の例は、Set-CGIIdentityPoolRole を使用する方法を説明しています。

のツール PowerShell

例 1: 認証されていない IAM ロールを持つように特定の ID プールを設定します。

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API の詳細については、「 コマンドレットリファレンスSetIdentityPoolRoles」の「」を参照してください。 AWS Tools for PowerShell

Update-CGIIdentityPool

次の例は、Update-CGIIdentityPool を使用する方法を説明しています。

のツール PowerShell

例 1: ID プールプロパティの一部を更新します。この場合は、ID プールの名前です。

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateIdentityPool」の「」を参照してください。 AWS Tools for PowerShell

AWS Config Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Config。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-CFGResourceTag`

次の例は、`Add-CFGResourceTag` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたタグをリソース ARN に関連付けます。この場合、リソース ARN は config-rule/config-rule-16iyn0 です。

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- API の詳細については、「 コマンドレットリファレンスTagResource」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateComplianceByConfigRuleList

次の例は、Get-CFGAggregateComplianceByConfigRuleList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定の設定ルールの ConfigurationAggregator 「kaju」フィルタリングから詳細を取得し、ルールの「Compliance」を展開/返します。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

出力:

```
ComplianceContributorCount                          ComplianceType
--------------------------                          --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

例 2: この例では、特定の から詳細を取得し ConfigurationAggregator、アグリゲータの対象となるすべてのリージョンについて特定のアカウントに対してフィルタリングし、すべてのルールのコンプライアンスをさらに再開します。

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

出力:

```
ConfigRuleName                    Compliance
--------------                    ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip         NON_COMPLIANT
desired-instance-type             NON_COMPLIANT
```

- API の詳細については、「 コマンドレットリファレンスDescribeAggregateComplianceByConfigRules」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateComplianceDetailsByConfigRule

次の例は、Get-CFGAggregateComplianceDetailsByConfigRule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のアカウント、アグリゲータ、リージョン、および設定ルールのdesired-instance-type「COMPLIANT」状態にある AWS Config ルール「」の resource-id と resource-type の出力を選択する評価結果を返します。

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

出力:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- API の詳細については、「 コマンドレットリファレンスGetAggregateComplianceDetailsByConfigRule」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateConfigRuleComplianceSummary

次の例は、Get-CFGAggregateConfigRuleComplianceSummary を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアグリゲータの非準拠ルールの数を返します。

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

出力:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- API の詳細については、「 コマンドレットリファレンスGetAggregateConfigRuleComplianceSummary」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateDiscoveredResourceCount

次の例は、Get-CFGAggregateDiscoveredResourceCount を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、us-east-1 リージョンでフィルタリングされた特定のアグリゲータのリソース数を返します。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

出力:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- ------------------------
           {}                              455
```

例 2: この例では、指定されたアグリゲータのフィルタリングされたリージョンのリソース数を RESOURCE_TYPE でグループ化して返します。

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetAggregateDiscoveredResourceCounts」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateDiscoveredResourceList

次の例は、Get-CFGAggregateDiscoveredResourceList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「アイルランド」アグリゲータに集約された特定のリソースタイプのリソース識別子を返します。リソースタイプのリストについては、https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TConfigServiceResourceType.html&tocid=Amazon_ConfigService_ を確認してください ResourceType。

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

出力:

```
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
```

例 2: この例では、region us-east-1 でフィルタリングされた特定のアグリゲータの「default**AwsEC2SecurityGroup**」という名前のリソースタイプを返します。

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListAggregateDiscoveredResources」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateResourceConfig

次の例は、Get-CFGAggregateResourceConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のリソースの 設定項目を集約して返し、設定を展開します。

```
(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
```

出力:

```
{"description":"default VPC security group","groupName":"default","ipPermissions":
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[{"groupId":"sg-4fc1d234","userId":"123456789012"}],"ipv4Ranges":
[],"ipRanges":[]},{"fromPort":3389,"ipProtocol":"tcp","ipv6Ranges":
[],"prefixListIds":[],"toPort":3389,"userIdGroupPairs":[],"ipv4Ranges":
[{"cidrIp":"54.240.197.224/29","description":"office subnet"},
{"cidrIp":"72.21.198.65/32","description":"home pc"}],"ipRanges":
["54.240.197.224/29","72.21.198.65/32"]}],"ownerId":"123456789012","groupId":"sg-4fc1d234","
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[],"ipv4Ranges":[{"cidrIp":"0.0.0.0/0"}],"ipRanges":["0.0.0.0/0"]}],"tags":
[],"vpcId":"vpc-2d1c2e34"}
```

- API の詳細については、「 コマンドレットリファレンス」のGetAggregateResourceconfig「-service」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregateResourceConfigBatch

次の例は、Get-CFGAggregateResourceConfigBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のアグリゲータに存在するリソース (識別済み) の現在の設定項目を取得します。

```
$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
```

出力:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- ------------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- API の詳細については、「 コマンドレットリファレンス」のBatchGetAggregateResourceconfig「-service」を参照してください。 AWS Tools for PowerShell

Get-CFGAggregationAuthorizationList

次の例は、Get-CFGAggregationAuthorizationList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アグリゲータに付与された認可を取得します。

```
Get-CFGAggregationAuthorizationList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAggregationAuthorizations」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceByConfigRule

次の例は、Get-CFGComplianceByConfigRule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では ebs-optimized-instance、ルールの現在の評価結果がないルール のコンプライアンスの詳細を取得するため、INSUFFICIENT_DATA を返します。

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

出力:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

例 2: この例では、ルール ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK の非準拠リソースの数を返します。

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

出力:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- API の詳細については、「 コマンドレットリファレンスDescribeComplianceByConfigRule」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceByResource

次の例は、Get-CFGComplianceByResource を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**AWS::SSM::ManagedInstanceInventory**リソースタイプに「COMPLIANT」コンプライアンスタイプがないかをチェックします。

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

出力:

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- API の詳細については、「 コマンドレットリファレンスDescribeComplianceByResource」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceDetailsByConfigRule

次の例は、Get-CFGComplianceDetailsByConfigRule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ルールの評価結果を取得し access-keys-rotated 、コンプライアンスタイプ別にグループ化された出力を返します。

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

出力:

```
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...
```

例 2: この例では、COMPLIANT リソースのルール access-keys-rotated のコンプライアンスの詳細をクエリします。

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

出力:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- API の詳細については、「 コマンドレットリファレンスGetComplianceDetailsByConfigRule」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceDetailsByResource

次の例は、Get-CFGComplianceDetailsByResource を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のリソースのエビギュレーション結果を示します。

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

出力:

```
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               :
```

- API の詳細については、「 コマンドレットリファレンスGetComplianceDetailsByResource」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceSummaryByConfigRule

次の例は、Get-CFGComplianceSummaryByConfigRule を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、非準拠の Config ルールの数を返します。

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

出力:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- API の詳細については、「 コマンドレットリファレンスGetComplianceSummaryByConfigRule」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGComplianceSummaryByResourceType

次の例は、Get-CFGComplianceSummaryByResourceType を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、準拠または非準拠のリソースの数を返し、出力を json に変換します。

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
```

```
}
```

- API の詳細については、「 コマンドレットリファレンスGetComplianceSummaryByResourceType」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigRule

次の例は、Get-CFGConfigRule を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、選択したプロパティを使用して、アカウントの設定ルールを一覧表示します。

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

出力:

```
ConfigRuleName                                  ConfigRuleId       ConfigRuleArn
                                                    ConfigRuleState
--------------                                  ------------       -------------
                                                    ---------------
ALB_REDIRECTION_CHECK                           config-rule-12iyn3 arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3 ACTIVE
access-keys-rotated                             config-rule-aospfr arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr ACTIVE
autoscaling-group-elb-healthcheck-required      config-rule-cn1f2x arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x ACTIVE
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigRules」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigRuleEvaluationStatus

次の例は、Get-CFGConfigRuleEvaluationStatus を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された設定ルールのステータス情報を返します。

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigRuleEvaluationStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigurationAggregatorList

次の例は、Get-CFGConfigurationAggregatorList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、リージョン/アカウントのすべてのアグリゲータを返します。

```
Get-CFGConfigurationAggregatorList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigurationAggregators」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigurationAggregatorSourcesStatus

次の例は、Get-CFGConfigurationAggregatorSourcesStatus を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、指定されたアグリゲータ内のソースにリクエストされたフィールドが表示されます。

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigurationAggregatorSourcesStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigurationRecorder

次の例は、Get-CFGConfigurationRecorder を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、設定レコーダーの詳細を返します。

```
Get-CFGConfigurationRecorder | Format-List
```

出力:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigurationRecorders」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGConfigurationRecorderStatus

次の例は、Get-CFGConfigurationRecorderStatus を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、設定レコーダーのステータスを返します。

```
Get-CFGConfigurationRecorderStatus
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeConfigurationRecorderStatus」の「」を参照してください。 AWS Tools for PowerShell

`Get-CFGConformancePack`

次の例は、Get-CFGConformancePack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、すべてのコンフォーマンスパックを一覧表示します。

```
Get-CFGConformancePack
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeConformancePacks」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGDeliveryChannel

次の例は、Get-CFGDeliveryChannel を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョンの配信チャネルを取得し、詳細を表示します。

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

出力:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- API の詳細については、「 コマンドレットリファレンスDescribeDeliveryChannels」の「」を参照してください。 AWS Tools for PowerShell

Get-CFGResourceTag

次の例は、Get-CFGResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のリソースに関連するタグを一覧表示します。

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

出力:

```
Key     Value
---     -----
Version 1.3
```

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

Remove-CFGConformancePack

次の例は、Remove-CFGConformancePack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定されたコンフォーマンスパックと、そのパックのすべてのルール、修復アクション、評価結果を削除します。

```
Remove-CFGConformancePack -ConformancePackName dono
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteConformancePack」の「」を参照してください。 AWS Tools for PowerShell

Write-CFGConformancePack

次の例は、Write-CFGConformancePack を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、特定の yaml ファイルからテンプレートを取得して、コンフォーマンスパックを作成します。

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket kt-ps-examples
 -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- API の詳細については、「 コマンドレットリファレンスPutConformancePack」の「」を参照してください。 AWS Tools for PowerShell

Write-CFGDeliveryChannel

次の例は、Write-CFGDeliveryChannel を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、既存の配信チャネルの deliveryFrequency プロパティを変更します。

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName config-
bucket-NA -DeliveryChannel_S3KeyPrefix my
```

- API の詳細については、「 コマンドレットリファレンスPutDeliveryChannel」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Device Farm の例 PowerShell

次のコード例は、Device Farm AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`New-DFUpload`

次の例は、`New-DFUpload` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Android アプリケーション用の AWS Device Farm アップロードを作成します。プロジェクト ARN は、New-DFProject または Get-DF の出力から取得できます

ProjectList。New-DFUpload 出力の署名付き URL を使用して、ファイルを Device Farm にアップロードします。

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateUpload」の「」を参照してください。 AWS Tools for PowerShell

AWS Directory Service Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Directory Service。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-DSIpRoute`

次の例は、`Add-DSIpRoute` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された Directory-id に割り当てられたリソースタグを削除します

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- API の詳細については、「 コマンドレットリファレンスAddIpRoutes」の「」を参照してください。 AWS Tools for PowerShell

Add-DSResourceTag

次の例は、Add-DSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された Directory-id にリソースタグを追加します。

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- API の詳細については、「 コマンドレットリファレンスAddTagsToResource」の「」を参照してください。 AWS Tools for PowerShell

Approve-DSTrust

次の例は、Approve-DSTrust を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Trustid の AWS Directory Service VerifyTrust API オペレーションを呼び出します。

```
Approve-DSTrust -TrustId t-9067157123
```

- API の詳細については、「 コマンドレットリファレンスVerifyTrust」の「」を参照してください。 AWS Tools for PowerShell

Confirm-DSSharedDirectory

次の例は、Confirm-DSSharedDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリ所有者 から送信されたディレクトリ共有リクエストを受け入れます AWS アカウント。

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスAcceptSharedDirectory」の「」を参照してください。 AWS Tools for PowerShell

Connect-DSDirectory

次の例は、Connect-DSDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オンプレミスディレクトリに接続するための AD Connector を作成します。

```
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
```

- API の詳細については、「 コマンドレットリファレンスConnectDirectory」の「」を参照してください。 AWS Tools for PowerShell

Deny-DSSharedDirectory

次の例は、Deny-DSSharedDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリ所有者アカウントから送信されたディレクトリ共有リクエストを拒否します。

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

出力:

```
d-9067012345
```

- API の詳細については、「 コマンドレットリファレンスRejectSharedDirectory」の「」を参照してください。 AWS Tools for PowerShell

`Disable-DSDirectoryShare`

次の例は、`Disable-DSDirectoryShare` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリ所有者とコンシューマーアカウント間のディレクトリ共有を停止します。

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

出力:

```
d-9067012345
```

- API の詳細については、「 コマンドレットリファレンスUnshareDirectory」の「」を参照してください。 AWS Tools for PowerShell

`Disable-DSLDAPS`

次の例は、`Disable-DSLDAPS` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したディレクトリの LDAP セキュアコールを非アクティブ化します。

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- API の詳細については、「 コマンドレットリファレンス」のDisableLDAPS」を参照してください。 AWS Tools for PowerShell

Disable-DSRadius

次の例は、Disable-DSRadius を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、AD Connector または Microsoft AD ディレクトリ用に設定された RADIUS サーバーを無効にします。

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスDisableRadius」の「」を参照してください。 AWS Tools for PowerShell

Disable-DSSso

次の例は、Disable-DSSso を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリのシングルサインオンを無効にします。

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスDisableSso」の「」を参照してください。 AWS Tools for PowerShell

Enable-DSDirectoryShare

次の例は、Enable-DSDirectoryShare を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ハンドシェイク方式を使用して AWS 、アカウント内の指定されたディレクトリを別の AWS アカウントと共有します。

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

出力:

```
d-9067012345
```

- API の詳細については、「 コマンドレットリファレンスShareDirectory」の「」を参照してください。 AWS Tools for PowerShell

Enable-DSLDAPS

次の例は、Enable-DSLDAPS を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のディレクトリのスイッチをアクティブ化して、常に LDAP セキュアコールを使用します。

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- API の詳細については、「 コマンドレットリファレンス」のEnableLDAPS」を参照してください。 AWS Tools for PowerShell

Enable-DSRadius

次の例は、Enable-DSRadius を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、AD Connector または Microsoft AD ディレクトリ用に提供された RADIUS サーバー設定で多要素認証 (MFA) を有効にします。

```
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
```

- API の詳細については、「 コマンドレットリファレンスEnableRadius」の「」を参照してください。 AWS Tools for PowerShell

Enable-DSSso

次の例は、Enable-DSSso を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリのシングルサインオンを有効にします。

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスEnableSso」の「」を参照してください。 AWS Tools for PowerShell

Get-DSCertificate

次の例は、Get-DSCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、保護された LDAP 接続に登録された証明書に関する情報を表示します。

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

出力:

```
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.
```

- API の詳細については、「 コマンドレットリファレンスDescribeCertificate」の「」を参照してください。 AWS Tools for PowerShell

Get-DSCertificateList

次の例は、Get-DSCertificateList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたディレクトリのセキュリティで保護された LDAP 接続に登録されているすべての証明書を一覧表示します。

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

出力:

```
CertificateId CommonName                 ExpiryDateTime       State
------------- ----------                 --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- API の詳細については、「 コマンドレットリファレンスListCertificates」の「」を参照してください。 AWS Tools for PowerShell

Get-DSConditionalForwarder

次の例は、Get-DSConditionalForwarder を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された Directory-id のすべての設定済み条件付きフォワーダーを取得します。

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

出力:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- API の詳細については、「 コマンドレットリファレンスDescribeConditionalForwarders」の「」を参照してください。 AWS Tools for PowerShell

Get-DSDirectory

次の例は、Get-DSDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、このアカウントに属するディレクトリに関する情報を取得します。

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeDirectories」の「」を参照してください。 AWS Tools for PowerShell

Get-DSDirectoryLimit

次の例は、Get-DSDirectoryLimit を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、us-east-1 リージョンのディレクトリ制限情報を支払います。

```
Get-DSDirectoryLimit -Region us-east-1
```

出力:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
```

```
ConnectedDirectoriesLimit        : 10
```

- API の詳細については、「 コマンドレットリファレンスGetDirectoryLimits」の「」を参照してください。 AWS Tools for PowerShell

Get-DSDomainControllerList

次の例は、Get-DSDomainControllerList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、前述の directory-id に対して起動されたドメインコントローラーの詳細なリストを取得します。

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeDomainControllers」の「」を参照してください。 AWS Tools for PowerShell

Get-DSEventTopic

次の例は、Get-DSEventTopic を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、ディレクトリのステータスが変更されたときに通知するように設定された SNS トピックの情報を表示します。

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

出力:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- API の詳細については、「 コマンドレットリファレンスDescribeEventTopics」の「」を参照してください。 AWS Tools for PowerShell

Get-DSIpRouteList

次の例は、Get-DSIpRouteList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、ディレクトリ IP ルーティングで設定されたパブリック IP アドレスブロックを取得します。

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

出力:

```
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 :
```

- API の詳細については、「 コマンドレットリファレンスListIpRoutes」の「」を参照してください。 AWS Tools for PowerShell

Get-DSLDAPSSetting

次の例は、Get-DSLDAPSSetting を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したディレクトリの LDAP セキュリティのステータスについて説明します。

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

出力:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」のDescribeLDAPSSettings」を参照してください。

Get-DSLogSubscriptionList

次の例は、Get-DSLogSubscriptionList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された directory-id のログサブスクリプション情報を取得します。

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

出力:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 --------------------------
```

```
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- API の詳細については、「 コマンドレットリファレンスListLogSubscriptions」の「」を参照してください。 AWS Tools for PowerShell

Get-DSResourceTag

次の例は、Get-DSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定されたディレクトリのすべてのタグを取得します。

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

出力:

```
Key   Value
---   -----
myTag myTagValue
```

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

Get-DSSchemaExtension

次の例は、Get-DSSchemaExtension を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Microsoft AD Directory に適用されるすべてのスキーマ拡張を一覧表示します。

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListSchemaExtensions」の「」を参照してください。 AWS Tools for PowerShell

Get-DSSharedDirectory

次の例は、Get-DSSharedDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウントの共有ディレクトリを取得します。

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeSharedDirectories」の「」を参照してください。 AWS Tools for PowerShell

Get-DSSnapshot

次の例は、Get-DSSnapshot を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、このアカウントに属する指定されたディレクトリスナップショットに関する情報を取得します。

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeSnapshots」の「」を参照してください。 AWS Tools for PowerShell

Get-DSSnapshotLimit

次の例は、Get-DSSnapshotLimit を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定されたディレクトリの手動スナップショット制限を取得します。

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

出力:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- API の詳細については、「 コマンドレットリファレンスGetSnapshotLimits」の「」を参照してください。 AWS Tools for PowerShell

Get-DSTrust

次の例は、Get-DSTrust を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された directory-id に対して作成された信頼関係の情報を取得します。

```
Get-DSTrust -DirectoryId d-123456abcd
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrusts」の「」を参照してください。 AWS Tools for PowerShell

New-DSAlias

次の例は、New-DSAlias を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドはディレクトリのエイリアスを作成し、指定された directory-id にエイリアスを割り当てます。

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

出力:

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスCreateAlias」の「」を参照してください。 AWS Tools for PowerShell

New-DSComputer

次の例は、New-DSComputer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しい Active Directory コンピュータオブジェクトを作成します。

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

出力:

```
ComputerAttributes                  ComputerId
 ComputerName
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- API の詳細については、「 コマンドレットリファレンスCreateComputer」の「」を参照してください。 AWS Tools for PowerShell

New-DSConditionalForwarder

次の例は、New-DSConditionalForwarder を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AWS Directory-id に条件付きフォワーダーを作成します。

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- API の詳細については、「 コマンドレットリファレンスCreateConditionalForwarder」の「」を参照してください。 AWS Tools for PowerShell

New-DSDirectory

次の例は、New-DSDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しい Simple AD ディレクトリを作成します。

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API の詳細については、「 コマンドレットリファレンスCreateDirectory」の「」を参照してください。 AWS Tools for PowerShell

New-DSLogSubscription

次の例は、New-DSLogSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 内の指定された Amazon ロググループにリアルタイムの Directory Service ドメインコントローラーセキュリティ CloudWatch ログを転送するサブスクリプションを作成します AWS アカウント。

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- API の詳細については、「 コマンドレットリファレンスCreateLogSubscription」の「」を参照してください。 AWS Tools for PowerShell

New-DSMicrosoftAD

次の例は、New-DSMicrosoftAD を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 に新しい Microsoft AD Directory を作成します AWS クラウド。

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API の詳細については、「 コマンドレットリファレンス」のCreateMicrosoft「AD」を参照してください。 AWS Tools for PowerShell

New-DSSnapshot

次の例は、New-DSSnapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例ではディレクトリスナップショットを作成します。

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスCreateSnapshot」の「」を参照してください。 AWS Tools for PowerShell

New-DSTrust

次の例は、New-DSTrust を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS Managed Microsoft AD ディレクトリと既存のオンプレミス Microsoft Active Directory との間に双方向のフォレスト全体の信頼を作成します。

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

出力:

```
t-9067157123
```

- API の詳細については、「 コマンドレットリファレンスCreateTrust」の「」を参照してください。 AWS Tools for PowerShell

Register-DSCertificate

次の例は、Register-DSCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、セキュリティで保護された LDAP 接続の証明書を登録します。

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

出力:

```
c-906731e350
```

- API の詳細については、「 コマンドレットリファレンスRegisterCertificate」の「」を参照してください。 AWS Tools for PowerShell

Register-DSEventTopic

次の例は、Register-DSEventTopic を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ディレクトリをパブリッシャーとして SNS トピックに関連付けます。

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API の詳細については、「 コマンドレットリファレンスRegisterEventTopic」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSConditionalForwarder

次の例は、Remove-DSConditionalForwarder を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Direcotry AWS 用に設定された条件付きフォワーダーを削除します。

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- API の詳細については、「 コマンドレットリファレンスDeleteConditionalForwarder」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSDirectory

次の例は、Remove-DSDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS ディレクトリサービスディレクトリ (Simple AD/Microsoft AD/AD Connector) を削除します。

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスDeleteDirectory」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSIpRoute

次の例は、Remove-DSIpRoute を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された IP を Directory-id の設定済み IP ルートから削除します。

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- API の詳細については、「 コマンドレットリファレンスRemoveIpRoutes」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSLogSubscription

次の例は、Remove-DSLogSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された Directory-id のログサブスクリプションを削除します

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- API の詳細については、「 コマンドレットリファレンスDeleteLogSubscription」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSResourceTag

次の例は、Remove-DSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された Directory-id に割り当てられたリソースタグを削除します

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- API の詳細については、「 コマンドレットリファレンスRemoveTagsFromResource」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSSnapshot

次の例は、Remove-DSSnapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、手動で作成されたスナップショットを削除します。

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- API の詳細については、「 コマンドレットリファレンスDeleteSnapshot」の「」を参照してください。 AWS Tools for PowerShell

Remove-DSTrust

次の例は、Remove-DSTrust を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS Managed AD Directory と外部ドメイン間の既存の信頼リレーションチップを削除します。

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

出力:

```
t-9067157123
```

- API の詳細については、「 コマンドレットリファレンスDeleteTrust」の「」を参照してください。 AWS Tools for PowerShell

Reset-DSUserPassword

次の例は、Reset-DSUserPassword を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS Managed microsoft AD または Simple AD Directory で ADUser という名前の Active Direcotry ユーザーのパスワードをリセットします。

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- API の詳細については、「 コマンドレットリファレンスResetUserPassword」の「」を参照してください。 AWS Tools for PowerShell

Restore-DSFromSnapshot

次の例は、Restore-DSFromSnapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、既存のディレクトリスナップショットを使用してディレクトリを復元します。

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- API の詳細については、「 コマンドレットリファレンスRestoreFromSnapshot」の「」を参照してください。 AWS Tools for PowerShell

Set-DSDomainControllerCount

次の例は、Set-DSDomainControllerCount を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された directory-id のドメインコントローラーの数を 3 に設定します。

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- API の詳細については、「 コマンドレットリファレンスUpdateNumberOfDomainControllers」の「」を参照してください。 AWS Tools for PowerShell

`Start-DSSchemaExtension`

次の例は、`Start-DSSchemaExtension` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、スキーマ拡張を Microsoft AD ディレクトリに適用します。

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

出力:

```
e-9067306643
```

- API の詳細については、「 コマンドレットリファレンスStartSchemaExtension」の「」を参照してください。 AWS Tools for PowerShell

`Stop-DSSchemaExtension`

次の例は、`Stop-DSSchemaExtension` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Microsoft AD ディレクトリへの進行中のスキーマ拡張をキャンセルします。

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- API の詳細については、「 コマンドレットリファレンスCancelSchemaExtension」の「」を参照してください。 AWS Tools for PowerShell

Unregister-DSCertificate

次の例は、Unregister-DSCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、セキュリティで保護された LDAP 接続に登録された証明書をシステムから削除します。

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- API の詳細については、「 コマンドレットリファレンスDeregisterCertificate」の「」を参照してください。 AWS Tools for PowerShell

Unregister-DSEventTopic

次の例は、Unregister-DSEventTopic を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された SNS トピックのパブリッシャーとして指定されたディレクトリを削除します。

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API の詳細については、「 コマンドレットリファレンスDeregisterEventTopic」の「」を参照してください。 AWS Tools for PowerShell

Update-DSConditionalForwarder

次の例は、Update-DSConditionalForwarder を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS ディレクトリ用に設定された条件付きフォワーダーを更新します。

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- API の詳細については、「 コマンドレットリファレンスUpdateConditionalForwarder」の「」を参照してください。 AWS Tools for PowerShell

Update-DSRadius

次の例は、Update-DSRadius を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、AD Connector または Microsoft AD ディレクトリの RADIUS サーバー情報を更新します。

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- API の詳細については、「 コマンドレットリファレンスUpdateRadius」の「」を参照してください。 AWS Tools for PowerShell

Update-DSTrust

次の例は、Update-DSTrust を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された trust-id の SelectiveAuth パラメータを Disabled から Enabled に更新します。

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

出力:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- API の詳細については、「 コマンドレットリファレンスUpdateTrust」の「」を参照してください。 AWS Tools for PowerShell

AWS DMS Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS DMS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

New-DMSReplicationTask

次の例は、New-DMSReplicationTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 CdcStartTime の代わりに を使用する新しい AWS Database Migration Service レプリケーションタスクを作成します CdcStartPosition。 MigrationType はfull-load-and-cdc「」に設定されています。つまり、ターゲットテーブルは空である必要があります。新しいタスクには、Stage のキーと Test のキー値を持つタグが付けられます。このコマンドレットで使用される値の詳細については、 Database Migration Service ユーザーガイドの「タスクの作成 (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) AWS 」を参照してください。

```
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"
```

- API の詳細については、「 コマンドレットリファレンスCreateReplicationTask」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した DynamoDB の例 PowerShell

次のコード例は、DynamoDB AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Add-DDBIndexSchema

次の例は、Add-DDBIndexSchema を使用する方法を説明しています。

のツール PowerShell

例 1: オブジェクトをパイプラインに書き込む前に、空の TableSchema オブジェクト TableSchema を作成し、新しいローカルセカンダリインデックス定義を追加します。

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

出力:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

例 2: TableSchema オブジェクトをパイプラインに書き戻す前に、指定された TableSchema オブジェクトに新しいローカルセカンダリインデックス定義を追加します。 TableSchema オブジェクトは、-Schema パラメータを使用して指定することもできます。

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

出力:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

- API の詳細については、「 コマンドレットリファレンス」の「Add-DDBIndexSchema」を参照してください。 AWS Tools for PowerShell

Add-DDBKeySchema

次の例は、Add-DDBKeySchema を使用する方法を説明しています。

のツール PowerShell

例 1: オブジェクトを TableSchema パイプラインに書き込む前に、空の TableSchema オブジェクトを作成し、指定されたキーデータを使用してキーと属性定義エントリを追加します。キータイプはデフォルトで「HASH」と宣言されます。範囲キーを宣言するには、値「RANGE」の -KeyType paameter を使用します。

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

出力:

```
AttributeSchema                                 KeySchema
    LocalSecondaryIndexSchema
---------------                                 ---------
    -------------------------
{ForumName}                                     {ForumName}
    {}
```

例 2: TableSchema オブジェクトをパイプラインに書き込む前に、指定された TableSchema オブジェクトに新しいキーと属性定義エントリを追加します。キータイプはデフォルトで「HASH」と宣言されます。範囲キーを宣言するには、値「RANGE」の -KeyType paameter を使用します。 TableSchema オブジェクトは、-Schema パラメータを使用して指定することもできます。

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

出力:

```
AttributeSchema                                 KeySchema
    LocalSecondaryIndexSchema
---------------                                 ---------
    -------------------------
{ForumName}                                     {ForumName}
    {}
```

- API の詳細については、「 コマンドレットリファレンス」の「Add-DDBKeySchema」を参照してください。 AWS Tools for PowerShell

ConvertFrom-DDBItem

次の例は、ConvertFrom-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: ConvertFrom-DDBItem は、Get-DDBItem の結果を DynamoDB AttributeValues のハッシュテーブルから文字列や double などの一般的なタイプのハッシュテーブルに変換するために使用されます。

```
@{
```

```
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

出力:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API の詳細については、「 コマンドレットリファレンス」のConvertFrom「-DDBItem」を参照してください。 AWS Tools for PowerShell

`ConvertTo-DDBItem`

次の例は、ConvertTo-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: ハッシュテーブルを DynamoDB 属性値のディクショナリに変換する例。

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

例 2: ハッシュテーブルを DynamoDB 属性値のディクショナリに変換する例。

```
@{
    MyMap        = @{
```

```
        MyString = 'my string'
    }
    MyStringSet  = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet  = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string')))
    )
    MyList1      = @('my', 'string')
    MyList2      = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3      = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンス」のConvertTo「-DDBItem」を参照してください。 AWS Tools for PowerShell

Get-DDBBatchItem

次の例は、Get-DDBBatchItem を使用する方法を説明しています。

のツール PowerShell

例 1: DynamoDB テーブル「Music」と SongTitle 「Songs」から「Somewhere Down The Road」を含む項目を取得します。

```
$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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスBatchGetItem」の「」を参照してください。 AWS Tools for PowerShell

Get-DDBItem

次の例は、Get-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: パーティションキー SongTitle とソートキー Artist を持つ DynamoDB 項目を返します。

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

出力:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- API の詳細については、「 コマンドレットリファレンスGetItem」の「」を参照してください。 AWS Tools for PowerShell

Get-DDBTable

次の例は、Get-DDBTable を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたテーブルの詳細を返します。

```
Get-DDBTable -TableName "myTable"
```

- API の詳細については、「 コマンドレットリファレンスDescribeTable」の「」を参照してください。 AWS Tools for PowerShell

Get-DDBTableList

次の例は、Get-DDBTableList を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのテーブルの詳細を返し、サービスが他にテーブルがないことを知らせるまで自動で繰り返します。

```
Get-DDBTableList
```

例 2: 呼び出しごとに 10 個のテーブルの詳細を返し、サービスが他にテーブルがないことを知らせるまで手動で繰り返します。

```
$nextToken = $null
do {
  Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
  $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスListTables」の「」を参照してください。 AWS Tools for PowerShell

`Invoke-DDBQuery`

次の例は、Invoke-DDBQuery を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された SongTitle と Artist を持つ DynamoDB 項目を返すクエリを呼び出します。

```
$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
```

出力:

```
Name                           Value
----                           -----
```

```
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference」の「Query」を参照してください。

Invoke-DDBScan

次の例は、Invoke-DDBScan を使用する方法を説明しています。

のツール PowerShell

例 1: Music テーブルのすべての項目を返します。

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

出力:

```
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
```

例 2: 9 CriticRating 以上の Music テーブルの項目を返します。

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
```

```
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

出力:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference」の「Scan」を参照してください。

New-DDBTable

次の例は、New-DDBTable を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ForumName「」(キータイプハッシュ) と「Subject」(キータイプ範囲) で構成されるプライマリキーを持つ Thread という名前のテーブルを作成します。テーブルの作成に使用したスキーマは、図のように各 cmdlet にパイプ処理するか、-Schema パラメータを使用して指定できます。

```
$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
```

出力:

```
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  : {}
```

例 2: この例では、ForumName「」(キータイプハッシュ) と「Subject」(キータイプ範囲) で構成されるプライマリキーを持つ Thread という名前のテーブルを作成します。ローカルセカンダリインデックスも定義されます。ローカルセカンダリインデックスのキーは、テーブル () のプライマリハッシュキーから自動的に設定されますForumName。テーブルの作成に使用したスキーマは、図のように各 cmdlet にパイプ処理するか、-Schema パラメータを使用して指定できます。

```
$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
```

出力:

```
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}
```

例 3: この例では、単一のパイプラインを使用して、ForumName「」(キータイプハッシュ) と「Subject」(キータイプ範囲) で構成されるプライマリキーとローカルセカンダリインデックスを持つ Thread という名前のテーブルを作成する方法を示します。Add-DDB KeySchema と Add-DDB は、パイプラインまたは -Schema パラメータからオブジェクトが指定されていない場合に、新しい TableSchema オブジェクトIndexSchema を作成します。

```
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
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスCreateTable」の「」を参照してください。 AWS Tools for PowerShell

New-DDBTableSchema

次の例は、New-DDBTableSchema を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい Amazon DynamoDB テーブルの作成に使用するキーとインデックスの定義を受け入れる準備ができている空の TableSchema オブジェクトを作成します。返されたオブジェクトは、Add-DDB、Add-DDBKeySchema、IndexSchema および New-DDBTable コマンドレットにパイプするか、各コマンドレットの -Schema パラメータを使用して渡すことができます。

```
New-DDBTableSchema
```

出力:

```
AttributeSchema                          KeySchema
    LocalSecondaryIndexSchema
---------------                          ---------
    -------------------------
```

```
{}                                                  {}
    {}
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」の「New-DDBTableSchema」を参照してください。

Remove-DDBItem

次の例は、Remove-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたキーと一致する DynamoDB 項目を削除します。

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- API の詳細については、「 コマンドレットリファレンスDeleteItem」の「」を参照してください。 AWS Tools for PowerShell

Remove-DDBTable

次の例は、Remove-DDBTable を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたテーブルを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-DDBTable -TableName "myTable"
```

例 2: 指定されたテーブルを削除します。操作を続行する前に確認画面は表示されません。

```
Remove-DDBTable -TableName "myTable" -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteTable」の「」を参照してください。 AWS Tools for PowerShell

Set-DDBBatchItem

次の例は、Set-DDBBatchItem を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい項目を作成する、または既存の項目を「Music 」DynamoDB テーブルおよび「Songs」DynamoDB テーブルの新しい項目で置き換えます。

```
$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
```

出力:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- API の詳細については、「 コマンドレットリファレンスBatchWriteItem」の「」を参照してください。 AWS Tools for PowerShell

Set-DDBItem

次の例は、Set-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい項目を作成する、または既存の項目を新しい項目で置き換えます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスPutItem」の「」を参照してください。 AWS Tools for PowerShell

Update-DDBItem

次の例は、Update-DDBItem を使用する方法を説明しています。

のツール PowerShell

例 1: パーティションキー SongTitle とソートキー Artist を使用して、DynamoDB 項目で genre 属性を「Rap」に設定します。

```
$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
```

出力:

```
Name                           Value
----                           -----
Genre                          Rap
```

- API の詳細については、「 コマンドレットリファレンスUpdateItem」の「」を参照してください。 AWS Tools for PowerShell

Update-DDBTable

次の例は、Update-DDBTable を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたテーブルのプロビジョンされたスループットを更新します。

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- API の詳細については、「 コマンドレットリファレンスUpdateTable」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon EC2 の例 PowerShell

次のコード例は、Amazon EC2 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Add-EC2CapacityReservation

次の例は、Add-EC2CapacityReservation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された属性を使用して新しいキャパシティ予約を作成します。

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateCapacityReservation」の「」を参照してください。 AWS Tools for PowerShell

Add-EC2InternetGateway

次の例は、Add-EC2InternetGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインターネットゲートウェイを指定された VPC にアタッチします。

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

例 2: この例では、VPC とインターネットゲートウェイを作成し、インターネットゲートウェイを VPC にアタッチします。

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- API の詳細については、「 コマンドレットリファレンスAttachInternetGateway」の「」を参照してください。 AWS Tools for PowerShell

Add-EC2NetworkInterface

次の例は、Add-EC2NetworkInterface を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスを指定されたインスタンスにアタッチします。

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

出力:

```
eni-attach-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスAttachNetworkInterface」の「」を参照してください。 AWS Tools for PowerShell

Add-EC2Volume

次の例は、Add-EC2Volume を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームを指定されたインスタンスにアタッチし、指定されたデバイス名で公開します。

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

出力:

```
AttachTime          : 12/22/2015 1:53:58 AM
```

```
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- API の詳細については、「 コマンドレットリファレンスAttachVolume」の「」を参照してください。 AWS Tools for PowerShell

Add-EC2VpnGateway

次の例は、Add-EC2VpnGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイを指定された VPC にアタッチします。

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

出力:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスAttachVpnGateway」の「」を参照してください。 AWS Tools for PowerShell

Approve-EC2VpcPeeringConnection

次の例は、Approve-EC2VpcPeeringConnection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リクエストされた VpcPeeringConnectionId pcx-1dfad234b56ff78be を承認します。

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスAcceptVpcPeeringConnection」の「」を参照してください。 AWS Tools for PowerShell

Confirm-EC2ProductInstance

次の例は、Confirm-EC2ProductInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された製品コードが指定されたインスタンスに関連付けられているかどうかを決定します。

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- API の詳細については、「 コマンドレットリファレンスConfirmProductInstance」の「」を参照してください。 AWS Tools for PowerShell

Copy-EC2Image

次の例は、Copy-EC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「EU (アイルランド) 」リージョンの指定された AMI を「米国西部 (オレゴン) 」リージョンにコピーします。-Region が指定されていない場合、現在のデフォルトリージョンが送信先リージョンとして使用されます。

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2
 -Name "Copy of ami-12345678"
```

出力:

```
ami-87654321
```

- API の詳細については、「 コマンドレットリファレンスCopyImage」の「」を参照してください。 AWS Tools for PowerShell

Copy-EC2Snapshot

次の例は、Copy-EC2Snapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットを欧州 (アイルランド) リージョンから米国西部 (オレゴン) リージョンにコピーします。

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-
west-2
```

例 2: デフォルトのリージョンを設定し、リージョンパラメータを省略すると、デフォルトの送信先リージョンがデフォルトのリージョンになります。

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- API の詳細については、「 コマンドレットリファレンスCopySnapshot」の「」を参照してください。 AWS Tools for PowerShell

Deny-EC2VpcPeeringConnection

次の例は、Deny-EC2VpcPeeringConnection を使用する方法を説明しています。

のツール PowerShell

例 1: 上記の例では、リクエスト ID pcx-01a2b3ce45fe67eb8 の VpcPeering リクエストを拒否します。

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API の詳細については、「 コマンドレットリファレンスRejectVpcPeeringConnection」の「」を参照してください。 AWS Tools for PowerShell

`Disable-EC2VgwRoutePropagation`

次の例は、`Disable-EC2VgwRoutePropagation` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VGW が指定されたルーティングテーブルにルートを自動的に伝達することを無効にします。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDisableVgwRoutePropagation」の「」を参照してください。 AWS Tools for PowerShell

`Disable-EC2VpcClassicLink`

次の例は、`Disable-EC2VpcClassicLink` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、vpc-01eEC2VpcClassicLink 2 を無効にします。True または False を返します。

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API の詳細については、「 コマンドレットリファレンスDisableVpcClassicLink」の「」を参照してください。 AWS Tools for PowerShell

`Disable-EC2VpcClassicLinkDnsSupport`

次の例は、`Disable-EC2VpcClassicLinkDnsSupport` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、vpc-0b12d3456a7e8910d の ClassicLink DNS サポートを無効にします。

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API の詳細については、「 コマンドレットリファレンスDisableVpcClassicLinkDnsSupport」の「」を参照してください。 AWS Tools for PowerShell

`Dismount-EC2InternetGateway`

次の例は、`Dismount-EC2InternetGateway` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC から指定されたインターネットゲートウェイをデタッチします。

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスDetachInternetGateway」の「」を参照してください。 AWS Tools for PowerShell

`Dismount-EC2NetworkInterface`

次の例は、`Dismount-EC2NetworkInterface` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ネットワークインターフェイスとインスタンス間の指定されたアタッチメントを削除します。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API の詳細については、「 コマンドレットリファレンスDetachNetworkInterface」の「」を参照してください。 AWS Tools for PowerShell

`Dismount-EC2Volume`

次の例は、`Dismount-EC2Volume` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームをデタッチします。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

出力:

```
AttachTime          : 12/22/2015 1:53:58 AM
```

```
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

例 2: インスタンス ID とデバイス名を指定して、正しいボリュームをデタッチするようにすることもできます。

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- API の詳細については、「 コマンドレットリファレンスDetachVolume」の「」を参照してください。 AWS Tools for PowerShell

`Dismount-EC2VpnGateway`

次の例は、`Dismount-EC2VpnGateway` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイを指定された VPC からデタッチします。

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスDetachVpnGateway」の「」を参照してください。 AWS Tools for PowerShell

`Edit-EC2CapacityReservation`

次の例は、`Edit-EC2CapacityReservation` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタン数を 1 に変更して CapacityReservationId cr-0c1f2345db6f7cdba を変更します。

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

出力:

```
True
```

- API の詳細については、「 コマンドレットリファレンスModifyCapacityReservation」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2Host

次の例は、Edit-EC2Host を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、専用ホスト h-01e23f4cd567890f3 AutoPlacement の設定をオフに変更します。

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

出力:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- API の詳細については、「 コマンドレットリファレンスModifyHosts」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2IdFormat

次の例は、Edit-EC2IdFormat を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースタイプの長い ID 形式を有効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

例 2: この例では、指定されたリソースタイプの長い ID 形式を無効にします。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API の詳細については、「 コマンドレットリファレンスModifyIdFormat」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2ImageAttribute

次の例は、Edit-EC2ImageAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AMI の説明を更新します。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

例 2: この例では、AMI を公開します (例えば、 がそれを AWS アカウント 使用できる)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

例 3: この例では、AMI をプライベートにします (例えば、所有者として自分だけが使用できるように)。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

例 4: この例では、指定された に起動アクセス許可を付与します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

例 5: この例では、指定された から起動許可を削除します AWS アカウント。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- API の詳細については、「 コマンドレットリファレンスModifyImageAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2InstanceAttribute

次の例は、Edit-EC2InstanceAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスのインスタンスタイプを変更します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

例 2: この例では、単一ルート I/O 仮想化 (SR-IOV) ネットワークサポートパラメータ - の値として「simple」を指定することで、指定されたインスタンスの拡張ネットワーキングを有効にしますSriovNetSupport。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

例 3: この例では、指定したインスタンスのセキュリティグループを変更します。インスタンスは VPC 内にある必要があります。名前ではなく、各セキュリティグループの ID を指定する必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

例 4: この例では、指定したインスタンスの EBS I/O 最適化を有効にします。この機能は、すべてのインスタンスタイプで使用できるわけではありません。EBS 最適化インスタンスを使用する場合、追加料金が適用されます。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

例 5: この例では、指定したインスタンスの送信元/送信先チェックを有効にします。NAT インスタンスが NAT を実行するには、値が「false」である必要があります。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

例 6: この例では、指定したインスタンスの終了を無効にします。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

例 7: この例では、指定されたインスタンスを変更して、インスタンスからシャットダウンが開始されたときに終了します。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- API の詳細については、「 コマンドレットリファレンスModifyInstanceAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2InstanceCreditSpecification

次の例は、Edit-EC2InstanceCreditSpecification を使用する方法を説明しています。

のツール PowerShell

例 1: これにより、インスタンス i-01234567890abcdef の T2 無制限クレジットが有効になります。

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API の詳細については、「 コマンドレットリファレンスModifyInstanceCreditSpecification」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2NetworkInterfaceAttribute

次の例は、Edit-EC2NetworkInterfaceAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したネットワークインターフェイスを変更して、指定したアタッチメントが終了時に削除されるようにします。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

例 2: この例では、指定されたネットワークインターフェイスの説明を変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

例 3: この例では、指定されたネットワークインターフェイスのセキュリティグループを変更します。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

例 4: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックを無効にします。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- API の詳細については、「 コマンドレットリファレンスModifyNetworkInterfaceAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2ReservedInstance

次の例は、Edit-EC2ReservedInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリザーブドインスタンスのアベイラビリティーゾーン、インスタンス数、プラットフォームを変更します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスModifyReservedInstances」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2SnapshotAttribute

次の例は、Edit-EC2SnapshotAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 CreateVolumePermission 属性を設定して、指定されたスナップショットを公開します。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- API の詳細については、「 コマンドレットリファレンスModifySnapshotAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2SpotFleetRequest

次の例は、Edit-EC2SpotFleetRequest を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットフリートリクエストのターゲット容量を更新します。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

出力:

```
True
```

- API の詳細については、「 コマンドレットリファレンスModifySpotFleetRequest」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2SubnetAttribute

次の例は、Edit-EC2SubnetAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したサブネットのパブリック IP アドレス指定を有効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

例 2: この例では、指定されたサブネットのパブリック IP アドレス指定を無効にします。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API の詳細については、「 コマンドレットリファレンスModifySubnetAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2VolumeAttribute

次の例は、Edit-EC2VolumeAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームの指定された属性を変更します。ボリュームの I/O オペレーションは、データに一貫性がないために中断された後に自動的に再開されます。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API の詳細については、「 コマンドレットリファレンスModifyVolumeAttribute」の「」を参照してください。 AWS Tools for PowerShell

Edit-EC2VpcAttribute

次の例は、Edit-EC2VpcAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC の DNS ホスト名のサポートを有効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

例 2: この例では、指定した VPC の DNS ホスト名のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

例 3: この例では、指定した VPC の DNS 解決のサポートを有効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

例 4: この例では、指定した VPC の DNS 解決のサポートを無効にします。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API の詳細については、「 コマンドレットリファレンスModifyVpcAttribute」の「」を参照してください。 AWS Tools for PowerShell

Enable-EC2VgwRoutePropagation

次の例は、Enable-EC2VgwRoutePropagation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VGW が指定されたルーティングテーブルに自動的にルートを伝播できるようにします。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスEnableVgwRoutePropagation」の「」を参照してください。 AWS Tools for PowerShell

Enable-EC2VolumeIO

次の例は、Enable-EC2VolumeIO を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、I/O オペレーションが無効になっている場合に、指定されたボリュームの I/O オペレーションを有効にします。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API の詳細については、「 コマンドレットリファレンスEnableVolumeIo」の「」を参照してください。 AWS Tools for PowerShell

Enable-EC2VpcClassicLink

次の例は、Enable-EC2VpcClassicLink を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 の VPC vpc-0123456b789b0d12f を有効にします。 ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

出力:

```
True
```

- API の詳細については、「 コマンドレットリファレンスEnableVpcClassicLink」の「」を参照してください。 AWS Tools for PowerShell

`Enable-EC2VpcClassicLinkDnsSupport`

次の例は、`Enable-EC2VpcClassicLinkDnsSupport` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、vpc-0b12d3456a7e8910d が の DNS ホスト名解決をサポートできるようにします。 ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API の詳細については、「 コマンドレットリファレンスEnableVpcClassicLinkDnsSupport」の「」を参照してください。 AWS Tools for PowerShell

`Get-EC2AccountAttribute`

次の例は、`Get-EC2AccountAttribute` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョンの EC2-Classic および EC2-VPC にインスタンスを起動できるか、EC2-VPC にのみインスタンスを起動できるかについて説明します。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

出力:

```
AttributeValue
--------------
EC2
```

```
VPC
```

例 2: この例では、デフォルトの VPC について説明しています。リージョンにデフォルトの VPC がない場合は「none」です。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

出力:

```
AttributeValue
--------------
vpc-12345678
```

例 3: この例では、実行できるオンデマンドインスタンスの最大数について説明します。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

出力:

```
AttributeValue
--------------
20
```

- API の詳細については、「 コマンドレットリファレンスDescribeAccountAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Address

次の例は、Get-EC2Address を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスについて説明します。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

出力:

```
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
```

例 2: この例では、VPC 内のインスタンスの Elastic IP アドレスについて説明します。この構文には PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

例 3: この例では、EC2-Classic のインスタンスに指定された Elastic IP アドレスについて説明します。

```
Get-EC2Address -PublicIp 203.0.113.17
```

出力:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

例 4: この例では、EC2-Classic のインスタンスの Elastic IP アドレスについて説明します。この構文には PowerShell バージョン 3 以降が必要です。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

例 5: この例では、すべての Elastic IP アドレスについて説明します。

```
Get-EC2Address
```

例 6: この例では、フィルターで指定されたインスタンス ID のパブリック IP とプライベート IP を返します。

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

出力:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

例 7: この例では、割り当て ID、関連付け ID、インスタンス ID を持つすべての Elastic IPs を取得します。

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

出力:

```
InstanceId          AssociationId              AllocationId               PublicIp
----------          -------------              ------------               --------
                                               eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                               eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

例 8: この例では、タグキー 'Category' と値 'Prod' に一致する EC2 IP アドレスのリストを取得します。

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスDescribeAddresses」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2AvailabilityZone

次の例は、Get-EC2AvailabilityZone を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在のリージョンで使用できるアベイラビリティーゾーンについて説明します。

```
Get-EC2AvailabilityZone
```

出力:

```
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
```

例 2: この例では、障害状態にあるアベイラビリティーゾーンについて説明します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

例 3: PowerShell バージョン 2 では、New-Object を使用してフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- API の詳細については、「 コマンドレットリファレンスDescribeAvailabilityZones」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2BundleTask

次の例は、Get-EC2BundleTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたバンドルタスクについて説明します。

```
Get-EC2BundleTask -BundleId bun-12345678
```

例 2: この例では、状態が「complete」または「failed」のバンドルタスクについて説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API の詳細については、「 コマンドレットリファレンスDescribeBundleTasks」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2CapacityReservation

次の例は、Get-EC2CapacityReservation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョンの 1 つ以上のキャパシティ予約について説明します。

```
Get-EC2CapacityReservation -Region eu-west-1
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeCapacityReservations」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ConsoleOutput

次の例は、Get-EC2ConsoleOutput を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Linux インスタンスのコンソール出力を取得します。コンソール出力はエンコードされます。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

出力:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

例 2: この例では、エンコードされたコンソール出力を変数に保存し、それをデコードします。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API の詳細については、「 コマンドレットリファレンスGetConsoleOutput」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2CustomerGateway

次の例は、Get-EC2CustomerGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたカスタマーゲートウェイについて説明します。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

例 2: この例では、状態が保留中または利用可能なカスタマーゲートウェイについて説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

例 3: この例では、すべてのカスタマーゲートウェイについて説明します。

```
Get-EC2CustomerGateway
```

- API の詳細については、「 コマンドレットリファレンスDescribeCustomerGateways」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2DhcpOption

次の例は、Get-EC2DhcpOption を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、DHCP オプションセットを一覧表示します。

```
Get-EC2DhcpOption
```

出力:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

例 2: この例では、指定された DHCP オプションセットの設定の詳細を取得します。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

出力:

```
Key                  Values
---                  ------
domain-name          {abc.local}
domain-name-servers  {10.0.0.101, 10.0.0.102}
```

- API の詳細については、「 コマンドレットリファレンスDescribeDhcpOptions」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2FlowLog

次の例は、Get-EC2FlowLog を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ログ送信先タイプが 's3' の 1 つ以上のフローログについて説明します。

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

出力:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- API の詳細については、「 コマンドレットリファレンスDescribeFlowLogs」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Host

次の例は、Get-EC2Host を使用する方法を説明しています。

のツール PowerShell

例 1: この例では EC2 ホストの詳細を返します。

```
Get-EC2Host
```

出力:

```
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              : {}
```

例 2: この例では、ホスト h-01e23f4cd567899f1 AvailableInstanceCapacity の をクエリします。

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

出力:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- API の詳細については、「 コマンドレットリファレンスDescribeHosts」の「」を参照してください。 AWS Tools for PowerShell

`Get-EC2HostReservationOffering`

次の例は、`Get-EC2HostReservationOffering` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたフィルター「instance-family」で購入できる Dedicated Host 予約について説明します。ここで、 PaymentOption はNoUpfront「」です。

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeHostReservationOfferings」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2HostReservationPurchasePreview

次の例は、Get-EC2HostReservationPurchasePreview を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Dedicated Host h-01e23f4cd567890f1 の設定と一致する予約購入をプレビューします。

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

出力:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- API の詳細については、「 コマンドレットリファレンスGetHostReservationPurchasePreview」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2IdFormat

次の例は、Get-EC2IdFormat を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースタイプの ID 形式について説明します。

```
Get-EC2IdFormat -Resource instance
```

出力:

```
Resource       UseLongIds
--------       ----------
instance       False
```

例 2: この例では、長い ID をサポートするすべてのリソースタイプの IDs形式について説明します。

```
Get-EC2IdFormat
```

出力:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- API の詳細については、「 コマンドレットリファレンスDescribeIdFormat」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2IdentityIdFormat

次の例は、Get-EC2IdentityIdFormat を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロールのリソース「image」の ID 形式を返します。

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

出力:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- API の詳細については、「 コマンドレットリファレンスDescribeIdentityIdFormat」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Image

次の例は、Get-EC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AMI について説明します。

```
Get-EC2Image -ImageId ami-12345678
```

出力:

```
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
```

例 2: この例では、所有している AMIsについて説明します。

```
Get-EC2Image -owner self
```

例 3: この例では、Microsoft Windows Server を実行するパブリック AMIs について説明します。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

例 4: この例では、AMIs について説明します。

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

- API の詳細については、「 コマンドレットリファレンスDescribeImages」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ImageAttribute

次の例は、Get-EC2ImageAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AMI の説明を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

出力:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

例 2: この例では、指定された AMI の起動許可を取得します。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

出力:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
```

```
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

例 3: この例では、拡張ネットワーキングが有効になっているかどうかをテストします。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

出力:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- API の詳細については、「 コマンドレットリファレンスDescribeImageAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ImageByName

次の例は、Get-EC2ImageByName を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在サポートされているフィルター名の完全なセットについて説明します。

```
Get-EC2ImageByName
```

出力:

```
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
```

例 2: この例では、指定された AMI について説明します。このコマンドを使用して AMI を検索すると、毎月最新の更新を含む新しい Windows AMIs が AWS リリースされるため便利です。指定したフィルターの現在の AMI を使用してインスタンスを起動New-EC2Instanceするには、ImageId「」を指定します。

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGet-EC2ImageByName」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ImportImageTask

次の例は、Get-EC2ImportImageTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたイメージのインポートタスクについて説明します。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

出力:

```
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   :
```

例 2: この例では、すべてのイメージのインポートタスクについて説明します。

```
Get-EC2ImportImageTask
```

出力:

```
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   :
```

- API の詳細については、「 コマンドレットリファレンスDescribeImportImageTasks」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ImportSnapshotTask

次の例は、Get-EC2ImportSnapshotTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットのインポートタスクについて説明します。

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

出力:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

例 2: この例では、スナップショットのすべてのインポートタスクについて説明します。

```
Get-EC2ImportSnapshotTask
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeImportSnapshotTasks」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Instance

次の例は、Get-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスについて説明します。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

出力:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
```

```
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

例 2: この例では、現在のリージョンのすべてのインスタンスを予約別にグループ化して説明します。インスタンスの詳細を表示するには、各予約オブジェクト内のインスタンスコレクションを展開します。

```
Get-EC2Instance
```

出力:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

例 3: この例は、フィルターを使用して VPC の特定のサブネット内の EC2 インスタンスをクエリする方法を示しています。

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2InstanceAttribute

次の例は、Get-EC2InstanceAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスのインスタンスタイプについて説明します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

出力:

```
InstanceType                    : t2.micro
```

例 2: この例では、指定したインスタンスで拡張ネットワーキングが有効になっているかどうかを示します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

出力:

```
SriovNetSupport                 : simple
```

例 3: この例では、指定したインスタンスのセキュリティグループについて説明します。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

出力:

```
GroupId
-------
```

```
sg-12345678
sg-45678901
```

例 4: この例では、指定したインスタンスで EBS 最適化が有効になっているかどうかを示します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

出力:

```
EbsOptimized                     : False
```

例 5: この例では、指定されたインスタンスのdisableApiTermination「」属性について説明します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

出力:

```
DisableApiTermination            : False
```

例 6: この例では、指定されたインスタンスのinstanceInitiatedShutdown「Behavior」属性について説明します。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

出力:

```
InstanceInitiatedShutdownBehavior : stop
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstanceAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2InstanceMetadata

次の例は、Get-EC2InstanceMetadata を使用する方法を説明しています。

のツール PowerShell

例 1: クエリできるインスタンスメタデータの利用可能なカテゴリを一覧表示します。

```
Get-EC2InstanceMetadata -ListCategory
```

出力:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

例 2: インスタンスの起動に使用された Amazon マシンイメージ (AMI) の ID を返します。

```
Get-EC2InstanceMetadata -Category AmiId
```

出力:

```
ami-b2e756ca
```

例 3: この例では、インスタンスの JSON 形式の ID ドキュメントをクエリします。

```
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"
}
```

例 4: この例では、パスクエリを使用してインスタンスのネットワークインターフェイス macs を取得します。

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

出力:

```
02:80:7f:ef:4c:e0/
```

例 5: インスタンスに関連付けられた IAM ロールがある場合、 は、インスタンス LastUpdated の日付、 など InstanceProfileArn、インスタンスプロファイルが最後に更新された時刻に関する情報を返します InstanceProfileId。

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

出力:

```
{
  "Code" : "Success",
```

```
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- API の詳細については、「 コマンドレットリファレンスGet-EC2InstanceMetadata」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2InstanceStatus

次の例は、Get-EC2InstanceStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスのステータスについて説明します。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

出力:

```
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
```

出力:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

出力:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

出力:

```
Details           Status
-------           ------
{reachability}    ok
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstanceStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2InternetGateway

次の例は、Get-EC2InternetGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインターネットゲートウェイについて説明します。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

例 2: この例では、すべてのインターネットゲートウェイについて説明します。

```
Get-EC2InternetGateway
```

出力:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- API の詳細については、「 コマンドレットリファレンスDescribeInternetGateways」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2KeyPair

次の例は、Get-EC2KeyPair を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキーペアについて説明します。

```
Get-EC2KeyPair -KeyName my-key-pair
```

出力:

```
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
```

例 2: この例では、すべてのキーペアについて説明します。

```
Get-EC2KeyPair
```

- API の詳細については、「 コマンドレットリファレンスDescribeKeyPairs」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2NetworkAcl

次の例は、Get-EC2NetworkAcl を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL について説明します。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

例 2: この例では、指定されたネットワーク ACL のルールについて説明します。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

出力:

```
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
```

例 3: この例では、すべてのネットワーク ACLs について説明します。

```
Get-EC2NetworkAcl
```

- API の詳細については、「 コマンドレットリファレンスDescribeNetworkAcls」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2NetworkInterface

次の例は、Get-EC2NetworkInterface を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
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
```

例 2: この例では、すべてのネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterface
```

- API の詳細については、「 コマンドレットリファレンスDescribeNetworkInterfaces」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2NetworkInterfaceAttribute

次の例は、Get-EC2NetworkInterfaceAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

出力:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

例 2: この例では、指定されたネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

出力:

```
Description        : My description
```

例 3: この例では、指定されたネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

出力:

```
Groups             : {my-security-group}
```

例 4: この例では、指定されたネットワークインターフェイスについて説明します。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

出力:

```
SourceDestCheck    : True
```

- API の詳細については、「 コマンドレットリファレンスDescribeNetworkInterfaceAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2PasswordData

次の例は、Get-EC2PasswordData を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した Windows インスタンスの管理者アカウントに Amazon EC2 が割り当てたパスワードを復号します。pem ファイルを指定すると、-Decrypt スイッチの設定が自動的に引き受けられます。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

出力:

```
mYZ(PA9?C)Q
```

例 2: (Windows PowerShell のみ) インスタンスを検査し、インスタンスの起動に使用されるキーペアの名前を決定し、 AWS Toolkit for Visual Studio の設定ストアで対応するキーペアデータを見つけようとします。キーペアデータが見つかった場合、パスワードは復号化されます。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

出力:

```
mYZ(PA9?C)Q
```

例 3: インスタンスの暗号化されたパスワードデータを返します。

```
Get-EC2PasswordData -InstanceId i-12345678
```

出力:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API の詳細については、「 コマンドレットリファレンスGetPasswordData」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2PlacementGroup

次の例は、Get-EC2PlacementGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたプレイスメントグループについて説明します。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

出力:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- API の詳細については、「 コマンドレットリファレンスDescribePlacementGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2PrefixList

次の例は、Get-EC2PrefixList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョンのプレフィックスリスト AWS のサービス 形式で使用可能な を取得します。

```
Get-EC2PrefixList
```

出力:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- API の詳細については、「 コマンドレットリファレンスDescribePrefixLists」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Region

次の例は、Get-EC2Region を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、使用可能なリージョンについて説明します。

```
Get-EC2Region
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeRegions」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2RouteTable

次の例は、Get-EC2RouteTable を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのルートテーブルについて説明します。

```
Get-EC2RouteTable
```

出力:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
```

```
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

例 2: この例では、指定されたルートテーブルの詳細を返します。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

例 3: この例では、指定された VPC のルートテーブルについて説明します。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

出力:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDescribeRouteTables」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ScheduledInstance

次の例は、Get-EC2ScheduledInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスケジュールされたインスタンスについて説明します。

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

出力:

```
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
```

例 2: この例では、スケジュールされたすべてのインスタンスについて説明します。

```
Get-EC2ScheduledInstance
```

- API の詳細については、「 コマンドレットリファレンスDescribeScheduledInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2ScheduledInstanceAvailability

次の例は、Get-EC2ScheduledInstanceAvailability を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した日付から毎週日曜日に行われるスケジュールについて説明します。

```
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
```

出力:

```
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

...
```

例 2: 結果を絞り込むには、オペレーティングシステム、ネットワーク、インスタンスタイプなどの基準にフィルターを追加できます。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API の詳細については、「 コマンドレットリファレンスDescribeScheduledInstanceAvailability」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SecurityGroup

次の例は、Get-EC2SecurityGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VPC に指定されたセキュリティグループについて説明します。VPC に属するセキュリティグループを使用する場合は、名前 (-GroupId parameter) ではなくセキュリティグループ ID (-GroupName parameter) を使用してグループを参照する必要があります。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```
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
```

例 2: この例では、EC2-Classic に指定されたセキュリティグループについて説明します。EC2-Classic のセキュリティグループを使用する場合は、グループ名 (-GroupName パラメータ) またはグループ ID (-GroupId パラメータ) を使用してセキュリティグループを参照できます。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

出力:

```
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               :
```

例 3: この例では、vpc-0fc1ff23456b789eb のすべてのセキュリティグループを取得します。

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API の詳細については、「 コマンドレットリファレンスDescribeSecurityGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Snapshot

次の例は、Get-EC2Snapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットについて説明します。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
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
```

例 2: この例では、「Name」タグを持つスナップショットについて説明します。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

例 3: この例では、値 " の「Name」タグを持つスナップショットについて説明しますTestValue。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

例 4: この例では、すべてのスナップショットについて説明します。

```
Get-EC2Snapshot -Owner self
```

- API の詳細については、「 コマンドレットリファレンスDescribeSnapshots」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SnapshotAttribute

次の例は、Get-EC2SnapshotAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットの指定された属性について説明します。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

出力:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

例 2: この例では、指定されたスナップショットの指定された属性について説明します。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

出力:

```
Group    UserId
-----    ------
all
```

- API の詳細については、「 コマンドレットリファレンスDescribeSnapshotAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotDatafeedSubscription

次の例は、Get-EC2SpotDatafeedSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、スポットインスタンスのデータフィードについて説明します。

```
Get-EC2SpotDatafeedSubscription
```

出力:

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotDatafeedSubscription」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotFleetInstance

次の例は、Get-EC2SpotFleetInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットフリートリクエストに関連付けられたインスタンスについて説明します。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

出力:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotFleetInstances」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotFleetRequest

次の例は、Get-EC2SpotFleetRequest を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットフリートリクエストについて説明します。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

出力:

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime           : 12/26/2015 8:23:33 AM
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

例 2: この例では、すべてのスポットフリートリクエストについて説明します。

```
Get-EC2SpotFleetRequest
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotFleetRequests」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotFleetRequestHistory

次の例は、Get-EC2SpotFleetRequestHistory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットフリートリクエストの履歴について説明します。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

出力:

```
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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotFleetRequestHistory」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotInstanceRequest

次の例は、Get-EC2SpotInstanceRequest を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットインスタンスリクエストについて説明します。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

出力:

```
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
```

例 2: この例では、すべてのスポットインスタンスリクエストについて説明します。

```
Get-EC2SpotInstanceRequest
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotInstanceRequests」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2SpotPriceHistory

次の例は、Get-EC2SpotPriceHistory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスタイプとアベイラビリティーゾーンのスポット料金履歴の最後の 10 個のエントリを取得します。-AvailabilityZone parameter に指定された値は、コマンドレットの -Region パラメータ (例には示されていません) に指定されたリージョン値に対して有効であるか、シェルでデフォルトとして設定されている必要があることに注意してください。このコマンド例では、 環境で「us-west-2」のデフォルトリージョンが設定されていることを前提としています。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

出力:

```
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
...
```

- API の詳細については、「 コマンドレットリファレンスDescribeSpotPriceHistory」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Subnet

次の例は、Get-EC2Subnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたサブネットについて説明します。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
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
```

例 2: この例では、すべてのサブネットについて説明します。

```
Get-EC2Subnet
```

- API の詳細については、「 コマンドレットリファレンスDescribeSubnets」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Tag

次の例は、Get-EC2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リソースタイプ 'image' のタグを取得します。

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

出力:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

例 2: この例では、すべてのリソースのすべてのタグを取得し、リソースタイプ別にグループ化します。

```
Get-EC2Tag | Group-Object resourcetype
```

出力:

```
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}
```

例 3: この例では、指定したリージョンの値 'no' のタグ 'auto-delete' を持つすべてのリソースを表示します。

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

出力:

```
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
```

例 4: この例では、「no」値を持つタグ「auto-delete」を持つすべてのリソースを取得し、次のパイプでさらにフィルターして「instance」リソースタイプのみを解析し、最終的には値がインスタンス ID 自体であるインスタンスリソースごとにThisInstance「」タグを作成します。

```
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}}
```

例 5: この例では、すべてのインスタンスリソースと「Name」キーのタグを取得し、テーブル形式で表示します。

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

出力:

```
ResourceId          Name-Tag
----------          --------
```

```
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Volume

次の例は、Get-EC2Volume を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された EBS ボリュームについて説明します。

```
Get-EC2Volume -VolumeId vol-12345678
```

出力:

```
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
```

例 2: この例では、ステータスが「使用可能」の EBS ボリュームについて説明します。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

出力:

```
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
...
```

例 3: この例では、すべての EBS ボリュームについて説明します。

```
Get-EC2Volume
```

- API の詳細については、「 コマンドレットリファレンスDescribeVolumes」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VolumeAttribute

次の例は、Get-EC2VolumeAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームの指定された属性について説明します。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

出力:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- API の詳細については、「 コマンドレットリファレンスDescribeVolumeAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VolumeStatus

次の例は、Get-EC2VolumeStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したボリュームのステータスについて説明します。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

出力:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

出力:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

出力:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- API の詳細については、「 コマンドレットリファレンスDescribeVolumeStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2Vpc

次の例は、Get-EC2Vpc を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC について説明します。

```
Get-EC2Vpc -VpcId vpc-12345678
```

出力:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

例 2: この例では、デフォルトの VPC について説明します (リージョンごとに 1 つのみ指定できます) 。アカウントがこのリージョンで EC2-Classic をサポートしている場合、デフォルトの VPC はありません。

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

出力:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

例 3: この例では、指定されたフィルターに一致する VPCs (つまり、値 '10.0.0.0/16' に一致する CIDR があり、状態が 'available') について説明します。

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

例 4: この例では、すべての VPCsについて説明します。

```
Get-EC2Vpc
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcs」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpcAttribute

次の例は、Get-EC2VpcAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、enableDnsSupport「」属性について説明します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

出力:

```
EnableDnsSupport
----------------
True
```

例 2: この例では、enableDnsHostnames「」属性について説明します。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

出力:

```
EnableDnsHostnames
------------------
True
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcAttribute」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpcClassicLink

次の例は、Get-EC2VpcClassicLink を使用する方法を説明しています。

のツール PowerShell

例 1: 上記の例では、リージョン ClassicLinkEnabled の状態を持つすべての VPCs が返されます。

```
Get-EC2VpcClassicLink -Region eu-west-1
```

出力:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcClassicLink」の「」を参照してください。 AWS Tools for PowerShell

`Get-EC2VpcClassicLinkDnsSupport`

次の例は、`Get-EC2VpcClassicLinkDnsSupport` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョン eu-west-1 の VPCs ClassicLink DNS サポートステータスについて説明します。

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

出力:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcClassicLinkDnsSupport」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpcEndpoint

次の例は、Get-EC2VpcEndpoint を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リージョン eu-west-1 の 1 つ以上の VPC エンドポイントについて説明します。次に、出力を次のコマンドにパイプします。このコマンドは、 VpcEndpointId プロパティを選択し、配列 VPC ID を文字列配列として返します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

出力:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

例 2: この例では、リージョン eu-west-1 のすべての vpc エンドポイントについて説明し VpcEndpointId、、 VpcId、 ServiceName および PrivateDnsEnabled プロパティを選択して表形式で表示します。

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

出力:

```
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
```

例 3: この例では、VPC エンドポイント vpce-01a2ab3f4f5cc6f7d のポリシードキュメントを JSON ファイルにエクスポートします。

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcEndpoints」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpcEndpointService

次の例は、Get-EC2VpcEndpointService を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたフィルターを持つ EC2 VPC エンドポイントサービスについて説明します。この場合は com.amazonaws.eu-west-1.ecs です。さらに、 ServiceDetails プロパティも展開され、詳細が表示されます。

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

出力:

```
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
```

例 2: この例では、すべての EC2 VPC エンドポイントサービスを取得し、 ServiceNames 一致する「ssm」を返します。

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

出力:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpcEndpointServices」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpnConnection

次の例は、Get-EC2VpnConnection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPN 接続について説明します。

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
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
```

例 2: この例では、状態が保留中または使用可能な VPN 接続について説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

例 3: この例では、すべての VPN 接続について説明します。

```
Get-EC2VpnConnection
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpnConnections」の「」を参照してください。 AWS Tools for PowerShell

Get-EC2VpnGateway

次の例は、Get-EC2VpnGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイについて説明します。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

例 2: この例では、状態が保留中または使用可能な仮想プライベートゲートウェイについて説明します。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

例 3: この例では、すべての仮想プライベートゲートウェイについて説明します。

```
Get-EC2VpnGateway
```

- API の詳細については、「 コマンドレットリファレンスDescribeVpnGateways」の「」を参照してください。 AWS Tools for PowerShell

`Grant-EC2SecurityGroupEgress`

次の例は、`Grant-EC2SecurityGroupEgress` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-VPC の指定されたセキュリティグループの出力ルールを定義します。このルールは、TCP ポート 80 で指定された IP アドレス範囲へのアクセスを許可します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを許可します。

```
$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 } )
```

- API の詳細については、「 コマンドレットリファレンスAuthorizeSecurityGroupEgress」の「」を参照してください。 AWS Tools for PowerShell

Grant-EC2SecurityGroupIngress

次の例は、Grant-EC2SecurityGroupIngress を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-VPC のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。EC2-VPC のセキュリティグループは、セキュリティグループ名ではなくセキュリティグループ ID を使用して識別する必要があることに注意してください。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 3: この例では、EC2-Classic のセキュリティグループの進入ルールを定義します。これらのルールは、SSH (ポート 22) と RDC (ポート 3389) の特定の IP アドレスへのアクセスを許可します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$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 )
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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 )
```

例 5: この例では、指定されたソースセキュリティグループ (sg-1a2b3c4d) から指定されたセキュリティグループ (sg-12345678) への TCP ポート 8081 アクセスを許可します。

```
$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 } )
```

例 6: この例では、CIDR 5.5.5.5/32 を TCP ポート 22 トラフィックのセキュリティグループ sg-1234abcd の Ingress ルールに追加します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスAuthorizeSecurityGroupIngress」の「」を参照してください。 AWS Tools for PowerShell

Import-EC2Image

次の例は、Import-EC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、冪等性トークンを使用して、指定された Amazon S3 バケットから Amazon EC2 に単一ディスクの仮想マシンイメージをインポートします。この例では、VM Import Prequisites トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import Service ロールが存在し、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーが必要です。カスタムロールを使用するには、 **-RoleName**パラメータを使用してロール名を指定します。

```
$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
```

出力:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
```

```
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- API の詳細については、「 コマンドレットリファレンスImportImage」の「」を参照してください。 AWS Tools for PowerShell

Import-EC2KeyPair

次の例は、Import-EC2KeyPair を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パブリックキーを EC2 にインポートします。最初の行は、パブリックキーファイル (*.pub) の内容を変数 に保存します**$publickey**。次に、パブリックキーファイルの UTF8 形式を Base64-encodedされた文字列に変換し、変換された文字列を変数 に保存します**$pkbase64**。最後の行では、変換されたパブリックキーが EC2 にインポートされます。コマンドレットは、結果としてキーフィンガープリントと名前を返します。

```
$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
```

出力:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- API の詳細については、「 コマンドレットリファレンスImportKeyPair」の「」を参照してください。 AWS Tools for PowerShell

Import-EC2Snapshot

次の例は、Import-EC2Snapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「VMDK」形式の VM ディスクイメージを Amazon EBS スナップショットにインポートします。この例では、http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VM .html の**VM Import Prequisites**トピックで説明されているように、デフォルトの名前が「vmimport」の VM Import Service Role が必要ですImportPrerequisites。このロールには、指定されたバケットへの Amazon EC2 アクセスを許可するポリシーがあります。カスタムロールを使用するには、 **-RoleName**パラメータを使用してロール名を指定します。

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "myVirtualMachineImages"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

出力:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- API の詳細については、「 コマンドレットリファレンスImportSnapshot」の「」を参照してください。 AWS Tools for PowerShell

Move-EC2AddressToVpc

次の例は、Move-EC2AddressToVpc を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パブリック IP アドレスが 12.345.67.89 の EC2 インスタンスを、米国東部 (バージニア北部) リージョンの EC2-VPC プラットフォームに移動します。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

例 2: この例では、Get-EC2Instanceコマンドの結果を Move-EC2AddressToVpc コマンドレットにパイプします。Get-EC2Instance コマンドは、インスタンス ID で指定されたインスタンスを取得し、インスタンスのパブリック IP アドレスプロパティを返します。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- API の詳細については、「 コマンドレットリファレンスMoveAddressToVpc」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Address

次の例は、New-EC2Address を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VPC 内のインスタンスで使用する Elastic IP アドレスを割り当てます。

```
New-EC2Address -Domain Vpc
```

出力:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

例 2: この例では、EC2-Classic のインスタンスで使用する Elastic IP アドレスを割り当てます。

```
New-EC2Address
```

出力:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- API の詳細については、「 コマンドレットリファレンスAllocateAddress」の「」を参照してください。 AWS Tools for PowerShell

New-EC2CustomerGateway

次の例は、New-EC2CustomerGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたカスタマーゲートウェイを作成します。

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

出力:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- API の詳細については、「 コマンドレットリファレンスCreateCustomerGateway」の「」を参照してください。 AWS Tools for PowerShell

New-EC2DhcpOption

次の例は、New-EC2DhcpOption を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された DHCP オプションのセットを作成します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

出力:

```
DhcpConfigurations                    DhcpOptionsId    Tags
```

```
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

例 2: PowerShell バージョン 2 では、New-Object を使用して各 DHCP オプションを作成する必要があります。

```
$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)
```

出力:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- API の詳細については、「 コマンドレットリファレンスCreateDhcpOptions」の「」を参照してください。 AWS Tools for PowerShell

New-EC2FlowLog

次の例は、New-EC2FlowLog を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「Admin」ロールの権限を使用して、すべての「REJECT」トラフィックについて、サブネット subnet-1d234567 から「subnet1-log」という cloud-watch-log 名前の EC2 フローログを作成します。

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

出力:

```
ClientToken                                    FlowLogIds             Unsuccessful
-----------                                    ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- API の詳細については、「 コマンドレットリファレンスCreateFlowLogs」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Host

次の例は、New-EC2Host を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスタイプとアベイラビリティーゾーンのアカウントに Dedicated Host を割り当てます。

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

出力:

```
h-01e23f4cd567890f3
```

- API の詳細については、「 コマンドレットリファレンスAllocateHosts」の「」を参照してください。 AWS Tools for PowerShell

New-EC2HostReservation

次の例は、New-EC2HostReservation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Dedicated Host h-01e23f4cd567890f1 の設定に一致する設定で hro-0c1f23456789d0ab を提供する予約を購入します。

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

出力:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- API の詳細については、「 コマンドレットリファレンスPurchaseHostReservation」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Image

次の例は、New-EC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、イメージを作成する前にインスタンスのクリーンシャットダウンを試み、完了時にインスタンスを再起動します。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

例 2: この例では、指定されたインスタンスから、指定された名前と説明を持つ AMI を作成します。Amazon EC2 は、インスタンスをシャットダウンして再起動せずにイメージを作成します。したがって、作成されたイメージのファイルシステムの整合性は保証できません。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

例 3: この例では、3 つのボリュームを持つ AMI を作成します。最初のボリュームは Amazon EBS スナップショットに基づいています。2 番目のボリュームは、空の 100 GiB Amazon EBS ボリュームです。3 番目のボリュームはインスタンスストアボリュームです。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$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"})
```

- API の詳細については、「 コマンドレットリファレンスCreateImage」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Instance

次の例は、New-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-Classic またはデフォルト VPC で指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

例 2: この例では、VPC 内の指定された AMI の単一のインスタンスを起動します。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

例 3: EBS ボリュームまたはインスタンスストアボリュームを追加するには、ブロックデバイスマッピングを定義して コマンドに追加します。この例では、インスタンスストアボリュームを追加します。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

例 4: 現在の Windows AMIs の 1 つを指定するには、 を使用してその AMI ID を取得しますGet-EC2ImageByName。この例では、Windows Server 2016 の現在のベース AMI からインスタンスを起動します。

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE
```

```
New-EC2Instance -ImageId $ami.ImageId ...
```

例 5: 指定された専用ホスト環境でインスタンスを起動します。

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

例 6: このリクエストは 2 つのインスタンスを起動し、ウェブサーバーのキーと本番稼働用の値を含むタグをインスタンスに適用します。リクエストは、作成されたボリューム (この場合は各インスタンスのルートボリューム) に、コストセンターのキーと cc123 の値を持つタグも適用します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスRunInstances」の「」を参照してください。 AWS Tools for PowerShell

New-EC2InstanceExportTask

次の例は、New-EC2InstanceExportTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、停止したインスタンス を仮想ハードディスク (VHD) **i-0800b00a00EXAMPLE**として S3 バケット にエクスポートします**testbucket-export-instances-2019**。ターゲット環境は で**Microsoft**、インスタンスが **us-east-1**リージョ

ンにあり、ユーザーのデフォルト AWS リージョンが us-east-1 ではないため、リージョンパラメータが追加されます。エクスポートタスクのステータスを取得するには、このコマンドの結果から**ExportTaskId**値をコピーし、 を実行します。 **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
```

出力:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- API の詳細については、「 コマンドレットリファレンスCreateInstanceExportTask」の「」を参照してください。 AWS Tools for PowerShell

New-EC2InternetGateway

次の例は、New-EC2InternetGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インターネットゲートウェイを作成します。

```
New-EC2InternetGateway
```

出力:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- API の詳細については、「 コマンドレットリファレンスCreateInternetGateway」の「」を参照してください。 AWS Tools for PowerShell

New-EC2KeyPair

次の例は、New-EC2KeyPair を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、キーペアを作成し、指定した名前のファイルで PEM エンコードされた RSA プライベートキーをキャプチャします。を使用する場合 PowerShell、有効なキーを生成するには、エンコーディングを ascii に設定する必要があります。詳細については、 AWS コマンドラインインターフェイスユーザーガイドのAmazon EC2 キーペアの作成、表示、削除 (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html)」を参照してください。

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- API の詳細については、「 コマンドレットリファレンスCreateKeyPair」の「」を参照してください。 AWS Tools for PowerShell

New-EC2NetworkAcl

次の例は、New-EC2NetworkAcl を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC のネットワーク ACL を作成します。

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

出力:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスCreateNetworkAcl」の「」を参照してください。 AWS Tools for PowerShell

New-EC2NetworkAclEntry

次の例は、New-EC2NetworkAclEntry を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL のエントリを作成します。このルールでは、UDP ポート 53 (DNS) 上の任意の場所 (0.0.0.0/0) から、関連付けられたサブネットへのインバウンドトラフィックを許可します。

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateNetworkAclEntry」の「」を参照してください。 AWS Tools for PowerShell

New-EC2NetworkInterface

次の例は、New-EC2NetworkInterface を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスを作成します。

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateNetworkInterface」の「」を参照してください。 AWS Tools for PowerShell

New-EC2PlacementGroup

次の例は、New-EC2PlacementGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した名前のプレイスメントグループを作成します。

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- API の詳細については、「 コマンドレットリファレンスCreatePlacementGroup」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Route

次の例は、New-EC2Route を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルの指定されたルートを作成します。ルートはすべてのトラフィックに一致し、指定されたインターネットゲートウェイに送信します。

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId
 igw-1a2b3c4d
```

出力:

```
True
```

- API の詳細については、「 コマンドレットリファレンスCreateRoute」の「」を参照してください。 AWS Tools for PowerShell

New-EC2RouteTable

次の例は、New-EC2RouteTable を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC のルートテーブルを作成します。

```
New-EC2RouteTable -VpcId vpc-12345678
```

出力:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスCreateRouteTable」の「」を参照してください。 AWS Tools for PowerShell

New-EC2ScheduledInstance

次の例は、New-EC2ScheduledInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスケジュールされたインスタンスを起動します。

```
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
```

- API の詳細については、「 コマンドレットリファレンスRunScheduledInstances」の「」を参照してください。 AWS Tools for PowerShell

New-EC2ScheduledInstancePurchase

次の例は、New-EC2ScheduledInstancePurchase を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、スケジュールされたインスタンスを購入します。

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスPurchaseScheduledInstances」の「」を参照してください。 AWS Tools for PowerShell

New-EC2SecurityGroup

次の例は、New-EC2SecurityGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC のセキュリティグループを作成します。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

出力:

```
sg-12345678
```

例 2: この例では、EC2-Classic のセキュリティグループを作成します。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

出力:

```
sg-45678901
```

- API の詳細については、「 コマンドレットリファレンスCreateSecurityGroup」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Snapshot

次の例は、New-EC2Snapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したボリュームのスナップショットを作成します。

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateSnapshot」の「」を参照してください。 AWS Tools for PowerShell

New-EC2SpotDatafeedSubscription

次の例は、New-EC2SpotDatafeedSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、スポットインスタンスのデータフィードを作成します。

```
New-EC2SpotDatafeedSubscription -Bucket my-s3-bucket -Prefix spotdata
```

出力:

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- API の詳細については、「 コマンドレットリファレンスCreateSpotDatafeedSubscription」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Subnet

次の例は、New-EC2Subnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された CIDR を使用してサブネットを作成します。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateSubnet」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Tag

次の例は、New-EC2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースに単一のタグを追加します。タグキーは「myTag」で、タグ値はmyTagValue「」です。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

例 2: この例では、指定したタグを更新または指定したリソースに追加します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

例 3: PowerShell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"
```

```
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API の詳細については、「 コマンドレットリファレンスCreateTags」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Volume

次の例は、New-EC2Volume を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームを作成します。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

出力:

```
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
```

例 2: このリクエスト例では、ボリュームを作成し、スタックのキーと本番稼働用の値を含むタグを適用します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateVolume」の「」を参照してください。 AWS Tools for PowerShell

New-EC2Vpc

次の例は、New-EC2Vpc を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された CIDR を使用して VPC を作成します。Amazon VPC では、VPC 用に、デフォルトの DHCP オプションセット、メインルートテーブル、およびデフォルトのネットワーク ACL も作成されます。

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

出力:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスCreateVpc」の「」を参照してください。 AWS Tools for PowerShell

New-EC2VpcEndpoint

次の例は、New-EC2VpcEndpoint を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VPC vpc-0fc1ff23f45b678eb でサービス com.amazonaws.eu-west-1.s3 の新しい VPC エンドポイントを作成します。

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

出力:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- API の詳細については、「 コマンドレットリファレンスCreateVpcEndpoint」の「」を参照してください。 AWS Tools for PowerShell

New-EC2VpnConnection

次の例は、New-EC2VpnConnection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイと指定されたカスタマーゲートウェイの間に VPN 接続を作成します。出力には、ネットワーク管理者が必要とする設定情報が XML 形式で含まれます。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

出力:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

例 2: この例では、VPN 接続を作成し、指定した名前のファイルに設定をキャプチャします。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

例 3: この例では、指定された仮想プライベートゲートウェイと指定されたカスタマーゲートウェイの間に、静的ルーティングを使用して VPN 接続を作成します。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API の詳細については、「 コマンドレットリファレンスCreateVpnConnection」の「」を参照してください。 AWS Tools for PowerShell

New-EC2VpnConnectionRoute

次の例は、New-EC2VpnConnectionRoute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPN 接続用に指定された静的ルートを作成します。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- API の詳細については、「 コマンドレットリファレンスCreateVpnConnectionRoute」の「」を参照してください。 AWS Tools for PowerShell

New-EC2VpnGateway

次の例は、New-EC2VpnGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイを作成します。

```
New-EC2VpnGateway -Type ipsec.1
```

出力:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
```

```
VpnGatewayId     : vgw-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスCreateVpnGateway」の「」を参照してください。 AWS Tools for PowerShell

Register-EC2Address

次の例は、Register-EC2Address を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Elastic IP アドレスを VPC 内の指定されたインスタンスに関連付けます。

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

出力:

```
eipassoc-12345678
```

例 2: この例では、指定された Elastic IP アドレスを EC2-Classic の指定されたインスタンスに関連付けます。

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- API の詳細については、「 コマンドレットリファレンスAssociateAddress」の「」を参照してください。 AWS Tools for PowerShell

Register-EC2DhcpOption

次の例は、Register-EC2DhcpOption を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された DHCP オプションセットを指定された VPC に関連付けます。

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

例 2: この例では、デフォルトの DHCP オプションセットを指定された VPC に関連付けます。

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- API の詳細については、「 コマンドレットリファレンスAssociateDhcpOptions」の「」を参照してください。 AWS Tools for PowerShell

Register-EC2Image

次の例は、Register-EC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Amazon S3 で指定されたマニフェストファイルを使用して AMI を登録します。

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/image.manifest.xml -
Name my-web-server-ami
```

- API の詳細については、「 コマンドレットリファレンスRegisterImage」の「」を参照してください。 AWS Tools for PowerShell

Register-EC2PrivateIpAddress

次の例は、Register-EC2PrivateIpAddress を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたセカンダリプライベート IP アドレスを指定されたネットワークインターフェイスに割り当てます。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

例 2: この例では、2 つのセカンダリプライベート IP アドレスを作成し、指定されたネットワークインターフェイスに割り当てます。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- API の詳細については、「 コマンドレットリファレンスAssignPrivateIpAddresses」の「」を参照してください。 AWS Tools for PowerShell

Register-EC2RouteTable

次の例は、Register-EC2RouteTable を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルを指定されたサブネットに関連付けます。

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

出力:

```
rtbassoc-12345678
```

- API の詳細については、「 コマンドレットリファレンスAssociateRouteTable」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Address

次の例は、Remove-EC2Address を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VPC 内のインスタンス用に指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

例 2: この例では、EC2-Classic のインスタンス用に指定された Elastic IP アドレスを解放します。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API の詳細については、「 コマンドレットリファレンスReleaseAddress」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2CapacityReservation

次の例は、Remove-EC2CapacityReservation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、キャパシティ予約 cr-0c1f2345db6f7cdba をキャンセルします。

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCancelCapacityReservation」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2CustomerGateway

次の例は、Remove-EC2CustomerGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたカスタマーゲートウェイを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteCustomerGateway」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2DhcpOption

次の例は、Remove-EC2DhcpOption を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された DHCP オプションセットを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteDhcpOptions」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2FlowLog

次の例は、Remove-EC2FlowLog を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された FlowLogId fl-01a2b3456a789c01 を削除します。

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteFlowLogs」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Host

次の例は、Remove-EC2Host を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたホスト ID h-0badafd1dcb2f3456 をリリースします。

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

出力:

```
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} {}
```

- API の詳細については、「 コマンドレットリファレンスReleaseHosts」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Instance

次の例は、Remove-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを終了します (インスタンスは実行中または停止状態である可能性があります)。コマンドレットは続行する前に確認を求めます。-Force スイッチを使用してプロンプトを非表示にします。

```
Remove-EC2Instance -InstanceId i-12345678
```

出力:

```
CurrentState                        InstanceId    PreviousState
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API の詳細については、「 コマンドレットリファレンスTerminateInstances」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2InternetGateway

次の例は、Remove-EC2InternetGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインターネットゲートウェイを削除します。Force パラメータも指定しない限り、操作が続行する前に確認を求められます。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteInternetGateway」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2KeyPair

次の例は、Remove-EC2KeyPair を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキーペアを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteKeyPair」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2NetworkAcl

次の例は、Remove-EC2NetworkAcl を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL を削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteNetworkAcl」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2NetworkAclEntry

次の例は、Remove-EC2NetworkAclEntry を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL から指定されたルールを削除します。Force パラメータも指定しない限り、オペレーションが続行する前に確認を求められます。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteNetworkAclEntry」の「」を参照してください。 AWS Tools for PowerShell

`Remove-EC2NetworkInterface`

次の例は、Remove-EC2NetworkInterface を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteNetworkInterface」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2PlacementGroup

次の例は、Remove-EC2PlacementGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたプレイスメントグループを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeletePlacementGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Route

次の例は、Remove-EC2Route を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルから指定されたルートを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteRoute」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2RouteTable

次の例は、Remove-EC2RouteTable を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteRouteTable」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2SecurityGroup

次の例は、Remove-EC2SecurityGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-VPC の指定されたセキュリティグループを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

出力:

```
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"):
```

例 2: この例では、EC2-Classic の指定されたセキュリティグループを削除します。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteSecurityGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Snapshot

次の例は、Remove-EC2Snapshot を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットを削除します。Force パラメータも指定しない限り、オペレーションが続行する前に確認を求められます。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteSnapshot」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2SpotDatafeedSubscription

次の例は、Remove-EC2SpotDatafeedSubscription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、スポットインスタンスのデータフィードを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2SpotDatafeedSubscription
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteSpotDatafeedSubscription」の「」を参照してください。 AWS Tools for PowerShell

`Remove-EC2Subnet`

次の例は、Remove-EC2Subnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたサブネットを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteSubnet」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Tag

次の例は、Remove-EC2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、タグ値に関係なく、指定されたタグを指定されたリソースから削除します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

例 2: この例では、指定されたタグ値が一致する場合にのみ、指定されたタグを指定されたリソースから削除します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

例 3: この例では、タグ値に関係なく、指定されたタグを指定されたリソースから削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

例 4: この例では、指定されたタグ値が一致する場合にのみ、指定されたタグを指定されたリソースから削除します。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteTags」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Volume

次の例は、Remove-EC2Volume を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたボリュームをデタッチします。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2Volume -VolumeId vol-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteVolume」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2Vpc

次の例は、Remove-EC2Vpc を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC を削除します。Force パラメータも指定しない限り、操作が続行する前に確認を求められます。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteVpc」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2VpnConnection

次の例は、Remove-EC2VpnConnection を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPN 接続を削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteVpnConnection」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2VpnConnectionRoute

次の例は、Remove-EC2VpnConnectionRoute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPN 接続から指定された静的ルートを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteVpnConnectionRoute」の「」を参照してください。 AWS Tools for PowerShell

Remove-EC2VpnGateway

次の例は、Remove-EC2VpnGateway を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された仮想プライベートゲートウェイを削除します。Force パラメータも指定しない限り、オペレーションが続行する前に確認を求められます。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteVpnGateway」の「」を参照してください。 AWS Tools for PowerShell

Request-EC2SpotFleet

次の例は、Request-EC2SpotFleet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスタイプの最低価格のスポットフリートリクエストをアベイラビリティーゾーンに作成します。アカウントが EC2-VPC のみをサポートしている場合、スポットフリートは、デフォルトサブネットを持つ最低価格のアベイラビリティーゾーンでインスタンスを起動します。アカウントが EC2-Classic をサポートしている場合、スポットフリートは

最低価格のアベイラビリティーゾーンで EC2-Classic でインスタンスを起動します。支払う料金は、リクエストに対して指定されたスポット料金を超えないことに注意してください。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスRequestSpotFleet」の「」を参照してください。 AWS Tools for PowerShell

Request-EC2SpotInstance

次の例は、Request-EC2SpotInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたサブネットに 1 回限りのスポットインスタンスをリクエストします。セキュリティグループは、指定されたサブネットを含む VPC 用に作成する必要があり、ネットワークインターフェイスを使用して ID で指定する必要があることに注意してください。ネットワークインターフェイスを指定するときは、ネットワークインターフェイスを使用してサブネット ID を含める必要があります。

```
$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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスRequestSpotInstances」の「」を参照してください。 AWS Tools for PowerShell

Reset-EC2ImageAttribute

次の例は、Reset-EC2ImageAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、launchPermission」属性をデフォルト値にリセットします。デフォルトでは、AMIsプライベートです。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API の詳細については、「 コマンドレットリファレンスResetImageAttribute」の「」を参照してください。 AWS Tools for PowerShell

Reset-EC2InstanceAttribute

次の例は、Reset-EC2InstanceAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスのsriovNetSupport「」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

例 2: この例では、指定したインスタンスのebsOptimized」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

例 3: この例では、指定したインスタンスのsourceDestCheck「」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

例 4: この例では、指定したインスタンスのdisableApiTermination「」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

例 5: この例では、指定したインスタンスのinstanceInitiatedShutdown「Behavior」属性をリセットします。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- API の詳細については、「 コマンドレットリファレンスResetInstanceAttribute」の「」を参照してください。 AWS Tools for PowerShell

`Reset-EC2NetworkInterfaceAttribute`

次の例は、`Reset-EC2NetworkInterfaceAttribute` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワークインターフェイスの送信元/送信先チェックをリセットします。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- API の詳細については、「 コマンドレットリファレンスResetNetworkInterfaceAttribute」の「」を参照してください。 AWS Tools for PowerShell

Reset-EC2SnapshotAttribute

次の例は、Reset-EC2SnapshotAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスナップショットの指定された属性をリセットします。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- API の詳細については、「 コマンドレットリファレンスResetSnapshotAttribute」の「」を参照してください。 AWS Tools for PowerShell

Restart-EC2Instance

次の例は、Restart-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを再起動します。

```
Restart-EC2Instance -InstanceId i-12345678
```

- API の詳細については、「 コマンドレットリファレンスRebootInstances」の「」を参照してください。 AWS Tools for PowerShell

Revoke-EC2SecurityGroupEgress

次の例は、Revoke-EC2SecurityGroupEgress を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-VPC の指定されたセキュリティグループのルールを削除します。これにより、TCP ポート 80 で指定された IP アドレス範囲へのアクセスが取り消されます。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、TCP ポート 80 で指定されたソースセキュリティグループへのアクセスを取り消します。

```
$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 } )
```

- API の詳細については、「 コマンドレットリファレンスRevokeSecurityGroupEgress」の「」を参照してください。 AWS Tools for PowerShell

`Revoke-EC2SecurityGroupIngress`

次の例は、`Revoke-EC2SecurityGroupIngress` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、EC2-VPC の指定されたセキュリティグループの指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。セキュリティグループ名ではなくセキュリティグループ ID を使用して EC2-VPC のセキュリティグループを識別する必要があることに注意してください。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

例 2: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

例 3: この例では、EC2-Classic の指定されたセキュリティグループの指定されたアドレス範囲から TCP ポート 22 へのアクセスを取り消します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

例 4: PowerShell バージョン 2 では、New-Object を使用して IpPermission オブジェクトを作成する必要があります。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスRevokeSecurityGroupIngress」の「」を参照してください。 AWS Tools for PowerShell

Send-EC2InstanceStatus

次の例は、Send-EC2InstanceStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスのステータスフィードバックを報告します。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- API の詳細については、「 コマンドレットリファレンスReportInstanceStatus」の「」を参照してください。 AWS Tools for PowerShell

Set-EC2NetworkAclAssociation

次の例は、Set-EC2NetworkAclAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL を、指定されたネットワーク ACL 関連付けのサブネットに関連付けます。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

出力:

```
aclassoc-87654321
```

- API の詳細については、「 コマンドレットリファレンスReplaceNetworkAclAssociation」の「」を参照してください。 AWS Tools for PowerShell

Set-EC2NetworkAclEntry

次の例は、Set-EC2NetworkAclEntry を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたネットワーク ACL の指定されたエントリを置き換えます。新しいルールでは、指定されたアドレスから関連付けられたサブネットへのインバウンドトラフィックが許可されます。

```
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
```

- API の詳細については、「 コマンドレットリファレンスReplaceNetworkAclEntry」の「」を参照してください。 AWS Tools for PowerShell

Set-EC2Route

次の例は、Set-EC2Route を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルの指定されたルートを置き換えます。新しいルートは、指定されたトラフィックを指定された仮想プライベートゲートウェイに送信します。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスReplaceRoute」の「」を参照してください。 AWS Tools for PowerShell

Set-EC2RouteTableAssociation

次の例は、Set-EC2RouteTableAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたルートテーブルを、指定されたルートテーブルの関連付けのサブネットに関連付けます。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

出力:

```
rtbassoc-87654321
```

- API の詳細については、「 コマンドレットリファレンスReplaceRouteTableAssociation」の「」を参照してください。 AWS Tools for PowerShell

Start-EC2Instance

次の例は、Start-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを起動します。

```
Start-EC2Instance -InstanceId i-12345678
```

出力:

```
CurrentState                        InstanceId    PreviousState
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

例 2: この例では、指定されたインスタンスを起動します。

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

例 3: この例では、現在停止しているインスタンスのセットを起動します。によって返されるインスタンスオブジェクトGet-EC2Instanceは、 にパイプされますStart-EC2Instance。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

例 4: PowerShell バージョン 2 では、New-Object を使用して Filter パラメータのフィルターを作成する必要があります。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- API の詳細については、「 コマンドレットリファレンスStartInstances」の「」を参照してください。 AWS Tools for PowerShell

`Start-EC2InstanceMonitoring`

次の例は、`Start-EC2InstanceMonitoring` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスの詳細モニタリングを有効にします。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API の詳細については、「 コマンドレットリファレンスMonitorInstances」の「」を参照してください。 AWS Tools for PowerShell

Stop-EC2ImportTask

次の例は、Stop-EC2ImportTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインポートタスク (スナップショットまたはイメージのインポート) をキャンセルします。必要に応じて、 **-CancelReason**パラメータを使用して理由を指定できます。

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- API の詳細については、「 コマンドレットリファレンスCancelImportTask」の「」を参照してください。 AWS Tools for PowerShell

Stop-EC2Instance

次の例は、Stop-EC2Instance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたインスタンスを停止します。

```
Stop-EC2Instance -InstanceId i-12345678
```

出力:

```
CurrentState                    InstanceId    PreviousState
```

```
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API の詳細については、「 コマンドレットリファレンスStopInstances」の「」を参照してください。 AWS Tools for PowerShell

Stop-EC2InstanceMonitoring

次の例は、Stop-EC2InstanceMonitoring を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したインスタンスの詳細モニタリングを無効にします。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

出力:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API の詳細については、「 コマンドレットリファレンスUnmonitorInstances」の「」を参照してください。 AWS Tools for PowerShell

Stop-EC2SpotFleetRequest

次の例は、Stop-EC2SpotFleetRequest を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットフリートリクエストをキャンセルし、関連付けられたスポットインスタンスを終了します。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

例 2: この例では、関連付けられたスポットインスタンスを終了せずに、指定されたスポットフリートリクエストをキャンセルします。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- API の詳細については、「 コマンドレットリファレンスCancelSpotFleetRequests」の「」を参照してください。 AWS Tools for PowerShell

Stop-EC2SpotInstanceRequest

次の例は、Stop-EC2SpotInstanceRequest を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたスポットインスタンスリクエストをキャンセルします。

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

出力:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- API の詳細については、「 コマンドレットリファレンスCancelSpotInstanceRequests」の「」を参照してください。 AWS Tools for PowerShell

Unregister-EC2Address

次の例は、Unregister-EC2Address を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された Elastic IP アドレスと VPC 内の指定されたインスタンスの関連付けを解除します。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

例 2: この例では、EC2-Classic の指定されたインスタンスから指定された Elastic IP アドレスの関連付けを解除します。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API の詳細については、「 コマンドレットリファレンスDisassociateAddress」の「」を参照してください。 AWS Tools for PowerShell

`Unregister-EC2Image`

次の例は、`Unregister-EC2Image` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された AMI の登録を解除します。

```
Unregister-EC2Image -ImageId ami-12345678
```

- API の詳細については、「 コマンドレットリファレンスDeregisterImage」の「」を参照してください。 AWS Tools for PowerShell

`Unregister-EC2PrivateIpAddress`

次の例は、`Unregister-EC2PrivateIpAddress` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたプライベート IP アドレスを指定されたネットワークインターフェイスから割り当て解除します。

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- API の詳細については、「 コマンドレットリファレンスUnassignPrivateIpAddresses」の「」を参照してください。 AWS Tools for PowerShell

`Unregister-EC2RouteTable`

次の例は、`Unregister-EC2RouteTable` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ルートテーブルとサブネット間の指定された関連付けを削除します。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDisassociateRouteTable」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon ECR の例 PowerShell

次のコード例は、Amazon ECR AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

Get-ECRLoginCommand

次の例は、Get-ECRLoginCommand を使用する方法を説明しています。

のツール PowerShell

例 1: IAM プリンシパルがアクセスできる任意の Amazon ECR レジストリへの認証に使用できるログイン情報を含む PSObject を返します。認証トークンを取得するために呼び出しに必要な認証情報とリージョンエンドポイントは、シェルのデフォルト (**Set-AWSCredential/ Set-DefaultAWSRegion**または コマンドレットによってセットアップ) **Initialize-AWSDefaultConfiguration** から取得されます。コマンドプロパティを Invoke-Expression で使用して、指定されたレジストリにログインしたり、ログインを必要とする他のツールで返された認証情報を使用したりできます。

```
Get-ECRLoginCommand
```

出力:

```
Username      : AWS
Password      : eyJwYXlsb2Fk...kRBVEFfS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
 eyJwYXlsb2Fk...kRBVEFfS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

例 2: Docker ログインコマンドへの入力として使用するログイン情報を含む PSObject を取得します。IAM プリンシパルがそのレジストリにアクセスできる限り、認証する任意の Amazon ECR レジストリ URI を指定できます。

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- API の詳細については、「 コマンドレットリファレンス」の「Get-ECRLoginCommand」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon ECS の例 PowerShell

次のコード例は、Amazon ECS AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Get-ECSClusterDetail`

次の例は、`Get-ECSClusterDetail` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットでは、1 つ以上の ECS クラスターについて説明します。

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

出力:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- API の詳細については、「 コマンドレットリファレンスDescribeClusters」の「」を参照してください。 AWS Tools for PowerShell

`Get-ECSClusterList`

次の例は、`Get-ECSClusterList` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、既存の ECS クラスターのリストを返します。

```
Get-ECSClusterList
```

出力:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- API の詳細については、「 コマンドレットリファレンスListClusters」の「」を参照してください。 AWS Tools for PowerShell

Get-ECSClusterService

次の例は、Get-ECSClusterService を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、デフォルトクラスターで実行されているすべてのサービスを一覧表示します。

```
Get-ECSClusterService
```

例 2: この例では、指定したクラスターで実行されているすべてのサービスを一覧表示します。

```
Get-ECSClusterService -Cluster myCluster
```

例 3: この例では、指定したクラスターで実行されているサービスを一覧表示し、一度に最大 10 個のサービスの詳細を取得します。

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスListServices」の「」を参照してください。 AWS Tools for PowerShell

Get-ECSService

次の例は、Get-ECSService を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、デフォルトクラスターから特定のサービスの詳細を取得する方法を示します。

```
Get-ECSService -Service my-hhtp-service
```

例 2: この例は、名前付きクラスターで実行されている特定のサービスの詳細を取得する方法を示しています。

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- API の詳細については、「 コマンドレットリファレンスDescribeServices」の「」を参照してください。 AWS Tools for PowerShell

New-ECSCluster

次の例は、New-ECSCluster を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは新しい Amazon ECS クラスターを作成します。

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

出力:

```
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                             : {}
```

- API の詳細については、「 コマンドレットリファレンスCreateCluster」の「」を参照してください。 AWS Tools for PowerShell

New-ECSService

次の例は、New-ECSService を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンド例では、デフォルトのクラスターにecs-simple-service「」というサービスを作成します。このサービスは「ecs-demo」タスク定義を使用し、そのタスクのインスタンス化を 10 回維持します。

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

例 2: このコマンド例では、「」という名前のデフォルトクラスターのロードバランサーの背後にサービスを作成しますecs-simple-service。このサービスは「ecs-demo」タスク定義を使用し、そのタスクのインスタンス化を 10 回維持します。

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- API の詳細については、「 コマンドレットリファレンスCreateService」の「」を参照してください。 AWS Tools for PowerShell

Remove-ECSCluster

次の例は、Remove-ECSCluster を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定された ECS クラスターを削除します。削除する前に、このクラスターからすべてのコンテナインスタンスを登録解除する必要があります。

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteCluster」の「」を参照してください。 AWS Tools for PowerShell

Remove-ECSService

次の例は、Remove-ECSService を使用する方法を説明しています。

のツール PowerShell

例 1: デフォルトクラスターで「」という名前my-http-serviceのサービスを削除します。サービスを削除する前に、サービスに必要な数と実行数が 0 である必要があります。コマンドを続行する前に、確認を求められます。確認プロンプトをバイパスするには、-Force スイッチを追加します。

```
Remove-ECSService -Service my-http-service
```

例 2: 名前付きクラスター内のmy-http-service「」という名前のサービスを削除します。

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- API の詳細については、「 コマンドレットリファレンスDeleteService」の「」を参照してください。 AWS Tools for PowerShell

Update-ECSClusterSetting

次の例は、Update-ECSClusterSetting を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、ECS クラスターに使用する設定を変更します。

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

出力:

```
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                               : {}
```

- API の詳細については、「 コマンドレットリファレンスUpdateClusterSettings」の「」を参照してください。 AWS Tools for PowerShell

Update-ECSService

次の例は、Update-ECSService を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンド例では、my-http-service`` タスク定義を使用するように amazon-ecs-sample`` サービスを更新します。

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

例 2: このコマンド例では、my-http-service`` サービスの必要数を 10 に更新します。

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- API の詳細については、「 コマンドレットリファレンスUpdateService」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon EFS の例 PowerShell

次のコード例は、Amazon EFS AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

アクション

`Edit-EFSMountTargetSecurityGroup`

次の例は、`Edit-EFSMountTargetSecurityGroup` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたマウントターゲットに対して有効なセキュリティグループを更新します。「sg-xxxxxxxx」の形式で最大 5 つ指定できます。

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- API の詳細については、「 コマンドレットリファレンスModifyMountTargetSecurityGroups」の「」を参照してください。 AWS Tools for PowerShell

`Get-EFSFileSystem`

次の例は、`Get-EFSFileSystem` を使用する方法を説明しています。

のツール PowerShell

例 1: リージョン内の発信者のアカウントが所有するすべてのファイルシステムのコレクションを返します。

```
Get-EFSFileSystem
```

出力:

```
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
...
```

例 2: 指定されたファイルシステムの詳細を返します。

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

例 3: ファイルシステムの作成時に指定されたべき等作成トークンを使用して、ファイルシステムの詳細を返します。

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- API の詳細については、「 コマンドレットリファレンスDescribeFileSystems」の「」を参照してください。 AWS Tools for PowerShell

`Get-EFSMountTarget`

次の例は、`Get-EFSMountTarget` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたファイルシステムに関連付けられたマウントターゲットのコレクションを返します。

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

出力:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDescribeMountTargets」の「」を参照してください。 AWS Tools for PowerShell

Get-EFSMountTargetSecurityGroup

次の例は、Get-EFSMountTargetSecurityGroup を使用する方法を説明しています。

のツール PowerShell

例 1: マウントターゲットに関連付けられたネットワークインターフェイスに現在割り当てられているセキュリティグループの ID を返します。

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

出力:

```
sg-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDescribeMountTargetSecurityGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-EFSTag

次の例は、Get-EFSTag を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたファイルシステムに現在関連付けられているタグのコレクションを返します。

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

出力:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

New-EFSFileSystem

次の例は、New-EFSFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: 新しい空のファイルシステムを作成します。冪等性の作成を確実に行うために使用されるトークンは自動的に生成され、返されたオブジェクト**CreationToken**のメンバーからアクセスできます。

```
New-EFSFileSystem
```

出力:

```
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
```

例 2: カスタムトークンを使用して新しい空のファイルシステムを作成し、冪等性の作成を確実にします。

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- API の詳細については、「 コマンドレットリファレンスCreateFileSystem」の「」を参照してください。 AWS Tools for PowerShell

New-EFSMountTarget

次の例は、New-EFSMountTarget を使用する方法を説明しています。

のツール PowerShell

例 1: ファイルシステムの新しいマウントターゲットを作成します。指定されたサブネットを使用して、マウントターゲットが作成される仮想プライベートクラウド (VPC) と、 (サブネットのアドレス範囲から) 自動割り当てされる IP アドレスを決定します。割り当てられた IP アドレスを使用して、このファイルシステムを Amazon EC2 インスタンスにマウントできます。セキュリティグループが指定されていないため、ターゲット用に作成されたネットワークインターフェイスは、サブネットの VPC のデフォルトのセキュリティグループに関連付けられます。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

出力:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

例 2: 自動割り当て IP アドレスを使用して、指定されたファイルシステムの新しいマウントターゲットを作成します。マウントターゲット用に作成されたネットワークインターフェイスは、指定されたセキュリティグループに関連付けられます (「sg-xxxxxxxx」の形式で最大 5 つまで指定できます) 。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

例 3: 指定された IP アドレスを使用して、指定されたファイルシステムの新しいマウントターゲットを作成します。

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- API の詳細については、「 コマンドレットリファレンスCreateMountTarget」の「」を参照してください。 AWS Tools for PowerShell

New-EFSTag

次の例は、New-EFSTag を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたファイルシステムにタグのコレクションを適用します。指定されたキーを持つタグがファイルシステムにすでに存在する場合、タグの値は更新されます。

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

例 2: 指定されたファイルシステムの名前タグを設定します。この値は、Get-EFSFileSystem コマンドレットが使用されるときに、他のファイルシステムの詳細とともに返されます。

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- API の詳細については、「 コマンドレットリファレンスCreateTags」の「」を参照してください。 AWS Tools for PowerShell

Remove-EFSFileSystem

次の例は、Remove-EFSFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: 使用されなくなった指定されたファイルシステムを削除します (ファイルシステムにマウントターゲットがある場合は、最初に削除する必要があります)。コマンドレットが進む前に確認を求められます。確認を抑制するには、 **-Force**スイッチを使用します。

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDeleteFileSystem」の「」を参照してください。 AWS Tools for PowerShell

Remove-EFSMountTarget

次の例は、Remove-EFSMountTarget を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたマウントターゲットを削除します。操作を続行する前に確認画面が表示されます。プロンプトを抑制するには、 **-Force**スイッチを使用します。このオペレーションでは、ターゲット経由でファイルシステムのマウントが強制的に中断されることに注意してください。可能であれば、このコマンドを実行する前にファイルシステムのアンマウントを検討してください。

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- API の詳細については、「 コマンドレットリファレンスDeleteMountTarget」の「」を参照してください。 AWS Tools for PowerShell

Remove-EFSTag

次の例は、Remove-EFSTag を使用する方法を説明しています。

のツール PowerShell

例 1: ファイルシステムから 1 つ以上のタグのコレクションを削除します。コマンドレットが進む前に確認を求められます。確認を抑制するには、 **-Force**スイッチを使用します。

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- API の詳細については、「 コマンドレットリファレンスDeleteTags」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon EKS の例 PowerShell

次のコード例は、Amazon EKS AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Add-EKSResourceTag

次の例は、Add-EKSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定されたタグを指定された resourceArn のリソースに関連付けます。

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- API の詳細については、「 コマンドレットリファレンスTagResource」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSCluster

次の例は、Get-EKSCluster を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS クラスターに関する説明情報を返します。

```
Get-EKSCluster -Name "PROD"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeCluster」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSClusterList

次の例は、Get-EKSClusterList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定したリージョンの にある Amazon EKS クラスター AWS アカウント を一覧表示します。

```
Get-EKSClusterList
```

出力:

```
PROD
```

- API の詳細については、「 コマンドレットリファレンスListClusters」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSFargateProfile

次の例は、Get-EKSFargateProfile を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、 AWS Fargate プロファイルに関する説明情報を返します。

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

出力:

```
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                : {}
```

- API の詳細については、「 コマンドレットリファレンスDescribeFargateProfile」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSFargateProfileList

次の例は、Get-EKSFargateProfileList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定されたリージョンの で指定されたクラスターに関連付けられた AWS Fargate プロファイル AWS アカウント を一覧表示します。

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

出力:

```
EKSFargate
```

```
EKSFargateProfile
```

- API の詳細については、「 コマンドレットリファレンスListFargateProfiles」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSNodegroup

次の例は、Get-EKSNodegroup を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS ノードグループに関する説明情報を返します。

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeNodegroup」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSNodegroupList

次の例は、Get-EKSNodegroupList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定したリージョンの の指定したクラスターに関連付けられた Amazon EKS ノードグループ AWS アカウント を一覧表示します。

```
Get-EKSNodegroupList -ClusterName PROD
```

出力:

```
 ProdEKSNodeGroup
```

- API の詳細については、「 コマンドレットリファレンスListNodegroups」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSResourceTag

次の例は、Get-EKSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS リソースのタグを一覧表示します。

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

出力:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSUpdate

次の例は、Get-EKSUpdate を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS クラスターまたは関連するマネージド型ノードグループに対する更新に関する説明情報を返します。

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

出力:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- API の詳細については、「 コマンドレットリファレンスDescribeUpdate」の「」を参照してください。 AWS Tools for PowerShell

Get-EKSUpdateList

次の例は、Get-EKSUpdateList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、指定したリージョンの の Amazon EKS クラスターまたはマネージド型ノードグループに関連付けられた更新 AWS アカウントを一覧表示します。

```
Get-EKSUpdateList -Name "PROD"
```

出力:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- API の詳細については、「 コマンドレットリファレンスListUpdates」の「」を参照してください。 AWS Tools for PowerShell

New-EKSCluster

次の例は、New-EKSCluster を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「prod」という名前の新しいクラスターを作成します。

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateCluster」の「」を参照してください。 AWS Tools for PowerShell

New-EKSFargateProfile

次の例は、New-EKSFargateProfile を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS クラスターの AWS Fargate プロファイルを作成します。Fargate インフラストラクチャでポッドをスケジュールするには、クラスターに少なくとも 1 つの Fargate プロファイルが必要です。

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

出力:

```
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                : {}
```

- API の詳細については、「 コマンドレットリファレンスCreateFargateProfile」の「」を参照してください。 AWS Tools for PowerShell

New-EKSNodeGroup

次の例は、New-EKSNodeGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS クラスターのマネージドワーカーノードグループを作成します。クラスターの現在の Kubernetes バージョンと同じクラスター対してのみノードグループを作成できます。すべてのノードグループは、クラスターのそれぞれのマイナー Kubernetes バージョンの最新の AMI リリースバージョンで作成されます。

```
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"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateNodegroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-EKSCluster

次の例は、Remove-EKSCluster を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは Amazon EKS クラスターコントロールプレーンを削除します。

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

出力:

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

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
```

```
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- API の詳細については、「 コマンドレットリファレンスDeleteCluster」の「」を参照してください。 AWS Tools for PowerShell

Remove-EKSFargateProfile

次の例は、Remove-EKSFargateProfile を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは AWS Fargate プロファイルを削除します。Fargate プロファイルを削除すると、プロファイルで作成された Fargate で実行されているポッドはすべて削除されます。

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

出力:

```
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                : {}
```

- API の詳細については、「 コマンドレットリファレンスDeleteFargateProfile」の「」を参照してください。 AWS Tools for PowerShell

Remove-EKSNodegroup

次の例は、Remove-EKSNodegroup を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、クラスターの Amazon EKS ノードグループを削除します。

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteNodegroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-EKSResourceTag

次の例は、Remove-EKSResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、EKS リソースから指定されたタグを削除します。

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUntagResource」の「」を参照してください。 AWS Tools for PowerShell

Update-EKSClusterConfig

次の例は、Update-EKSClusterConfig を使用する方法を説明しています。

のツール PowerShell

例 1: Amazon EKS クラスター設定を更新します。更新中もクラスターは引き続き機能します。

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

出力:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- API の詳細については、「 コマンドレットリファレンスUpdateClusterConfig」の「」を参照してください。 AWS Tools for PowerShell

Update-EKSClusterVersion

次の例は、Update-EKSClusterVersion を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドレットは、Amazon EKS クラスターを指定された Kubernetes バージョンに更新します。更新中もクラスターは引き続き機能します。

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateClusterVersion」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Elastic Load Balancing - バージョン 1 の例 PowerShell

次のコード例は、Elastic Load Balancing - バージョン 1 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Add-ELBLoadBalancerToSubnet

次の例は、Add-ELBLoadBalancerToSubnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたサブネットを、指定されたロードバランサー用に設定された一連のサブネットに追加します。出力には、サブネットの完全なリストが含まれます。

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

出力:

```
subnet-12345678
subnet-87654321
```

- API の詳細については、「 コマンドレットリファレンスAttachLoadBalancerToSubnets」の「」を参照してください。 AWS Tools for PowerShell

Add-ELBResourceTag

次の例は、Add-ELBResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたタグを指定されたロードバランサーに追加します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

例 2: PowerShell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- API の詳細については、「 コマンドレットリファレンスAddTags」の「」を参照してください。 AWS Tools for PowerShell

`Disable-ELBAvailabilityZoneForLoadBalancer`

次の例は、`Disable-ELBAvailabilityZoneForLoadBalancer` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアベイラビリティーゾーンを指定されたロードバランサーから削除します。出力には残りのアベイラビリティーゾーンが含まれます。

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

出力:

```
us-west-2b
```

- API の詳細については、「 コマンドレットリファレンスDisableAvailabilityZonesForLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Dismount-ELBLoadBalancerFromSubnet

次の例は、Dismount-ELBLoadBalancerFromSubnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーに設定された一連のサブネットから指定されたサブネットを削除します。出力には残りのサブネットが含まれます。

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

出力:

```
subnet-87654321
```

- API の詳細については、「 コマンドレットリファレンスDetachLoadBalancerFromSubnets」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELBLoadBalancerAttribute

次の例は、Edit-ELBLoadBalancerAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーのクロスゾーン負荷分散を有効にします。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

例 2: この例では、指定されたロードバランサーの接続ドレインを無効にします。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

例 3: この例では、指定されたロードバランサーのアクセスログ記録を有効にします。

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName my-logs-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
```

```
>> -AccessLog_EmitInterval 60
```

- API の詳細については、「 コマンドレットリファレンスModifyLoadBalancerAttributes」の「」を参照してください。 AWS Tools for PowerShell

Enable-ELBAvailabilityZoneForLoadBalancer

次の例は、Enable-ELBAvailabilityZoneForLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアベイラビリティーゾーンを指定されたロードバランサーに追加します。出力には、アベイラビリティーゾーンの完全なリストが含まれます。

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

出力:

```
us-west-2a
us-west-2b
```

- API の詳細については、「 コマンドレットリファレンスEnableAvailabilityZonesForLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBInstanceHealth

次の例は、Get-ELBInstanceHealth を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーに登録されているインスタンスの状態について説明します。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

出力:

```
Description                    InstanceId                     ReasonCode
      State
-----------                    ----------                     ----------
      -----
N/A                            i-87654321                     N/A
      InService
Instance has failed at lea... i-12345678                     Instance
      OutOfService
```

例 2: この例では、指定されたロードバランサーに登録された指定されたインスタンスの状態について説明します。

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

例 3: この例では、指定されたインスタンスの状態の完全な説明を表示します。

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

出力:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstanceHealth」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBLoadBalancer

次の例は、Get-ELBLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ロードバランサーの名前を一覧表示します。

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

出力:

```
LoadBalancerName
```

```
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

例 2: この例では、指定されたロードバランサーについて説明します。

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

出力:

```
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
```

例 3: この例では、現在の AWS リージョンのすべてのロードバランサーについて説明します。

```
Get-ELBLoadBalancer
```

例 4: この例では、使用可能なすべての のすべてのロードバランサーについて説明します AWS リージョン。

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBLoadBalancerAttribute

次の例は、Get-ELBLoadBalancerAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーの属性について説明します。

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

出力:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancerAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBLoadBalancerPolicy

次の例は、Get-ELBLoadBalancerPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーに関連付けられたポリシーについて説明します。

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

出力:

```
PolicyAttributeDescriptions            PolicyName
 PolicyTypeName
---------------------------            ----------
 --------------
{ProxyProtocol}                        my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                           my-app-cookie-policy
 AppCookieStickinessPolicyType
```

例 2: この例では、指定されたポリシーの属性について説明します。

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

出力:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

例 3: この例では、サンプルポリシーを含む事前定義されたポリシーについて説明します。サンプルポリシーの名前には、ELBSample - プレフィックスが付いています。

```
Get-ELBLoadBalancerPolicy
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancerPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBLoadBalancerPolicyType

次の例は、Get-ELBLoadBalancerPolicyType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Elastic Load Balancing でサポートされているポリシータイプを取得します。

```
Get-ELBLoadBalancerPolicyType
```

出力:

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

例 2: この例では、指定されたポリシータイプについて説明します。

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

出力:

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

例 3: この例では、指定されたポリシータイプの詳細な説明を表示します。

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancerPolicyTypes」の「」を参照してください。 AWS Tools for PowerShell

Get-ELBResourceTag

次の例は、Get-ELBResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーのタグを一覧表示します。

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

出力:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

例 2: この例では、指定されたロードバランサーのタグについて説明します。

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

出力:

```
Key           Value
---           -----
project       lima
```

```
department    digital-media
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

Join-ELBSecurityGroupToLoadBalancer

次の例は、Join-ELBSecurityGroupToLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーの現在のセキュリティグループを指定されたセキュリティグループに置き換えます。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

出力:

```
sg-87654321
```

例 2: 現在のセキュリティグループを保持し、追加のセキュリティグループを指定するには、既存のセキュリティグループと新しいセキュリティグループの両方を指定します。

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

出力:

```
sg-12345678
sg-87654321
```

- API の詳細については、「 コマンドレットリファレンスApplySecurityGroupsToLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

New-ELBAppCookieStickinessPolicy

次の例は、New-ELBAppCookieStickinessPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたアプリケーション生成 Cookie のスティッキーセッションの有効期間に従う維持ポリシーを作成します。

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-app-cookie-policy -CookieName my-app-cookie
```

- API の詳細については、「 コマンドレットリファレンスCreateAppCookieStickinessPolicy」の「」を参照してください。 AWS Tools for PowerShell

New-ELBLBCookieStickinessPolicy

次の例は、New-ELBLBCookieStickinessPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された有効期限 (秒単位) によって制御されるスティッキーセッションの有効期間を持つ維持ポリシーを作成します。

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy -CookieExpirationPeriod 60
```

例 2: この例では、ブラウザ (user-agent) の有効期間によって によって制御されるスティッキーセッションの有効期間を持つ維持ポリシーを作成します。

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- API の詳細については、「 コマンドレットリファレンスCreateLbCookieStickinessPolicy」の「」を参照してください。 AWS Tools for PowerShell

New-ELBLoadBalancer

次の例は、New-ELBLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、VPC に HTTP リスナーを持つロードバランサーを作成します。

```
$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
```

例 2: この例では、EC2-Classic に HTTP リスナーを持つロードバランサーを作成します。

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

出力:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

例 3: この例では、HTTPS リスナーを使用してロードバランサーを作成します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

New-ELBLoadBalancerListener

次の例は、New-ELBLoadBalancerListener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したロードバランサーに HTTPS リスナーを追加します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateLoadBalancerListeners」の「」を参照してください。 AWS Tools for PowerShell

New-ELBLoadBalancerPolicy

次の例は、New-ELBLoadBalancerPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーの新しいプロキシプロトコルポリシーを作成します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateLoadBalancerPolicy」の「」を参照してください。 AWS Tools for PowerShell

Register-ELBInstanceWithLoadBalancer

次の例は、Register-ELBInstanceWithLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された EC2 インスタンスを指定されたロードバランサーに登録します。

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

出力:

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

- API の詳細については、「 コマンドレットリファレンスRegisterInstancesWithLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELBInstanceFromLoadBalancer

次の例は、Remove-ELBInstanceFromLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された EC2 インスタンスを指定されたロードバランサーから削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeregisterInstancesFromLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELBLoadBalancer

次の例は、Remove-ELBLoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELBLoadBalancerListener

次の例は、Remove-ELBLoadBalancerListener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーのポート 80 のリスナーを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteLoadBalancerListeners」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELBLoadBalancerPolicy

次の例は、Remove-ELBLoadBalancerPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーから指定されたポリシーを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

出力:

```
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"):
```

- API の詳細については、「 コマンドレットリファレンスDeleteLoadBalancerPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELBResourceTag

次の例は、Remove-ELBResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーから指定されたタグを削除します。Force パラメータも指定しない限り、操作が進む前に確認を求められます。この例で使用される構文には、PowerShell バージョン 3 以降が必要です。

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

出力:

```
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"):
```

例 2: Powershell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- API の詳細については、「 コマンドレットリファレンスRemoveTags」の「」を参照してください。 AWS Tools for PowerShell

Set-ELBHealthCheck

次の例は、Set-ELBHealthCheck を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーのヘルスチェック設定を構成します。

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
```

```
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

出力:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- API の詳細については、「 コマンドレットリファレンスConfigureHealthCheck」の「」を参照してください。 AWS Tools for PowerShell

Set-ELBLoadBalancerListenerSSLCertificate

次の例は、Set-ELBLoadBalancerListenerSSLCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーの SSL 接続を終了する証明書を置き換えます。

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- API の詳細については、「 コマンドレットリファレンスSetLoadBalancerListenerSslCertificate」の「」を参照してください。 AWS Tools for PowerShell

Set-ELBLoadBalancerPolicyForBackendServer

次の例は、Set-ELBLoadBalancerPolicyForBackendServer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたポートのポリシーを指定されたポリシーに置き換えます。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

例 2: この例では、指定されたポートに関連付けられているすべてのポリシーを削除します。

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- API の詳細については、「 コマンドレットリファレンスSetLoadBalancerPoliciesForBackendServer」の「」を参照してください。 AWS Tools for PowerShell

`Set-ELBLoadBalancerPolicyOfListener`

次の例は、`Set-ELBLoadBalancerPolicyOfListener` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーのポリシーを指定されたポリシーに置き換えます。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

例 2: この例では、指定されたリスナーに関連付けられているすべてのポリシーを削除します。

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- API の詳細については、「 コマンドレットリファレンスSetLoadBalancerPoliciesOfListener」の「」を参照してください。 AWS Tools for PowerShell

Elastic Load Balancing - Tools for を使用したバージョン 2 の例 PowerShell

次のコード例は、Elastic Load Balancing - バージョン 2 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

Add-ELB2ListenerCertificate

次の例は、Add-ELB2ListenerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーに証明書を追加します。

```
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'}
```

出力:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- API の詳細については、「 コマンドレットリファレンスAddListenerCertificates」の「」を参照してください。 AWS Tools for PowerShell

Add-ELB2Tag

次の例は、Add-ELB2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された**AWS.Tools.ElasticLoadBalancingV2**リソースに新しいタグを追加します。

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- API の詳細については、「 コマンドレットリファレンスAddTags」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELB2Listener

次の例は、Edit-ELB2Listener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーのデフォルトアクションを固定レスポンスに変更します。

```
$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
```

出力:

```
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       :
```

- API の詳細については、「 コマンドレットリファレンスModifyListener」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELB2LoadBalancerAttribute

次の例は、Edit-ELB2LoadBalancerAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーの属性を変更します。

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスModifyLoadBalancerAttributes」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELB2Rule

次の例は、Edit-ELB2Rule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナールール設定を変更します。

```
$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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスModifyRule」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELB2TargetGroup

次の例は、Edit-ELB2TargetGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループのプロパティを変更します。

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスModifyTargetGroup」の「」を参照してください。 AWS Tools for PowerShell

Edit-ELB2TargetGroupAttribute

次の例は、Edit-ELB2TargetGroupAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループの deregistration_delay 属性を変更します。

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスModifyTargetGroupAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2AccountLimit

次の例は、Get-ELB2AccountLimit を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、特定のリージョンの ELB2 アカウント制限を一覧表示します。

```
Get-ELB2AccountLimit
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAccountLimits」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2Listener

次の例は、Get-ELB2Listener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された ALB/NLB のリスナーについて説明します。

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeListeners」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2ListenerCertificate

次の例は、Get-ELB2ListenerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーの証明書について説明します。

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

出力:

```
CertificateArn
 IsDefault
--------------
 ---------
```

```
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- API の詳細については、「 コマンドレットリファレンスDescribeListenerCertificates」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2LoadBalancer

次の例は、Get-ELB2LoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、特定のリージョンのすべてのロードバランサーを表示します。

```
Get-ELB2LoadBalancer
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2LoadBalancerAttribute

次の例は、Get-ELB2LoadBalancerAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、特定のロードバランサーの属性を記述します。

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeLoadBalancerAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2Rule

次の例は、Get-ELB2Rule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナー ARN のリスナールールについて説明します。

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeRules」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2SSLPolicy

次の例は、Get-ELB2SSLPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 ElasticLoadBalancingV2 で使用できるすべてのリスナーポリシーを一覧表示します。

```
Get-ELB2SSLPolicy
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスDescribeSslPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2Tag

次の例は、Get-ELB2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースのタグを一覧表示します。

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

出力:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2TargetGroup

次の例は、Get-ELB2TargetGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループについて説明します。

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeTargetGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2TargetGroupAttribute

次の例は、Get-ELB2TargetGroupAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループの属性について説明します。

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeTargetGroupAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-ELB2TargetHealth

次の例は、Get-ELB2TargetHealth を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループに存在するターゲットのヘルスステータスを返します。

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

出力:

```
HealthCheckPort Target                                                TargetHealth
--------------- ------                                                ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- API の詳細については、「 コマンドレットリファレンスDescribeTargetHealth」の「」を参照してください。 AWS Tools for PowerShell

New-ELB2Listener

次の例は、New-ELB2Listener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、デフォルトのアクション「転送」を使用して新しい ALB リスナーを作成し、指定したターゲットグループにトラフィックを送信します。

```
$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
```

出力:

```
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       :
```

- API の詳細については、「 コマンドレットリファレンスCreateListener」の「」を参照してください。 AWS Tools for PowerShell

New-ELB2LoadBalancer

次の例は、New-ELB2LoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、2 つのサブネットを持つ新しいインターネット向け Application Load Balancer を作成します。

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

出力:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- API の詳細については、「 コマンドレットリファレンスCreateLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

New-ELB2Rule

次の例は、New-ELB2Rule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーの顧客ヘッダー値に基づいて、固定レスポンスアクションを持つ新しいリスナールールを作成します。

```
$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
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateRule」の「」を参照してください。 AWS Tools for PowerShell

New-ELB2TargetGroup

次の例は、New-ELB2TargetGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたパラメータを使用して新しいターゲットグループを作成します。

```
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'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateTargetGroup」の「」を参照してください。 AWS Tools for PowerShell

Register-ELB2Target

次の例は、Register-ELB2Target を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、「i-0672a4c4cdeae3111」インスタンスを指定されたターゲットグループに登録します。

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- API の詳細については、「 コマンドレットリファレンスRegisterTargets」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2Listener

次の例は、Remove-ELB2Listener を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナーを削除します。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

出力:

```
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
```

例 2: この例では、指定されたリスナーをロードバランサーから削除します。

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteListener」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2ListenerCertificate

次の例は、Remove-ELB2ListenerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループから指定された証明書を削除します。

```
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'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスRemoveListenerCertificates」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2LoadBalancer

次の例は、Remove-ELB2LoadBalancer を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーを削除します。

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteLoadBalancer」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2Rule

次の例は、Remove-ELB2Rule を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リスナーから指定されたルールを削除します。

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteRule」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2Tag

次の例は、Remove-ELB2Tag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキーの タグを削除します。

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスRemoveTags」の「」を参照してください。 AWS Tools for PowerShell

Remove-ELB2TargetGroup

次の例は、Remove-ELB2TargetGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループを削除します。

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssss/4e0b6076bc6483a7'
```

出力:

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

```
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testssssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API の詳細については、「 コマンドレットリファレンスDeleteTargetGroup」の「」を参照してください。 AWS Tools for PowerShell

Set-ELB2IpAddressType

次の例は、Set-ELB2IpAddressType を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ロードバランサーの IP アドレスタイプをIPv4」から「」に変更DualStackします。

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

出力:

```
Value
-----
dualstack
```

- API の詳細については、「 コマンドレットリファレンスSetIpAddressType」の「」を参照してください。 AWS Tools for PowerShell

Set-ELB2RulePriority

次の例は、Set-ELB2RulePriority を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリスナールールの優先度を変更します。

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスSetRulePriorities」の「」を参照してください。 AWS Tools for PowerShell

Set-ELB2SecurityGroup

次の例は、Set-ELB2SecurityGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、セキュリティグループ「sg-07c3414abb8811cbd」を指定されたロードバランサーに追加します。

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

出力:

```
sg-07c3414abb8811cbd
```

- API の詳細については、「 コマンドレットリファレンスSetSecurityGroups」の「」を参照してください。 AWS Tools for PowerShell

Set-ELB2Subnet

次の例は、Set-ELB2Subnet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたロードバランサーのサブネットを変更します。

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

出力:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- API の詳細については、「 コマンドレットリファレンスSetSubnets」の「」を参照してください。 AWS Tools for PowerShell

`Unregister-ELB2Target`

次の例は、`Unregister-ELB2Target` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたターゲットグループからインスタンス 'i-0672a4c4cdeae3111' を登録解除します。

```
$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'
```

- API の詳細については、「 コマンドレットリファレンスDeregisterTargets」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon FSx の例 PowerShell

次のコード例は、Amazon FSx AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-FSXResourceTag`

次の例は、`Add-FSXResourceTag` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースにタグを追加します。

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

出力:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- API の詳細については、「 コマンドレットリファレンスTagResource」の「」を参照してください。 AWS Tools for PowerShell

Get-FSXBackup

次の例は、Get-FSXBackup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、昨日以降に作成された特定のファイルシステム ID のバックアップを取得します。

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeBackups」の「」を参照してください。 AWS Tools for PowerShell

Get-FSXFileSystem

次の例は、Get-FSXFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された filesystemId の説明を返します。

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeFileSystems」の「」を参照してください。 AWS Tools for PowerShell

Get-FSXResourceTagList

次の例は、Get-FSXResourceTagList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、提供されたリソース ARN のタグを一覧表示します。

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

出力:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

New-FSXBackup

次の例は、New-FSXBackup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のファイルシステムのバックアップを作成します。

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateBackup」の「」を参照してください。 AWS Tools for PowerShell

New-FSXFileSystem

次の例は、New-FSXFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したサブネットからのアクセスを許可する新しい 300GB Windows ファイルシステムを作成し、1 秒あたり最大 8 メガバイトのスループットをサポートします。新しいファイルシステムは、指定された Microsoft Active Directory に自動的に結合されます。

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateFileSystem」の「」を参照してください。 AWS Tools for PowerShell

New-FSXFileSystemFromBackup

次の例は、New-FSXFileSystemFromBackup を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、既存の Amazon FSx for Windows File Server バックアップから新しい Amazon FSx ファイルシステムを作成します。

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateFileSystemFromBackup」の「」を参照してください。 AWS Tools for PowerShell

Remove-FSXBackup

次の例は、Remove-FSXBackup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された backup-id を削除します。

```
Remove-FSXBackup -BackupId $backupID
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteBackup」の「」を参照してください。 AWS Tools for PowerShell

Remove-FSXFileSystem

次の例は、Remove-FSXFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された FSX ファイルシステム ID を削除します。

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteFileSystem」の「」を参照してください。 AWS Tools for PowerShell

Remove-FSXResourceTag

次の例は、Remove-FSXResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された FSX ファイルシステムリソース ARN のリソースタグを削除します。

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUntagResource」の「」を参照してください。 AWS Tools for PowerShell

Update-FSXFileSystem

次の例は、Update-FSXFileSystem を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、FSX ファイルシステムの自動バックアップ保持日数を で更新します UpdateFileSystemWindowsConfiguration。

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
 $UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
 Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
 $UpdateFSXWinConfig
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateFileSystem」の「」を参照してください。 AWS Tools for PowerShell

AWS Glue Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Glue。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

New-GLUEJob

次の例は、New-GLUEJob を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS Glue で新しいジョブを作成します。コマンド名の値は常に ですglueetl。 AWS Glue は Python または Scala で記述されたジョブスクリプトの実行をサポートしています。この例では、ジョブスクリプト (MyTestGlueJob.py) は Python で記述されています。Python パラメータは $DefArgs変数で指定され、 DefaultArgumentsパラメータの PowerShell コマンドに渡されます。このパラメータはハッシュテーブルを受け入れ

ます。**$JobParams**変数のパラメータは、 AWS Glue CreateJob API リファレンスの「Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html)」トピックに記載されている API から取得されます。

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateJob」の「」を参照してください。 AWS Tools for PowerShell

AWS Health Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Health。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-HLTHEvent`

次の例は、`Get-HLTHEvent` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは AWS 、Personal Health Dashboard からイベントを返します。ユーザーは -Region パラメータを追加して、米国東部 (バージニア北部) リージョンのサービスで利用可能なイベントを表示しますが、-Filter_Region パラメータは、欧州 (ロンドン) および米国西部 (オレゴン) リージョン (eu-west-2 および us-west-2) に記録されたイベントをフィルタリングします。-Filter_StartTime parameter はイベントを開始できる時間の範囲をフィルタリングし、-Filter_EndTime parameter はイベントを終了できる時間の範囲をフィルタリングします。その結果、RDS のスケジュールされたメンテナンスイベントが、指定された -Filter_StartTime range 内で開始され、スケジュールされた -Filter_EndTime range 内で終了します。

```
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"}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeEvents」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した IAM の例 PowerShell

次のコード例は、IAM AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-IAMClientIDToOpenIDConnectProvider`

次の例は、`Add-IAMClientIDToOpenIDConnectProvider` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、クライアント ID (または対象者) **my-application-ID** を **server.example.com** という名前の既存の OIDC プロバイダーに追加します。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- API の詳細については、「 コマンドレットリファレンスAddClientIdToOpenIdConnectProvider」の「」を参照してください。 AWS Tools for PowerShell

Add-IAMRoleTag

次の例は、Add-IAMRoleTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ID 管理サービスのロールにタグを追加します。

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API の詳細については、「 コマンドレットリファレンスTagRole」の「」を参照してください。 AWS Tools for PowerShell

Add-IAMRoleToInstanceProfile

次の例は、Add-IAMRoleToInstanceProfile を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、**S3Access** という名前のロールを **webserver** という名前の既存のインスタンスプロファイルに追加します。インスタンスプロファイルを作成するには、**New-IAMInstanceProfile** コマンドを使用します。このコマンドを使用してインスタンスプロファイルを作成し、ロールに関連付けると、EC2 インスタンスにアタッチできます。そのためには、**New-EC2Instance** コマンドレットを **InstanceProfile_Arn** または **InstanceProfile-Name** パラメータと共に使用して、新しいインスタンスを起動します。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- API の詳細については、「 コマンドレットリファレンスAddRoleToInstanceProfile」の「」を参照してください。 AWS Tools for PowerShell

Add-IAMUserTag

次の例は、Add-IAMUserTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ID 管理サービスのユーザーにタグを追加します。

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API の詳細については、「 コマンドレットリファレンスTagUser」の「」を参照してください。 AWS Tools for PowerShell

Add-IAMUserToGroup

次の例は、Add-IAMUserToGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、**Bob** という名前のユーザーを **Admins** という名前のグループに追加します。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API の詳細については、「 コマンドレットリファレンスAddUserToGroup」の「」を参照してください。 AWS Tools for PowerShell

Disable-IAMMFADevice

次の例は、Disable-IAMMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、シリアル番号 **123456789012** を持つユーザー **Bob** に関連付けられているハードウェア MFA デバイスを無効にします。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

例 2: このコマンドは、ARN **arn:aws:iam::210987654321:mfa/David** を持つユーザー **David** に関連付けられている仮想 MFA デバイスを無効にします。仮想 MFA デバイスはアカウントから削除されないことに注意してください。仮想デバイスはまだ存在し、**Get-IAMVirtualMFADevice** コマンドの出力に表示されます。同じユーザーに対して新しい仮想 MFA デバイスを作成する前に、**Remove-IAMVirtualMFADevice** コマンドを使用して、古い仮想 MFA デバイスを削除する必要があります。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- API の詳細については、「 コマンドレットリファレンスDeactivateMfaDevice」の「」を参照してください。 AWS Tools for PowerShell

Edit-IAMPassword

次の例は、Edit-IAMPassword を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、コマンドを実行しているユーザーのパスワードを変更します。このコマンドは、IAM ユーザーのみが呼び出すことができます。 AWS アカウント (ルート) 認証情報でサインインしたときにこのコマンドが呼び出されると、コマンドは**InvalidUserType**エラーを返します。

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API の詳細については、「 コマンドレットリファレンスChangePassword」の「」を参照してください。 AWS Tools for PowerShell

Enable-IAMMFADevice

次の例は、Enable-IAMMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、シリアル番号 **987654321098** を持つハードウェア MFA デバイスを有効にし、このデバイスをユーザー **Bob** に関連付けます。また、このコマンドは、デバイスからの最初の 2 つのコードを順番に含めます。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

例 2: この例では、仮想 MFA デバイスを作成して有効にします。最初のコマンドは仮想デバイスを作成し、そのデバイスのオブジェクト表現を変数 **$MFADevice** に返します。**.Base32StringSeed** または **QRCodePng** プロパティを使用して、ユーザーのソフトウェアアプリケーションを設定できます。最後のコマンドはデバイスをユーザー **David** に割り当て、デバイスをシリアル番号で識別します。コマンドは、仮想 MFA デバイスからの最初の 2 つのコードを順番に含める AWS ことで、デバイスを と同期します。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- API の詳細については、「 コマンドレットリファレンスEnableMfaDevice」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccessKey

次の例は、Get-IAMAccessKey を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、**Bob** という名前の IAM ユーザーのアクセスキーを一覧表示します。IAM ユーザーのシークレットアクセスキーを一覧表示できないことに注意してください。シークレットアクセスキーを紛失した場合は、**New-IAMAccessKey** コマンドレットを使用して新しいアクセスキーを作成する必要があります。

```
Get-IAMAccessKey -UserName "Bob"
```

出力:

```
AccessKeyId               CreateDate                 Status             UserName
-----------               ----------                 ------             --------
AKIAIOSFODNN7EXAMPLE      12/3/2014 10:53:41 AM      Active             Bob
```

```
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM          Inactive             Bob
```

- API の詳細については、「 コマンドレットリファレンスListAccessKeys」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccessKeyLastUsed

次の例は、Get-IAMAccessKeyLastUsed を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたアクセスキーの所有ユーザー名と最終使用情報を返します。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスGetAccessKeyLastUsed」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccountAlias

次の例は、Get-IAMAccountAlias を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、 AWS アカウントのアカウントエイリアスを返します。

```
Get-IAMAccountAlias
```

出力:

```
ExampleCo
```

- API の詳細については、「 コマンドレットリファレンスListAccountAliases」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccountAuthorizationDetail

次の例は、Get-IAMAccountAuthorizationDetail を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウントの ID に関する認証の詳細を取得し、ユーザー、グループ、ロールなど、返されたオブジェクトの要素リストを表示します。例えば、**UserDetailList** プロパティには、ユーザーに関する詳細が表示されます。同様の情報は、**RoleDetailList** および **GroupDetailList** プロパティで入手可能です。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

出力:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスGetAccountAuthorizationDetails」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccountPasswordPolicy

次の例は、Get-IAMAccountPasswordPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在のアカウントのパスワードポリシーに関する詳細を返します。アカウントのためにパスワードポリシーが定義されていない場合、コマンドは **NoSuchEntity** エラーを返します。

```
Get-IAMAccountPasswordPolicy
```

出力:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- API の詳細については、「 コマンドレットリファレンスGetAccountPasswordPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAccountSummary

次の例は、Get-IAMAccountSummary を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウント内の現在の IAM エンティティの使用状況と現在の IAM エンティティのクォータに関する情報を返します。

```
Get-IAMAccountSummary
```

出力:

```
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
```

```
...
```

- API の詳細については、「 コマンドレットリファレンスGetAccountSummary」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAttachedGroupPolicyList

次の例は、Get-IAMAttachedGroupPolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、 AWS アカウントAdmins内の という名前の ARNs を返します。グループに埋め込まれているインラインポリシーのリストを表示するには、Get-IAMGroupPolicyList コマンドを使用します。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

出力:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- API の詳細については、「 コマンドレットリファレンスListAttachedGroupPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAttachedRolePolicyList

次の例は、Get-IAMAttachedRolePolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、 AWS アカウントの SecurityAuditRole という名前の IAM ロールにアタッチされている管理ポリシーの名前と ARN を返します。ロールに埋め込まれているインラインポリシーのリストを表示するには、Get-IAMRolePolicyList コマンドを使用します。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

出力:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
```

- API の詳細については、「 コマンドレットリファレンスListAttachedRolePolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMAttachedUserPolicyList

次の例は、Get-IAMAttachedUserPolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、 AWS アカウント**Bob**で という名前ARNs を返します。IAM ユーザーに埋め込まれているインラインポリシーのリストを表示するには、**Get-IAMUserPolicyList** コマンドを使用します。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

出力:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/TesterPolicy                       TesterPolicy
```

- API の詳細については、「 コマンドレットリファレンスListAttachedUserPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMContextKeysForCustomPolicy

次の例は、Get-IAMContextKeysForCustomPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、提供されたポリシー JSON に含まれるすべてのコンテキストキーを取得します。複数のポリシーを指定するには、値のカンマ区切りリストとして指定できます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスGetContextKeysForCustomPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMContextKeysForPrincipalPolicy

次の例は、Get-IAMContextKeysForPrincipalPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、提供されたポリシー JSON に含まれるすべてのコンテキストキーと IAM エンティティ (ユーザー/ロールなど) にアタッチされたポリシーを取得します。の場合 - 複数の値リストをカンマ区切りの値として指定PolicyInputList できます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスGetContextKeysForPrincipalPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMCredentialReport

次の例は、Get-IAMCredentialReport を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、返されたレポートを開き、それをテキスト行の配列としてパイプラインに出力します。最初の行は、カンマで区切られた列名のヘッダーです。連続する各行は 1 人のユーザー

の詳細行で、各フィールドはカンマで区切られています。レポートを表示するには、**Request-IAMCredentialReport** コマンドレットを使用してレポートを生成する必要があります。レポートを 1 つの文字列として取得するには、**-AsTextArray** ではなく **-Raw** を使用します。**-AsTextArray** スイッチには、エイリアス **-SplitLines** も使用できます。出力の列の完全なリストについては、サービス API リファレンスを参照してください。**-AsTextArray** または **-SplitLines** を使用しない場合は、.NET **StreamReader** クラスを使用して、**.Content** プロパティからテキストを抽出する必要があることに注意してください。

```
Request-IAMCredentialReport
```

出力:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

出力:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,passw
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/
A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- API の詳細については、「 コマンドレットリファレンスGetCredentialReport」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMEntitiesForPolicy

次の例は、Get-IAMEntitiesForPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ポリシー **arn:aws:iam::123456789012:policy/TestPolicy** がアタッチされている IAM グループ、ロール、ユーザーのリストを返します。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

出力:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- API の詳細については、「 コマンドレットリファレンスListEntitiesForPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMGroup

次の例は、Get-IAMGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、グループに属するすべての IAM ユーザーのコレクションを含む、IAM グループ **Testers** に関する詳細を返します。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

出力:

```
Group                                    IsTruncated           Marker
  Users
-----                                    -----------           ------
  -----
Amazon.IdentityManagement.Model.Group    False
  {Theresa, David}
```

```
$results.Group
```

出力:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

出力:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- API の詳細については、「 コマンドレットリファレンスGetGroup」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMGroupForUser

次の例は、Get-IAMGroupForUser を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **David** が属する IAM グループのリストを返します。

```
Get-IAMGroupForUser -UserName David
```

出力:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- API の詳細については、「 コマンドレットリファレンスListGroupsForUser」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMGroupList

次の例は、Get-IAMGroupList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の で定義されているすべての IAM グループのコレクションを返します AWS アカウント。

```
Get-IAMGroupList
```

出力:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- API の詳細については、「 コマンドレットリファレンスListGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMGroupPolicy

次の例は、Get-IAMGroupPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、グループ **Testers** の **PowerUserAccess-Testers** という名前の埋め込みインラインポリシーに関する詳細を返します。**PolicyDocument** プロパティは URL エンコードされています。この例では、**UrlDecode** .NET メソッドを使用してデコードされています。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

出力:

```
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": "*"
    }
  ]
}
```

- API の詳細については、「 コマンドレットリファレンスGetGroupPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMGroupPolicyList

次の例は、Get-IAMGroupPolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、グループ **Testers** に埋め込まれているインラインポリシーのリストを返します。グループにアタッチされている管理ポリシーを取得するには、コマンド **Get-IAMAttachedGroupPolicyList** を使用します。

```
Get-IAMGroupPolicyList -GroupName Testers
```

出力:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- API の詳細については、「 コマンドレットリファレンスListGroupPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMInstanceProfile

次の例は、Get-IAMInstanceProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントで定義されている **ec2instancerole** という名前のインスタンスプロファイルの詳細を返します。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

出力:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API の詳細については、「 コマンドレットリファレンスGetInstanceProfile」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMInstanceProfileForRole

次の例は、Get-IAMInstanceProfileForRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ロール **ec2instancerole** に関連付けられているインスタンスプロファイルの詳細を返します。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

出力:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- API の詳細については、「 コマンドレットリファレンスListInstanceProfilesForRole」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMInstanceProfileList

次の例は、Get-IAMInstanceProfileList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の で定義されているインスタンスプロファイルのコレクションを返します AWS アカウント。

```
Get-IAMInstanceProfileList
```

出力:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API の詳細については、「 コマンドレットリファレンスListInstanceProfiles」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMLoginProfile

次の例は、Get-IAMLoginProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パスワードの作成日と、IAM ユーザー **David** のパスワードのリセットが必要かどうかを返します。

```
Get-IAMLoginProfile -UserName David
```

出力:

```
CreateDate                  PasswordResetRequired               UserName
----------                  ---------------------               --------
12/10/2014 3:39:44 PM       False                               David
```

- API の詳細については、「 コマンドレットリファレンスGetLoginProfile」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMMFADevice

次の例は、Get-IAMMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **David** に割り当てられた MFA デバイスに関する詳細を返します。この例では、**SerialNumber** は物理デバイスの実際のシリアル番号ではなく ARN であるため、仮想デバイスであることがわかります。

```
Get-IAMMFADevice -UserName David
```

出力:

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- API の詳細については、「 コマンドレットリファレンスListMfaDevices」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMOpenIDConnectProvider

次の例は、Get-IAMOpenIDConnectProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** である OpenID Connect プロバイダーに関する詳細を返します。**ClientIDList** プロパティは、このプロバイダーに定義されているすべてのクライアント ID を含むコレクションです。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

出力:

```
ClientIDList         CreateDate               ThumbprintList
        Url
------------         ----------               --------------
        ---
```

```
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- API の詳細については、「 コマンドレットリファレンスGetOpenIdConnectProvider」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMOpenIDConnectProviderList

次の例は、Get-IAMOpenIDConnectProviderList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントで定義されているすべての OpenID Connect プロバイダーの ARN のリストを返します。

```
Get-IAMOpenIDConnectProviderList
```

出力:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- API の詳細については、「 コマンドレットリファレンスListOpenIdConnectProviders」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMPolicy

次の例は、Get-IAMPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MySamplePolicy** である管理ポリシーに関する詳細を返します。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMPolicyList

次の例は、Get-IAMPolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントで使用可能な最初の 3 つの管理ポリシーのコレクションを返します。**-scope**が指定されていないため、デフォルトで に設定されます。これには、AWS 管理**all**ポリシーとカスタマー管理ポリシーの両方が含まれます。

```
Get-IAMPolicyList -MaxItem 3
```

出力:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
```

```
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

例 2: この例では、現在の AWS アカウントで使用可能な最初の 2 つのカスタマー管理ポリシーのコレクションを返します。**-Scope local** を使用して、出力をカスタマー管理ポリシーのみに制限します。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

出力:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
```

```
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- API の詳細については、「 コマンドレットリファレンスListPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMPolicyVersion

次の例は、Get-IAMPolicyVersion を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MyManagedPolicy** であるポリシーの **v2** バージョンのポリシードキュメントを返します。**Document** プロパティ内のポリシードキュメントは URL でエンコードされ、この例では **UrlDecode** .NET メソッドを使用してデコードされます。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

出力:

```
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": "*"
    }
}
```

- API の詳細については、「 コマンドレットリファレンスGetPolicyVersion」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMPolicyVersionList

次の例は、Get-IAMPolicyVersionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MyManagedPolicy** であるポリシーの使用可能なバージョンのリストを返します。特定のバージョンのポリシードキュメントを取得するには、**Get-IAMPolicyVersion** コマンドを使用して、必要なバージョンの **VersionId** を指定します。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

出力:

```
CreateDate                  Document                IsDefaultVersion
    VersionId
----------                  --------                ----------------
    ---------
2/12/2015 9:39:53 AM                                True
    v2
2/12/2015 9:39:09 AM                                False
    v1
```

- API の詳細については、「 コマンドレットリファレンスListPolicyVersions」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMRole

次の例は、Get-IAMRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**lamda_exec_role** の詳細を返します。これには、このロールの引き受け先を指定する信頼ポリシードキュメントが含まれています。ポリシードキュメントは URL でエンコードされており、.NET **UrlDecode** メソッドを使用してデコードできます。この例では、元のポリシーは、ポリシーにアップロードされる前にすべての空白が削除されています。ロールを引き受けるユーザーが実行できる操作を決定するアクセス許可ポリシードキュメントを確認するには、インラインポリシーには **Get-IAMRolePolicy** を使用し、アタッチされた管理ポリシーには **Get-IAMPolicyVersion** を使用します。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

出力:

```
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
```

出力:

```
{"Version":"2012-10-17","Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- API の詳細については、「 コマンドレットリファレンスGetRole」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMRoleList

次の例は、Get-IAMRoleList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウント内のすべての IAM ロールのリストを取得します。

```
Get-IAMRoleList
```

例 2: このサンプルコードスニペットは、 AWS アカウント内の IAM ロールのリストを取得し、それらを一度に 3 つ表示し、各グループ間で Enter キーを押すのを待ちます。前の呼び出しの **Marker** 値を渡して、次のグループの開始位置を指定します。

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスListRoles」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMRolePolicy

次の例は、Get-IAMRolePolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ロール **lamda_exec_role** に埋め込まれている **oneClick_lambda_exec_role_policy** という名前のポリシーのアクセス許可ポリシードキュメントを返します。結果のポリシードキュメントは URL エンコードされます。この例では、**UrlDecode** .NET メソッドを使用してデコードされています。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

出力:

```
PolicyDocument                                                  PolicyName
           UserName
--------------                                                  ----------
           --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

出力:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- API の詳細については、「 コマンドレットリファレンスGetRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMRolePolicyList

次の例は、Get-IAMRolePolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ロール **lamda_exec_role** に埋め込まれているインラインポリシーの名前のリストを返します。インラインポリシーの詳細を表示するには、**Get-IAMRolePolicy** コマンドを使用します。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

出力:

```
oneClick_lambda_exec_role_policy
```

- API の詳細については、「 コマンドレットリファレンスListRolePolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMRoleTagList

次の例は、Get-IAMRoleTagList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ロールに関連付けられているタグを取得します。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API の詳細については、「 コマンドレットリファレンスListRoleTags」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMSAMLProvider

次の例は、Get-IAMSAMLProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARM が arn:aws:iam::123456789012:saml-provider/SAMLADFS である SAML 2.0 プロバイダーに関する詳細を取得します。レスポンスには、 AWS SAML プロバイダーエン

ティティを作成するために ID プロバイダーから取得したメタデータドキュメントと、作成日と有効期限が含まれます。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

出力:

```
CreateDate                 SAMLMetadataDocument
     ValidUntil
----------                 --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- API の詳細については、「 コマンドレットリファレンスGetSamlProvider」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMSAMLProviderList

次の例は、Get-IAMSAMLProviderList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントで作成された SAML 2.0 プロバイダーのリストを取得します。これは、各 SAML プロバイダーの ARN、作成日、有効期限を返します。

```
Get-IAMSAMLProviderList
```

出力:

```
Arn                                                  CreateDate
 ValidUntil
---                                                  ----------
 ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「ListSAMLProviders」を参照してください。

Get-IAMServerCertificate

次の例は、Get-IAMServerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyServerCertificate** という名前のサーバー証明書に関する詳細を取得します。証明書の詳細は、**CertificateBody** および **ServerCertificateMetadata** プロパティで確認できます。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

出力:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                           YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                           21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                           rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                           nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                           NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

```
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetServerCertificate」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMServerCertificateList

次の例は、Get-IAMServerCertificateList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントにアップロードされたサーバー証明書のリストを取得します。

```
Get-IAMServerCertificateList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListServerCertificates」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMServiceLastAccessedDetail

次の例は、Get-IAMServiceLastAccessedDetail を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リクエスト呼び出しに関連付けられた IAM エンティティ (ユーザー、グループ、ロール、またはポリシー) が最後にアクセスしたサービスの詳細が表示されます。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

出力:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- API の詳細については、「 コマンドレットリファレンスGetServiceLastAccessedDetails」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMServiceLastAccessedDetailWithEntity

次の例は、Get-IAMServiceLastAccessedDetailWithEntity を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、それぞれの IAM エンティティによるリクエスト内のサービスの最終アクセスタイムスタンプを提供します。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

出力:

```
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
```

出力:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

出力:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- API の詳細については、「 コマンドレットリファレンスGetServiceLastAccessedDetailsWithEntities」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMSigningCertificate

次の例は、Get-IAMSigningCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前のユーザーに関連付けられている署名証明書に関する詳細を取得します。

```
Get-IAMSigningCertificate -UserName Bob
```

出力:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- API の詳細については、「 コマンドレットリファレンスListSigningCertificates」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMUser

次の例は、Get-IAMUser を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**David** という名前のユーザーに関する詳細を取得します。

```
Get-IAMUser -UserName David
```

出力:

```
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
```

例 2: この例では、現在サインインしている IAM ユーザーに関する詳細を取得します。

```
Get-IAMUser
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetUser」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMUserList

次の例は、Get-IAMUserList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の のユーザーのコレクションを取得します AWS アカウント。

```
Get-IAMUserList
```

出力:

```
      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
```

- API の詳細については、「 コマンドレットリファレンスListUsers」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMUserPolicy

次の例は、Get-IAMUserPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**David** という名前の IAM ユーザーに埋め込まれている **Davids_IAM_Admin_Policy** という名前のインラインポリシーの詳細を取得します。ポリシードキュメントは URL エンコードされています。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

出力:

```
PolicyDocument                                                  PolicyName
    UserName
--------------                                                  ----------
    --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
    David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
```

```
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- API の詳細については、「 コマンドレットリファレンスGetUserPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMUserPolicyList

次の例は、Get-IAMUserPolicyList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**David** という名前の IAM ユーザーに埋め込まれているインラインポリシーの名前一覧を取得します。

```
Get-IAMUserPolicyList -UserName David
```

出力:

```
Davids_IAM_Admin_Policy
```

- API の詳細については、「 コマンドレットリファレンスListUserPolicies」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMUserTagList

次の例は、Get-IAMUserTagList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ユーザーに関連付けられているタグを取得します。

```
Get-IAMUserTagList -UserName joe
```

- API の詳細については、「 コマンドレットリファレンスListUserTags」の「」を参照してください。 AWS Tools for PowerShell

Get-IAMVirtualMFADevice

次の例は、Get-IAMVirtualMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS アカウントのユーザーに割り当てられた仮想 MFA デバイスのコレクションを取得します。それぞれの **User** プロパティは、デバイスが割り当てられている IAM ユーザーの詳細を含むオブジェクトです。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListVirtualMfaDevices」の「」を参照してください。 AWS Tools for PowerShell

New-IAMAccessKey

次の例は、New-IAMAccessKey を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しいアクセスキーとシークレットアクセスキーのペアを作成し、それをユーザー **David** に割り当てます。**SecretAccessKey** を取得できるのはこのときだけなので、**AccessKeyId** と **SecretAccessKey** の値は必ずファイルに保存してください。後で取得することはできません。シークレットアクセスキーを紛失した場合は、新しいアクセスキーペアを作成する必要があります。

```
New-IAMAccessKey -UserName David
```

出力:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- API の詳細については、「 コマンドレットリファレンスCreateAccessKey」の「」を参照してください。 AWS Tools for PowerShell

New-IAMAccountAlias

次の例は、New-IAMAccountAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントのアカウントエイリアスを AWS に変更します**mycompanyaws**。ユーザーログオンページのアドレスが、https://mycompanyaws.signin.aws.amazon.com/console に変わります。エイリアスの代わりにアカウント ID 番号を使用する元の URL (https://<accountidnumber>.signin.aws.amazon.com/console) は引き続き機能します。ただし、以前に定義したエイリアスベースの URL は機能しなくなります。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- API の詳細については、「 コマンドレットリファレンスCreateAccountAlias」の「」を参照してください。 AWS Tools for PowerShell

New-IAMGroup

次の例は、New-IAMGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Developers** という名前の新しい IAM グループを作成します。

```
New-IAMGroup -GroupName Developers
```

出力:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- API の詳細については、「 コマンドレットリファレンスCreateGroup」の「」を参照してください。 AWS Tools for PowerShell

New-IAMInstanceProfile

次の例は、New-IAMInstanceProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**ProfileForDevEC2Instance** という名前の新しい IAM インスタンスプロファイルを作成します。**Add-IAMRoleToInstanceProfile** コマンドを個別に実行して、インスタンスにアクセス許可を与える既存の IAM ロールにインスタンスプロファイルを関連付ける必要があります。最後に、EC2 インスタンスを起動する際に、インスタンスプロファイルを EC2 インスタンスにアタッチします。そのためには、**New-EC2Instance** コマンドレットを **InstanceProfile_Arn** または **InstanceProfile_Name** パラメータと共に使用します。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

出力:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
```

```
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- API の詳細については、「 コマンドレットリファレンスCreateInstanceProfile」の「」を参照してください。 AWS Tools for PowerShell

New-IAMLoginProfile

次の例は、New-IAMLoginProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Bob という名前の IAM ユーザーの (一時的な) パスワードを作成し、次回 **Bob** がサインインしたときに、ユーザーにパスワードを変更するように要求するフラグを設定しています。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

出力:

```
CreateDate                  PasswordResetRequired               UserName
----------                  ---------------------               --------
4/14/2015 12:26:30 PM       True                                Bob
```

- API の詳細については、「 コマンドレットリファレンスCreateLoginProfile」の「」を参照してください。 AWS Tools for PowerShell

New-IAMOpenIDConnectProvider

次の例は、New-IAMOpenIDConnectProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、URL **https://example.oidcprovider.com** とクライアント ID **my-testapp-1** にある OIDC 互換プロバイダーサービスに関連付けられた IAM OIDC プロバイダーを作成します。OIDC プロバイダーがサムプリントを提供します。サムプリントを認証

するには、http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html の手順に従ってください。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

出力:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API の詳細については、「 コマンドレットリファレンスCreateOpenIdConnectProvider」の「」を参照してください。 AWS Tools for PowerShell

New-IAMPolicy

次の例は、New-IAMPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の AWS アカウントに新しい IAM **MySamplePolicy** ポリシーを作成します。このファイルはポリシーコンテンツ**MySamplePolicy.json**を提供します。JSON ポリシーファイルを正常に処理するには、**-Raw** switch パラメータを使用する必要があることに注意してください。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreatePolicy」の「」を参照してください。 AWS Tools for PowerShell

New-IAMPolicyVersion

次の例は、New-IAMPolicyVersion を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MyPolicy** である IAM ポリシーの新しい「v2」バージョンを作成し、それをデフォルトのバージョンにします。**NewPolicyVersion.json** ファイルは、ポリシーの内容を提供します。JSON ポリシーファイルを正常に処理するには、**-Raw** switch パラメータを使用する必要があることに注意してください。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

出力:

```
CreateDate                              Document                IsDefaultVersion
        VersionId
----------                              --------                ----------------
        ---------
4/15/2015 10:54:54 AM                                           True
        v2
```

- API の詳細については、「 コマンドレットリファレンスCreatePolicyVersion」の「」を参照してください。 AWS Tools for PowerShell

New-IAMRole

次の例は、New-IAMRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyNewRole** という名前の新しいロールを作成し、**NewRoleTrustPolicy.json** ファイルにあるポリシーをそのロールにアタッチします。JSON ポリシーファイルを正常に処理するには、**-Raw** switch パラメータを使用する必要が

あることに注意してください。出力に表示されるポリシードキュメントは、URL でエンコードされています。この例では、**UrlDecode** .NET メソッドを使用してデコードされています。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

出力:

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D
%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A
%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

- API の詳細については、「 コマンドレットリファレンスCreateRole」の「」を参照してください。 AWS Tools for PowerShell

New-IAMSAMLProvider

次の例は、New-IAMSAMLProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM に新しい SAML プロバイダーエンティティを作成します。これは **MySAMLProvider** という名前で、SAML サービスプロバイダーのウェブサイトから個別にダウンロードされた **SAMLMetaData.xml** ファイルにある SAML メタデータドキュメントによって記述されます。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

出力:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「CreateSAMLProvider」を参照してください。

New-IAMServiceLinkedRole

次の例は、New-IAMServiceLinkedRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、自動スケーリングサービスのサービスにリンクされたロールを作成します。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- API の詳細については、「 コマンドレットリファレンスCreateServiceLinkedRole」の「」を参照してください。 AWS Tools for PowerShell

New-IAMUser

次の例は、New-IAMUser を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーを作成します。Bob が AWS コンソールにサインインする必要がある場合は、別途 コマンド**New-IAMLoginProfile**を実行して、パスワードでサインインプロファイルを作成する必要があります。Bob が CLI コマンドを実行 AWS PowerShell またはクロスプラットフォーム CLI コマンドを実行したり、 AWS API コールを実行したりする必要がある場合は、 **New-IAMAccessKey** コマンドを個別に実行してアクセスキーを作成する必要があります。

```
New-IAMUser -UserName Bob
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateUser」の「」を参照してください。 AWS Tools for PowerShell

New-IAMVirtualMFADevice

次の例は、New-IAMVirtualMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しい仮想 MFA デバイスを作成します。2 行目と 3 行目は、仮想 MFA ソフトウェアプログラムが (QR コードの代わりに) アカウントを作成するのに必要な **Base32StringSeed** 値を抽出します。この値でプログラムを設定したら、プログラムから 2 つの連続した認証コードを取得します。最後に、最後のコマンドを使用して仮想 MFA デバイスを IAM ユーザー **Bob** にリンクし、アカウントを 2 つの認証コードと同期します。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
```

```
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

出力:

```
-- 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
```

例 2: この例では、新しい仮想 MFA デバイスを作成します。2 行目と 3 行目は、**QRCodePNG** 値を抽出してファイルに書き込みます。このイメージは、仮想 MFA ソフトウェアプログラムでスキャンしてアカウントを作成できます (Base32 StringSeed 値を手動で入力する代わりに)。仮想 MFA プログラムでアカウントを作成したら、2 つの連続した認証コードを取得して最後のコマンドに入力し、仮想 MFA デバイスを IAM ユーザー **Bob** にリンクして、アカウントを同期します。

```
$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
```

出力:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- API の詳細については、「 コマンドレットリファレンスCreateVirtualMfaDevice」の「」を参照してください。 AWS Tools for PowerShell

Publish-IAMServerCertificate

次の例は、Publish-IAMServerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しいサーバー証明書を IAM アカウントにアップロードします。証明書本文、プライベートキー、および (オプションで) 証明書チェーンを含むファイルは、すべて PEM エンコードされる必要があります。パラメータにはファイル名ではなくファイルの実際の内容が必要であることに注意してください。ファイルの内容を正常に処理するには、**-Raw** スイッチパラメータを使用する必要があります。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUploadServerCertificate」の「」を参照してください。 AWS Tools for PowerShell

Publish-IAMSigningCertificate

次の例は、Publish-IAMSigningCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しい X.509 署名証明書をアップロードし、**Bob** という名前の IAM ユーザーに関連付けます。証明書の本文を含むファイルは PEM でエンコードされています。**CertificateBody** パラメータには、ファイル名ではなく証明書ファイルの実際の内容が必要です。ファイルを正常に処理するには、**-Raw** スイッチパラメータを使用する必要があります。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

出力:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- API の詳細については、「 コマンドレットリファレンスUploadSigningCertificate」の「」を参照してください。 AWS Tools for PowerShell

Register-IAMGroupPolicy

次の例は、Register-IAMGroupPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**TesterPolicy** という名前のカスタマー管理ポリシーを IAM グループ **Testers** にアタッチします。そのグループのユーザーは、そのポリシーのデフォルトバージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

例 2: この例では、 という名前の AWS マネージドポリシーを IAM グループ **AdministratorAccess**にアタッチします**Admins**。そのグループのユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- API の詳細については、「 コマンドレットリファレンスAttachGroupPolicy」の「」を参照してください。 AWS Tools for PowerShell

Register-IAMRolePolicy

次の例は、Register-IAMRolePolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 という名前の AWS マネージドポリシーを IAM ロール **SecurityAudit**にアタッチします**CoSecurityAuditors**。そのロールを引き受けるユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- API の詳細については、「 コマンドレットリファレンスAttachRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

Register-IAMUserPolicy

次の例は、Register-IAMUserPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 という名前の AWS マネージドポリシーを IAM ユーザー **AmazonCognitoPowerUser**にアタッチします**Bob**。ユーザーは、そのポリシーの最新バージョンで定義されているアクセス権限の影響をすぐに受けます。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- API の詳細については、「 コマンドレットリファレンスAttachUserPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMAccessKey

次の例は、Remove-IAMAccessKey を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 という名前のユーザー**AKIAIOSFODNN7EXAMPLE**からキー ID を持つ AWS アクセスキーペアを削除します**Bob**。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteAccessKey」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMAccountAlias

次の例は、Remove-IAMAccountAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントエイリアスを から削除します AWS アカウント。https://mycompanyaws.signin.aws.amazon.com/console で、エイリアスを持つユーザーサインインページは機能しなくなりました。代わりに、<accountidnumber>.https:// の AWS アカウント ID 番号で元の URL を使用する必要があります。signin.aws.amazon.com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API の詳細については、「 コマンドレットリファレンスDeleteAccountAlias」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMAccountPasswordPolicy

次の例は、Remove-IAMAccountPasswordPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 のパスワードポリシーを削除 AWS アカウント し、すべての値を元のデフォルトにリセットします。パスワードポリシーが現在存在しない場合、次のエラーメッセージが表示されます。名前 のアカウントポリシー PasswordPolicy が見つかりません。

```
Remove-IAMAccountPasswordPolicy
```

- API の詳細については、「 コマンドレットリファレンスDeleteAccountPasswordPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMClientIDFromOpenIDConnectProvider

次の例は、Remove-IAMClientIDFromOpenIDConnectProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** である IAM OIDC プロバイダーに関連付けられているクライアント ID のリストから、クライアント ID **My-TestApp-3** を削除します。

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- API の詳細については、「 コマンドレットリファレンスRemoveClientIdFromOpenIdConnectProvider」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMGroup

次の例は、Remove-IAMGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyTestGroup** という名前の IAM グループを削除します。最初のコマンドはグループのメンバーであるすべての IAM ユーザーを削除し、2 番目のコマンドは IAM グループを削除します。どちらのコマンドも、確認を求めるプロンプトが表示されなくても機能します。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMGroupPolicy

次の例は、Remove-IAMGroupPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**TesterPolicy** という名前のインラインポリシーを IAM グループ **Testers** から削除します。そのグループのユーザーは、そのポリシーで定義されているアクセス権限をすぐに失います。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- API の詳細については、「 コマンドレットリファレンスDeleteGroupPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMInstanceProfile

次の例は、Remove-IAMInstanceProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyAppInstanceProfile** という名前の EC2 インスタンスプロファイルを削除します。最初のコマンドはインスタンスプロファイルからすべてのロールをデタッチし、2 番目のコマンドはインスタンスプロファイルを削除します。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- API の詳細については、「 コマンドレットリファレンスDeleteInstanceProfile」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMLoginProfile

次の例は、Remove-IAMLoginProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーからログインプロファイルを削除します。これにより、ユーザーは AWS コンソールにサインインできなくなります。ユーザーが AWS CLI を実行したり PowerShell、ユーザーアカウントにアタッチされている可能性のある AWS アクセスキーを使用した API コールを実行したりすることを妨げたりすることはありません。

```
Remove-IAMLoginProfile -UserName Bob
```

- API の詳細については、「 コマンドレットリファレンスDeleteLoginProfile」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMOpenIDConnectProvider

次の例は、Remove-IAMOpenIDConnectProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、プロバイダー **example.oidcprovider.com** に接続する IAM OIDC プロバイダーを削除します。ロールの信頼ポリシーの **Principal** 要素で、このプロバイダーを参照するロールをすべて更新または削除してください。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API の詳細については、「 コマンドレットリファレンスDeleteOpenIdConnectProvider」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMPolicy

次の例は、Remove-IAMPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/MySamplePolicy** であるポリシーを削除します。ポリシーを削除する前に、**Remove-IAMPolicyVersion** を実行して、デフォルト以外のすべてのバージョンを削除する必要があります。また、すべての IAM ユーザー、グループ、またはロールからポリシーをデタッチする必要があります。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

例 2: この例では、最初にデフォルト以外のポリシーバージョンをすべて削除し、アタッチされているすべての IAM エンティティからデタッチして、最後にポリシー自体を削除することでポリシーを削除します。1 行目では、ポリシーオブジェクトを取得します。2 行目では、デフォルトバージョンとしてフラグが立てられていないすべてのポリシーバージョンをコレクションに取得し、コレクション内の各ポリシーを削除します。3 行目では、ポリシーがアタッチされているすべての IAM ユーザー、グループ、およびロールを取得します。4 行目から 6 行目では、アタッチされている各エンティティからポリシーをデタッチします。最後の行では、このコマンドを使用して管理ポリシーと残りのデフォルトバージョンを削除します。この例には、確認を求めるプロンプトを非表示にするための **-Force** switch パラメータが、このパラメータを必要とする行に含まれています。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスDeletePolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMPolicyVersion

次の例は、Remove-IAMPolicyVersion を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**v2** と識別されたバージョンを ARN が **arn:aws:iam::123456789012:policy/MySamplePolicy** であるポリシーから削除します。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

例 2: この例では、最初にデフォルト以外のポリシーバージョンをすべて削除し、次にポリシー自体を削除することでポリシーを削除します。1 行目では、ポリシーオブジェクトを取得します。2

行目では、デフォルトとしてフラグが立てられていないすべてのポリシーバージョンをコレクションに取得し、このコマンドを使用してコレクション内の各ポリシーを削除します。最後の行では、ポリシー自体と残りのデフォルトバージョンを削除します。管理ポリシーを正常に削除するには、**Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy**、**Unregister-IAMRolePolicy** コマンドを使用して、ユーザー、グループ、またはロールからポリシーをデタッチする必要があることに注意してください。**Remove-IAMPolicy** コマンドレットの例を参照してください。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスDeletePolicyVersion」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMRole

次の例は、Remove-IAMRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の IAM アカウントから **MyNewRole** という名前のロールを削除します。ロールを削除する前に、まず **Unregister-IAMRolePolicy** コマンドを使用して、管理ポリシーをデタッチする必要があります。インラインポリシーは、ロールと共に削除されます。

```
Remove-IAMRole -RoleName MyNewRole
```

例 2: この例では、**MyNewRole** という名前のロールから管理ポリシーをすべてデタッチして、ロールを削除します。最初の行では、ロールにアタッチされているすべての管理ポリシーをコレクションとして取得し、コレクション内の各ポリシーをロールからデタッチします。2 行目では、ロール自体を削除します。インラインポリシーは、ロールと共に削除されます。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- API の詳細については、「 コマンドレットリファレンスDeleteRole」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMRoleFromInstanceProfile

次の例は、Remove-IAMRoleFromInstanceProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyNewRole** という名前の EC2 インスタンスプロファイルから **MyNewRole** という名前のロールを削除します。IAM コンソールで作成されたインスタンスプロファイルは、この例のように、常にロールと同じ名前になります。API または CLI で作成する場合、名前は異なる場合があります。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- API の詳細については、「 コマンドレットリファレンスRemoveRoleFromInstanceProfile」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMRolePermissionsBoundary

次の例は、Remove-IAMRolePermissionsBoundary を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、IAM ロールにアタッチされたアクセス許可の境界を削除する方法を示しています。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API の詳細については、「 コマンドレットリファレンスDeleteRolePermissionsBoundary」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMRolePolicy

次の例は、Remove-IAMRolePolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ロール **S3BackupRole** に埋め込まれているインラインポリシー **S3AccessPolicy** を削除します。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API の詳細については、「 コマンドレットリファレンスDeleteRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMRoleTag

次の例は、Remove-IAMRoleTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、タグキーを「abac」として「」という名前MyRoleNameのロールからタグを削除します。複数のタグを削除するには、カンマで区切ったタグキーリストを指定します。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API の詳細については、「 コマンドレットリファレンスUntagRole」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMSAMLProvider

次の例は、Remove-IAMSAMLProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider** である IAM SAML 2.0 プロバイダーを削除します。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「DeleteSAMLProvider」を参照してください。

Remove-IAMServerCertificate

次の例は、Remove-IAMServerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyServerCert** という名前のサーバー証明書を削除します。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API の詳細については、「 コマンドレットリファレンスDeleteServerCertificate」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMServiceLinkedRole

次の例は、Remove-IAMServiceLinkedRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、サービスにリンクされたロールを削除しました。サービスがこのロールをまだ使用している場合、このコマンドは失敗することに注意してください。

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API の詳細については、「 コマンドレットリファレンスDeleteServiceLinkedRole」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMSigningCertificate

次の例は、Remove-IAMSigningCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーから ID **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** の付いた署名証明書を削除します。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API の詳細については、「 コマンドレットリファレンスDeleteSigningCertificate」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMUser

次の例は、Remove-IAMUser を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーを削除します。

```
Remove-IAMUser -UserName Bob
```

例 2: この例では、**Theresa** という名前の IAM ユーザーと、最初に削除する必要がある要素をすべて削除します。

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
 -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
```

```
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- API の詳細については、「 コマンドレットリファレンスDeleteUser」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMUserFromGroup

次の例は、Remove-IAMUserFromGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **Bob** をグループ **Testers** から削除します。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

例 2: この例では、IAM ユーザー **Theresa** がメンバーとなっているグループをすべて検索し、それらのグループから **Theresa** を削除します。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

例 3: この例は、IAM ユーザー **Bob** を **Testers** グループから削除する別の方法を示しています。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- API の詳細については、「 コマンドレットリファレンスRemoveUserFromGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMUserPermissionsBoundary

次の例は、Remove-IAMUserPermissionsBoundary を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、IAM ユーザーにアタッチされたアクセス許可の境界を削除する方法を示しています。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- API の詳細については、「 コマンドレットリファレンスDeleteUserPermissionsBoundary」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMUserPolicy

次の例は、Remove-IAMUserPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーに埋め込まれている **AccessToEC2Policy** という名前のインラインポリシーを削除します。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

例 2: この例では、**Theresa** という名前の IAM ユーザーに埋め込まれているすべてのインラインポリシーを検索し、削除します。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- API の詳細については、「 コマンドレットリファレンスDeleteUserPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMUserTag

次の例は、Remove-IAMUserTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、タグキーが「abac」と「xyzw」の「joe」という名前のユーザーからタグを削除します。複数のタグを削除するには、カンマで区切ったタグキーリストを指定します。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- API の詳細については、「 コマンドレットリファレンスUntagUser」の「」を参照してください。 AWS Tools for PowerShell

Remove-IAMVirtualMFADevice

次の例は、Remove-IAMVirtualMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:mfa/bob** である IAM 仮想 MFA デバイスを削除します。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

例 2: この例では、IAM ユーザー Theresa に MFA デバイスが割り当てられているかどうかを確認します。割り当てられているデバイスが見つかった場合、そのデバイスはその IAM ユーザーに対して無効になります。デバイスが仮想の場合は、そのデバイスも削除されます。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- API の詳細については、「 コマンドレットリファレンスDeleteVirtualMfaDevice」の「」を参照してください。 AWS Tools for PowerShell

Request-IAMCredentialReport

次の例は、Request-IAMCredentialReport を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、4 時間ごとに実行できる新しいレポートの生成を要求しています。最新のレポートがまだ新しい場合、[状態] フィールドには **COMPLETE** と表示されます。**Get-IAMCredentialReport** を使用して、完成したレポートを表示します。

```
Request-IAMCredentialReport
```

出力:

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- API の詳細については、「 コマンドレットリファレンスGenerateCredentialReport」の「」を参照してください。 AWS Tools for PowerShell

`Request-IAMServiceLastAccessedDetail`

次の例は、`Request-IAMServiceLastAccessedDetail` を使用する方法を説明しています。

のツール PowerShell

例 1: この例は GenerateServiceLastAccessedDetails API の同等のコマンドレットです。これにより、Get-IAM ServiceLastAccessedDetail および Get-IAM で使用できるジョブ ID が提供されます。ServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API の詳細については、「 コマンドレットリファレンスGenerateServiceLastAccessedDetails」の「」を参照してください。 AWS Tools for PowerShell

`Set-IAMDefaultPolicyVersion`

次の例は、`Set-IAMDefaultPolicyVersion` を使用する方法を説明しています。

のツール PowerShell

例 1: ARN が **arn:aws:iam::123456789012:policy/MyPolicy** であるポリシーの **v2** バージョンをデフォルトのアクティブなバージョンとして設定します。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- API の詳細については、「 コマンドレットリファレンスSetDefaultPolicyVersion」の「」を参照してください。 AWS Tools for PowerShell

Set-IAMRolePermissionsBoundary

次の例は、Set-IAMRolePermissionsBoundary を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、IAM ロールのアクセス許可の境界を設定する方法を示しています。 AWS 管理ポリシーまたはカスタムポリシーをアクセス許可の境界として設定できます。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API の詳細については、「 コマンドレットリファレンスPutRolePermissionsBoundary」の「」を参照してください。 AWS Tools for PowerShell

Set-IAMUserPermissionsBoundary

次の例は、Set-IAMUserPermissionsBoundary を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、ユーザーのアクセス許可の境界を設定する方法を示しています。 AWS 管理ポリシーまたはカスタムポリシーをアクセス許可の境界として設定できます。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API の詳細については、「 コマンドレットリファレンスPutUserPermissionsBoundary」の「」を参照してください。 AWS Tools for PowerShell

Sync-IAMMFADevice

次の例は、Sync-IAMMFADevice を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **Bob** に関連付けられていて、ARN が **arn:aws:iam::123456789012:mfa/bob** である MFA デバイスを、2 つの認証コードを提供した認証プログラムと同期します。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

例 2: この例では、IAM ユーザー **Theresa** に関連付けられている IAM MFA デバイスを、シリアル番号が **ABCD12345678** であり、2 つの認証コードを提供した物理デバイスと同期します。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- API の詳細については、「 コマンドレットリファレンスResyncMfaDevice」の「」を参照してください。 AWS Tools for PowerShell

Unregister-IAMGroupPolicy

次の例は、Unregister-IAMGroupPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/TesterAccessPolicy** である管理グループポリシーを **Testers** という名前のグループからデタッチします。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

例 2: この例では、**Testers** という名前のグループにアタッチされているすべての管理ポリシーを検索し、グループからデタッチします。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- API の詳細については、「 コマンドレットリファレンスDetachGroupPolicy」の「」を参照してください。 AWS Tools for PowerShell

Unregister-IAMRolePolicy

次の例は、Unregister-IAMRolePolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** である管理グループポリシーを **FedTesterRole** という名前のロールからデタッチします。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

例 2: この例では、**FedTesterRole** という名前のロールにアタッチされているすべての管理ポリシーを検索し、ロールからデタッチします。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- API の詳細については、「 コマンドレットリファレンスDetachRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

Unregister-IAMUserPolicy

次の例は、Unregister-IAMUserPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:policy/TesterPolicy** である管理ポリシーを **Bob** という名前の IAM ユーザーからデタッチします。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

例 2: この例では、**Theresa** という名前の IAM ユーザーにアタッチされているすべての管理ポリシーを検索し、それらのポリシーをユーザーからデタッチします。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- API の詳細については、「 コマンドレットリファレンスDetachUserPolicy」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMAccessKey

次の例は、Update-IAMAccessKey を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーのアクセスキー **AKIAIOSFODNN7EXAMPLE** のステータスを **Inactive** に変更します。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- API の詳細については、「 コマンドレットリファレンスUpdateAccessKey」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMAccountPasswordPolicy

次の例は、Update-IAMAccountPasswordPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した設定でアカウントのパスワードポリシーを更新します。コマンドに含まれていないパラメータは変更されないままにはならないことに注意してください。代わりに、デフォルト値にリセットされます。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- API の詳細については、「 コマンドレットリファレンスUpdateAccountPasswordPolicy」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMAssumeRolePolicy

次の例は、Update-IAMAssumeRolePolicy を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**ClientRole** という名前の IAM ロールを新しい信頼ポリシーで更新します。その内容は、ファイル **ClientRolePolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** スイッチパラメータを使用する必要があることに注意してください。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- API の詳細については、「 コマンドレットリファレンスUpdateAssumeRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMGroup

次の例は、Update-IAMGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM グループ **Testers** の名前を **AppTesters** に変更します。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

例 2: この例では、IAM グループ **AppTesters** のパスを **/Org1/Org2/** に変更します。これにより、グループの ARN が **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters** に変更されます。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- API の詳細については、「 コマンドレットリファレンスUpdateGroup」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMLoginProfile

次の例は、Update-IAMLoginProfile を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **Bob** に新しい一時パスワードを設定し、ユーザーが次回サインインしたときにパスワードを変更するようユーザーに要求します。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- API の詳細については、「 コマンドレットリファレンスUpdateLoginProfile」の「」を参照してください。 AWS Tools for PowerShell

`Update-IAMOpenIDConnectProviderThumbprint`

次の例は、`Update-IAMOpenIDConnectProviderThumbprint` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** である OIDC プロバイダーの証明書サムプリントリストを更新して、新しいサムプリントを使用します。OIDC プロバイダーは、プロバイダーに関連付けられている証明書が変更されると、新しい値を共有します。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- API の詳細については、「 コマンドレットリファレンスUpdateOpenIdConnectProviderThumbprint」の「」を参照してください。 AWS Tools for PowerShell

`Update-IAMRole`

次の例は、`Update-IAMRole` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ロールの説明と、ロールのセッションをリクエストできる最大セッション期間の値 (秒単位) を更新します。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- API の詳細については、「 コマンドレットリファレンスUpdateRole」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMRoleDescription

次の例は、Update-IAMRoleDescription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントの IAM ロールの説明を更新します。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API の詳細については、「 コマンドレットリファレンスUpdateRoleDescription」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMSAMLProvider

次の例は、Update-IAMSAMLProvider を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ARN が **arn:aws:iam::123456789012:saml-provider/SAMLADFS** である IAM の SAML プロバイダーを、ファイル **SAMLMetaData.xml** の新しい SAML メタデータドキュメントで更新します。JSON ファイルの内容を正常に処理するには、**-Raw** スイッチパラメータを使用する必要があることに注意してください。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- API の詳細については、「 コマンドレットリファレンスUpdateSamlProvider」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMServerCertificate

次の例は、Update-IAMServerCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**MyServerCertificate** という証明書の名前を **MyRenamedServerCertificate** に変更します。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

例 2: この例では、**MyServerCertificate** という証明書を /Org1/Org2/ というパスに移動します。これにより、リソースの ARN が **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate** に変更されます。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- API の詳細については、「 コマンドレットリファレンスUpdateServerCertificate」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMSigningCertificate

次の例は、Update-IAMSigningCertificate を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**Bob** という名前の IAM ユーザーに関連付けられ、証明書 ID が **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** である証明書を更新し、無効としてマークされるようにします。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- API の詳細については、「 コマンドレットリファレンスUpdateSigningCertificate」の「」を参照してください。 AWS Tools for PowerShell

Update-IAMUser

次の例は、Update-IAMUser を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、IAM ユーザー **Bob** の名前を **Robert** に変更します。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

例 2: この例では、IAM ユーザー **Bob** のパスを **/Org1/Org2/** に変更します。これにより、ユーザーの ARN は実質的に **arn:aws:iam::123456789012:user/Org1/Org2/bob** に変更されます。

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- API の詳細については、「 コマンドレットリファレンスUpdateUser」の「」を参照してください。 AWS Tools for PowerShell

`Write-IAMGroupPolicy`

次の例は、`Write-IAMGroupPolicy` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**AppTesterPolicy** という名前のインラインポリシーを作成し、IAM グループ **AppTesters** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容がファイル **apptesterpolicy.json** に送られます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要があることに注意してください。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- API の詳細については、「 コマンドレットリファレンスPutGroupPolicy」の「」を参照してください。 AWS Tools for PowerShell

`Write-IAMRolePolicy`

次の例は、`Write-IAMRolePolicy` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**FedTesterRolePolicy** という名前のインラインポリシーを作成し、IAM ロール **FedTesterRole** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容は、ファイル **FedTesterPolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要があることに注意してください。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- API の詳細については、「 コマンドレットリファレンスPutRolePolicy」の「」を参照してください。 AWS Tools for PowerShell

`Write-IAMUserPolicy`

次の例は、`Write-IAMUserPolicy` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、**EC2AccessPolicy** という名前のインラインポリシーを作成し、IAM ユーザー **Bob** に埋め込みます。同じ名前のインラインポリシーが既に存在する場合、上書きされます。JSON ポリシーの内容は、ファイル **EC2AccessPolicy.json** から取得されます。JSON ファイルの内容を正常に処理するには、**-Raw** パラメータを使用する必要があることに注意してください。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- API の詳細については、「 コマンドレットリファレンスPutUserPolicy」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Kinesis の例 PowerShell

次のコード例は、Kinesis AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-KINRecord`

次の例は、`Get-KINRecord` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、一連の 1 つ以上のレコードからデータを返して抽出する方法を示します。Get-KINRecord に供給されたイテレーターは、この例のどのレコードが変数 $records にキャプチャされるかを返すレコードの開始位置を決定します。その後、$records コレクションにインデックスを付けることで、個々のレコードにアクセスできます。レコード内のデータが UTF-8 でエンコードされたテキストであると仮定すると、最後のコマンドは、オブジェクト MemoryStream 内の からデータを抽出し、それをテキストとしてコンソールに返す方法を示しています。

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

出力:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

出力:

```
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())
```

出力:

```
test data from string
```

- API の詳細については、「 コマンドレットリファレンスGetRecords」の「」を参照してください。 AWS Tools for PowerShell

Get-KINShardIterator

次の例は、Get-KINShardIterator を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたシャードと開始位置のシャードイテレーターを返します。シャード識別子とシーケンス番号の詳細は、返されたストリームオブジェクトのシャードコレクションを参照することで、Get-KINStream コマンドレットの出力から取得できます。返されたイテレーターを Get-KINRecord コマンドレットとともに使用して、シャード内のデータレコードをプルできます。

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

出力:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- API の詳細については、「 コマンドレットリファレンスGetShardIterator」の「」を参照してください。 AWS Tools for PowerShell

Get-KINStream

次の例は、Get-KINStream を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたストリームの詳細を返します。

```
Get-KINStream -StreamName "mystream"
```

出力:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- API の詳細については、「 コマンドレットリファレンスDescribeStream」の「」を参照してください。 AWS Tools for PowerShell

New-KINStream

次の例は、New-KINStream を使用する方法を説明しています。

のツール PowerShell

例 1: 新しいストリームを作成します。デフォルトでは、このコマンドレットは出力を返さないため、後で使用できるように - StreamName パラメータに指定された値を返すように - PassThru スイッチが追加されます。

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- API の詳細については、「 コマンドレットリファレンスCreateStream」の「」を参照してください。 AWS Tools for PowerShell

Remove-KINStream

次の例は、Remove-KINStream を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたストリームを削除します。コマンドを実行する前に確認を求められます。確認プロンプトを抑制するには、-Force スイッチを使用します。

```
Remove-KINStream -StreamName "mystream"
```

- API の詳細については、「 コマンドレットリファレンスDeleteStream」の「」を参照してください。 AWS Tools for PowerShell

`Write-KINRecord`

次の例は、`Write-KINRecord` を使用する方法を説明しています。

のツール PowerShell

例 1: -Text パラメータに指定された文字列を含むレコードを書き込みます。

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

例 2: 指定されたファイルに含まれるデータを含むレコードを書き込みます。ファイルはバイトのシーケンスとして扱われるため、テキストが含まれている場合は、このコマンドレットで使用する前に、必要なエンコードで記述する必要があります。

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- API の詳細については、「 コマンドレットリファレンスPutRecord」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Lambda の例 PowerShell

次のコード例は、Lambda AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-LMResourceTag`

次の例は、Add-LMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: 3 つのタグ (Washington、Oregon、California) およびそれぞれに関連付けされた値を、ARN で識別される指定の関数に追加します。

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- API の詳細については、「 コマンドレットリファレンスTagResource」の「」を参照してください。 AWS Tools for PowerShell

`Get-LMAccountSetting`

次の例は、Get-LMAccountSetting を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、アカウント制限およびアカウント使用量を比較するために表示されます。

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

出力:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
```

```
        80530636800          15078795
```

- API の詳細については、「 コマンドレットリファレンスGetAccountSettings」の「」を参照してください。 AWS Tools for PowerShell

Get-LMAlias

次の例は、Get-LMAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定の Lambda 関数エイリアスのルーティング設定の重点を取得します。

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

出力:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- API の詳細については、「 コマンドレットリファレンスGetAlias」の「」を参照してください。 AWS Tools for PowerShell

Get-LMFunctionConcurrency

次の例は、Get-LMFunctionConcurrency を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の予約済み同時実行数が取得されます

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

出力:

```
ReservedConcurrentExecutions
----------------------------
100
```

- API の詳細については、「 コマンドレットリファレンスGetFunctionConcurrency」の「」を参照してください。 AWS Tools for PowerShell

Get-LMFunctionConfiguration

次の例は、Get-LMFunctionConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数のバージョン固有設定を返します。

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

出力:

```
CodeSha256                  : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                    : 1426
DeadLetterConfig            : Amazon.Lambda.Model.DeadLetterConfig
Description                 : Verson 3 to test Aliases
Environment                 : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                 : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                              :PowershellAlias
FunctionName                : MylambdaFunction123
Handler                     : lambda_function.launch_instance
KMSKeyArn                   :
LastModified                : 2019-12-25T09:52:59.872+0000
LastUpdateStatus            : Successful
LastUpdateStatusReason      :
LastUpdateStatusReasonCode  :
Layers                      : {}
MasterArn                   :
MemorySize                  : 128
RevisionId                  : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                        : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                     : python3.8
State                       : Active
StateReason                 :
StateReasonCode             :
Timeout                     : 600
TracingConfig               : Amazon.Lambda.Model.TracingConfigResponse
```

```
Version                    : 4
VpcConfig                  : Amazon.Lambda.Model.VpcConfigDetail
```

- API の詳細については、「 コマンドレットリファレンスGetFunctionConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-LMFunctionList

次の例は、Get-LMFunctionList を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、すべての Lambda 関数をソートされたコードサイズで表示されます

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

出力:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
 243
MylambdaFunction123                                             python3.8     600
 659
myfuncpython1                                                   python3.8     303
 675
```

- API の詳細については、「 コマンドレットリファレンスListFunctions」の「」を参照してください。 AWS Tools for PowerShell

Get-LMPolicy

次の例は、Get-LMPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、Lambda 関数の関数ポリシーが表示されます

```
Get-LMPolicy -FunctionName test -Select Policy
```

出力:

```
{"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"}]}
```

- API の詳細については、「 コマンドレットリファレンスGetPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-LMProvisionedConcurrencyConfig

次の例は、Get-LMProvisionedConcurrencyConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の指定されたエイリアスにプロビジョニングされた同時実行設定を取得します。

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

出力:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- API の詳細については、「 コマンドレットリファレンスGetProvisionedConcurrencyConfig」の「」を参照してください。 AWS Tools for PowerShell

Get-LMProvisionedConcurrencyConfigList

次の例は、Get-LMProvisionedConcurrencyConfigList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数にプロビジョニングされた同時実行設定のリストを取得します。

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- API の詳細については、「 コマンドレットリファレンスListProvisionedConcurrencyConfigs」の「」を参照してください。 AWS Tools for PowerShell

Get-LMResourceTag

次の例は、Get-LMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: 指定した関数に現在設定されているタグとその値を取得します。

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

出力:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- API の詳細については、「 コマンドレットリファレンスListTags」の「」を参照してください。 AWS Tools for PowerShell

Get-LMVersionsByFunction

次の例は、Get-LMVersionsByFunction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の各バージョンのバージョン固有設定に関するリストを返します。

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListVersionsByFunction」の「」を参照してください。 AWS Tools for PowerShell

New-LMAlias

次の例は、New-LMAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたバージョンおよびルーティング設定の新しい Lambda エイリアスを作成し、受信する呼び出しリクエストの割合を指定します。

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- API の詳細については、「 コマンドレットリファレンスCreateAlias」の「」を参照してください。 AWS Tools for PowerShell

Publish-LMFunction

次の例は、Publish-LMFunction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 AWS Lambda MyFunction で という名前の新しい C# (dotnetcore1.0 ランタイム) 関数を作成し、ローカルファイルシステムの zip ファイルから関数のコンパイルされたバイナリを提供します (相対パスまたは絶対パスを使用できます) 。C# Lambda 関数は、 AssemblyName::Namespace.:: という指定を使用して関数のハンドラーを指定しますClassNameMethodName。ハンドラー仕様のアセンブリ名 (.dll サフィックスなし)、名前空間、クラス名、メソッド名の部分を適切に置き換える必要があります。新しい関数には、指定された値で「envvar1」および「envvar2」の環境変数が設定されます。

```
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" }
```

出力:

```
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        :
```

例 2: この例は前の例と似ていますが、関数バイナリが最初に Amazon S3 バケット (目的の Lambda 関数と同じリージョンにある必要がある) にアップロードされ、結果の S3 オブジェクトが関数の作成時に参照される点が異なります。

```
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" }
```

- API の詳細については、「 コマンドレットリファレンスCreateFunction」の「」を参照してください。 AWS Tools for PowerShell

Publish-LMVersion

次の例は、Publish-LMVersion を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数コードの既存のスナップショットのバージョンを作成します

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- API の詳細については、「 コマンドレットリファレンスPublishVersion」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMAlias

次の例は、Remove-LMAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、コマンドに記述された Lambda 関数のエイリアスを削除します。

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- API の詳細については、「 コマンドレットリファレンスDeleteAlias」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMFunction

次の例は、Remove-LMFunction を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の特定のバージョンを削除します。

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- API の詳細については、「 コマンドレットリファレンスDeleteFunction」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMFunctionConcurrency

次の例は、Remove-LMFunctionConcurrency を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の関数同時実行数を削除します。

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- API の詳細については、「 コマンドレットリファレンスDeleteFunctionConcurrency」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMPermission

次の例は、Remove-LMPermission を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Lambda 関数の指定された StatementId の関数ポリシーを削除します。

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- API の詳細については、「 コマンドレットリファレンスRemovePermission」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMProvisionedConcurrencyConfig

次の例は、Remove-LMProvisionedConcurrencyConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、特定のエイリアスのプロビジョニングされた同時実行設定を削除します。

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- API の詳細については、「 コマンドレットリファレンスDeleteProvisionedConcurrencyConfig」の「」を参照してください。 AWS Tools for PowerShell

Remove-LMResourceTag

次の例は、Remove-LMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: 提供されたタグを関数から削除します。-Force スイッチが指定されていない限り、cmdlet は続行する前に確認を求めます。タグを削除するため、サービスが 1 回呼び出されます。

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

例 2: 提供されたタグを関数から削除します。-Force スイッチが指定されていない限り、cmdlet は続行する前に確認を求めます。提供されたタグにつき、サービスに 1 回呼び出しが行われます。

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- API の詳細については、「 コマンドレットリファレンスUntagResource」の「」を参照してください。 AWS Tools for PowerShell

Update-LMAlias

次の例は、Update-LMAlias を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、既存の Lambda 関数エイリアスの設定を更新します。トラフィックの 60% (0.6) をバージョン 1 に移行するように RoutingConfiguration 値を更新します。

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- API の詳細については、「 コマンドレットリファレンスUpdateAlias」の「」を参照してください。 AWS Tools for PowerShell

Update-LMFunctionCode

次の例は、Update-LMFunctionCode を使用する方法を説明しています。

のツール PowerShell

例 1: 'MyFunction' という名前の関数を、指定された zip ファイルに含まれる新しいコンテンツで更新します。C# .NET Core Lambda 関数には、zip ファイルはコンパイルされたアセンブリが含まれている必要があります。

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

例 2: この例は前の例と似ていますが、更新されたコードを含む Amazon S3 オブジェクトを使用して関数を更新します。

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName mybucket -Key
 UpdatedCode.zip
```

- API の詳細については、「 コマンドレットリファレンスUpdateFunctionCode」の「」を参照してください。 AWS Tools for PowerShell

Update-LMFunctionConfiguration

次の例は、Update-LMFunctionConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、既存の Lambda 関数の設定を更新します

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateFunctionConfiguration」の「」を参照してください。 AWS Tools for PowerShell

`Write-LMFunctionConcurrency`

次の例は、`Write-LMFunctionConcurrency` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、関数全体の同時実行設定を適用します。

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- API の詳細については、「 コマンドレットリファレンスPutFunctionConcurrency」の「」を参照してください。 AWS Tools for PowerShell

`Write-LMProvisionedConcurrencyConfig`

次の例は、`Write-LMProvisionedConcurrencyConfig` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、プロビジョニングされた同時実行設定を関数のエイリアスに追加します。

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- API の詳細については、「 コマンドレットリファレンスPutProvisionedConcurrencyConfig」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon ML の例 PowerShell

次のコード例は、Amazon ML AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Get-MLBatchPrediction`

次の例は、`Get-MLBatchPrediction` を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID のバッチ予測の詳細メタデータを返します。

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- API の詳細については、「 コマンドレットリファレンスGetBatchPrediction」の「」を参照してください。 AWS Tools for PowerShell

`Get-MLBatchPredictionList`

次の例は、`Get-MLBatchPredictionList` を使用する方法を説明しています。

のツール PowerShell

例 1: リクエストで指定された検索条件に一致するすべてのデータレコード BatchPredictions とそれに関連するデータレコードのリストを返します。

```
Get-MLBatchPredictionList
```

例 2: BatchPredictions ステータスが COMPLETED のすべての のリストを返します。

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- API の詳細については、「 コマンドレットリファレンスDescribeBatchPredictions」の「」を参照してください。 AWS Tools for PowerShell

Get-MLDataSource

次の例は、Get-MLDataSource を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID DataSource を持つ のメタデータ、ステータス、およびデータファイル情報を返します。

```
Get-MLDataSource -DataSourceId ID
```

- API の詳細については、「 コマンドレットリファレンスGetDataSource」の「」を参照してください。 AWS Tools for PowerShell

Get-MLDataSourceList

次の例は、Get-MLDataSourceList を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのデータレコード DataSources とそれに関連するデータレコードのリストを返します。

```
Get-MLDataSourceList
```

例 2: DataSources ステータスが COMPLETED のすべての のリストを返します。

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- API の詳細については、「 コマンドレットリファレンスDescribeDataSources」の「」を参照してください。 AWS Tools for PowerShell

Get-MLEvaluation

次の例は、Get-MLEvaluation を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID を持つ評価のメタデータとステータスを返します。

```
Get-MLEvaluation -EvaluationId ID
```

- API の詳細については、「 コマンドレットリファレンスGetEvaluation」の「」を参照してください。 AWS Tools for PowerShell

Get-MLEvaluationList

次の例は、Get-MLEvaluationList を使用する方法を説明しています。

のツール PowerShell

例 1: すべての評価リソースのリストを返します

```
Get-MLEvaluationList
```

例 2: ステータスが COMPLETED のすべてのエバリューションのリストを返します。

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- API の詳細については、「 コマンドレットリファレンスDescribeEvaluations」の「」を参照してください。 AWS Tools for PowerShell

Get-MLModel

次の例は、Get-MLModel を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID を持つ MLModel の詳細メタデータ、ステータス、スキーマ、およびデータファイル情報を返します。

```
Get-MLModel -ModelId ID
```

- API の詳細については、「 コマンドレットリファレンスGetMLModel」を参照してください。 AWS Tools for PowerShell

Get-MLModelList

次の例は、Get-MLModelList を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのモデルとそれに関連するデータレコードのリストを返します。

```
Get-MLModelList
```

例 2: ステータスが COMPLETED のすべてのモデルのリストを返します。

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンスDescribeMLModels」を参照してください。

Get-MLPrediction

次の例は、Get-MLPrediction を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID を持つモデルのリアルタイム予測エンドポイント URL にレコードを送信します。

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- API の詳細については、「 コマンドレットリファレンス」の「予測」を参照してください。 AWS Tools for PowerShell

New-MLBatchPrediction

次の例は、New-MLBatchPrediction を使用する方法を説明しています。

のツール PowerShell

例 1: ID ID を持つモデルの新しいバッチ予測リクエストを作成し、指定した S3 の場所に出力を配置します。

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- API の詳細については、「 コマンドレットリファレンスCreateBatchPrediction」の「」を参照してください。 AWS Tools for PowerShell

New-MLDataSourceFromS3

次の例は、New-MLDataSourceFromS3 を使用する方法を説明しています。

のツール PowerShell

例 1: 名前が NAME、スキーマが SCHEMA の S3 ロケーションのデータを使用してデータソースを作成します。

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- API の詳細については、「 コマンドレットリファレンス」のCreateDataSourceFromS3」を参照してください。 AWS Tools for PowerShell

New-MLEvaluation

次の例は、New-MLEvaluation を使用する方法を説明しています。

のツール PowerShell

例 1: 特定のデータソース ID とモデル ID の評価を作成する

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- API の詳細については、「 コマンドレットリファレンスCreateEvaluation」の「」を参照してください。 AWS Tools for PowerShell

`New-MLModel`

次の例は、`New-MLModel` を使用する方法を説明しています。

のツール PowerShell

例 1: トレーニングデータを使用して新しいモデルを作成する。

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」の「CreateMLModelCreateMLModel」を参照してください。

`New-MLRealtimeEndpoint`

次の例は、`New-MLRealtimeEndpoint` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたモデル ID の新しいリアルタイム予測エンドポイントを作成します。

```
New-MLRealtimeEndpoint -ModelId ID
```

- API の詳細については、「 コマンドレットリファレンスCreateRealtimeEndpoint」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Macie の例 PowerShell

次のコード例は、Macie AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Get-MAC2FindingList`

次の例は、`Get-MAC2FindingList` を使用する方法を説明しています。

のツール PowerShell

例 1: タイプ「CREDIT_CARD_NUMBER」または「US_SOCIAL_SECURITY_NUMBER FindingIds 」の機密データ検出を含む検出結果のリストを返します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスListFindings」の「」を参照してください。 AWS Tools for PowerShell

AWS OpsWorks Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS OpsWorks。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`New-OPSDeployment`

次の例は、New-OPSDeployment を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、 AWS OpsWorks スタックのレイヤー内のすべての Linux ベースのインスタンスに新しいアプリケーションデプロイを作成します。レイヤー ID を指定しても、スタック ID も指定する必要があります。コマンドを使用すると、デプロイは必要に応じてインスタンスを再起動できます。

```
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"}
```

例 2: このコマンドは、ク**phpapp**ックブックの **appsetup** recipe とク**testcookbook**ックブックの **secbaseline** recipe をデプロイします。デプロイターゲットは 1 つのインスタンスですが、スタック ID とレイヤー ID も必要です。Command_Arg パラメータ**allow_reboot**属性はに設定され**true**、必要に応じてデプロイによってインスタンスが再起動されます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateDeployment」の「」を参照してください。 AWS Tools for PowerShell

AWS の料金表 Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS の料金表。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

アクション

`Get-PLSAttributeValue`

次の例は、`Get-PLSAttributeValue` を使用する方法を説明しています。

のツール PowerShell

例 1: us-east-1 リージョンの Amazon EC2 volumeType」の値を返します。

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

出力:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- API の詳細については、「 コマンドレットリファレンスGetAttributeValues」の「」を参照してください。 AWS Tools for PowerShell

Get-PLSProduct

次の例は、Get-PLSProduct を使用する方法を説明しています。

のツール PowerShell

例 1: Amazon EC2 のすべての製品の詳細を返します。

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

出力:

```
{"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"},...
```

例 2: SSD-backed の「汎用」のボリュームタイプでフィルタリングされた us-east-1 リージョンの Amazon EC2 のデータを返します。

```
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
```

出力:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- API の詳細については、「 コマンドレットリファレンスGetProducts」の「」を参照してください。 AWS Tools for PowerShell

Get-PLSService

次の例は、Get-PLSService を使用する方法を説明しています。

のツール PowerShell

例 1: us-east-1 リージョンで使用可能なすべてのサービスコードのメタデータを返します。

```
Get-PLSService -Region us-east-1
```

出力:

```
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
...
```

例 2: us-east-1 リージョンの Amazon EC2 サービスのメタデータを返します。

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

出力:

```
AttributeNames                                                          ServiceCode
--------------                                                          -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- API の詳細については、「 コマンドレットリファレンスDescribeServices」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Resource Groups の例 PowerShell

次のコード例は、Resource Groups AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-RGResourceTag`

次の例は、`Add-RGResourceTag` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、値 'workboxes' のタグキー 'Instances' を指定されたリソースグループ arn に追加します。

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

出力:

```
Arn                                                              Tags
---                                                              ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- API の詳細については、「 コマンドレットリファレンス」の「タグ」を参照してください。AWS Tools for PowerShell

Find-RGResource

次の例は、Find-RGResource を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、タグフィルターを使用して ResourceQuery インスタンスリソースタイプの を作成し、リソースを検索します。

```
$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
```

出力:

```
ResourceArn                                                  ResourceType
-----------                                                  ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- API の詳細については、「 コマンドレットリファレンスSearchResources」の「」を参照してください。 AWS Tools for PowerShell

Get-RGGroup

次の例は、Get-RGGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、グループ名に従って としてリソースグループを取得します。

```
Get-RGGroup -GroupName auto-no
```

出力:

```
Description GroupArn                                                        Name
----------- --------                                                        ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- API の詳細については、「 コマンドレットリファレンスGetGroup」の「」を参照してください。 AWS Tools for PowerShell

Get-RGGroupList

次の例は、Get-RGGroupList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、既に作成されたリソースグループを一覧表示します。

```
Get-RGGroupList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-RGGroupQuery

次の例は、Get-RGGroupQuery を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースグループのリソースクエリを取得します。

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

出力:

```
Query
         Type
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- API の詳細については、「 コマンドレットリファレンスGetGroupQuery」の「」を参照してください。 AWS Tools for PowerShell

Get-RGGroupResourceList

次の例は、Get-RGGroupResourceList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リソースタイプでフィルタリングされた に基づいてグループリソースを一覧表示します。

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListGroupResources」の「」を参照してください。 AWS Tools for PowerShell

Get-RGResourceTag

次の例は、Get-RGResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたリソースグループの ARN のタグを一覧表示します。

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

出力:

```
Key       Value
---       -----
Instances workboxes
```

- API の詳細については、「 コマンドレットリファレンスGetTags」の「」を参照してください。 AWS Tools for PowerShell

New-RGGroup

次の例は、New-RGGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 という名前の新しいタグベースの AWS Resource Groups リソースグループを作成します TestPowerShellGroup。グループには、タグキー「Name」とタグ値「test2」でタグ付けされた現在のリージョンの Amazon EC2 インスタンスが含まれます。コマンドは、クエリとグループのタイプ、および オペレーションの結果を返します。

```
$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."
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-RGGroup

次の例は、Remove-RGGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、名前付きリソースグループを削除します。

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-RGResourceTag

次の例は、Remove-RGResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リソースグループから前述のタグを削除します。

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンス」の「タグを解除する」を参照してください。 AWS Tools for PowerShell

Update-RGGroup

次の例は、Update-RGGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、グループの説明を更新します。

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

出力:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- API の詳細については、「 コマンドレットリファレンスUpdateGroup」の「」を参照してください。 AWS Tools for PowerShell

Update-RGGroupQuery

次の例は、Update-RGGroupQuery を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、クエリオブジェクトを作成し、グループのクエリを更新します。

```
$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
```

出力:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- API の詳細については、「 コマンドレットリファレンスUpdateGroupQuery」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Resource Groups Tagging API の例 PowerShell

次のコード例は、Resource Groups Tagging API AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Add-RGTResourceTag`

次の例は、`Add-RGTResourceTag` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、値「beta」と「preprod_test」を含むタグキー「stage」と「version」を Amazon S3 バケットと Amazon DynamoDB テーブルに追加します。タグを適用するために、サービスに対して 1 回の呼び出しが行われます。

```
$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" }
```

例 2: この例では、指定されたタグと値を Amazon S3 バケットと Amazon DynamoDB テーブルに追加します。コマンドレットにパイプされたリソース ARN ごとに 1 つずつ、サービスに対して 2 つの呼び出しが行われます。

```
$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" }
```

- API の詳細については、「 コマンドレットリファレンスTagResources」の「」を参照してください。 AWS Tools for PowerShell

Get-RGTResource

次の例は、Get-RGTResource を使用する方法を説明しています。

のツール PowerShell

例 1: リージョン内のすべてのタグ付きリソースと、リソースに関連付けられたタグキーを返します。コマンドレットに -Region パラメータが指定されていない場合、シェルまたは EC2 インスタンスメタデータからリージョンを推測しようとします。

```
Get-RGTResource
```

出力:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

例 2: リージョン内の指定されたタイプのタグ付けされたリソースをすべて返します。各サービス名とリソースタイプの文字列は、リソースの Amazon リソースネーム (ARN) に埋め込まれている文字列と同じです。

```
Get-RGTResource -ResourceType "s3"
```

出力:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::mybucket                                           {stage, version,
 othertag}
```

例 3: リージョン内の指定されたタイプのタグ付けされたリソースをすべて返します。リソースタイプを コマンドレットにパイプすると、指定されたリソースタイプごとに サービスへの呼び出しが 1 回行われることに注意してください。

```
"dynamodb","s3" | Get-RGTResource
```

出力:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

例 4: 指定されたフィルターに一致するすべてのタグ付けされたリソースを返します。

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

出力:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:s3:::mybucket                                                {stage, version,
 othertag}
```

例 5: 指定されたフィルターとリソースタイプに一致するすべてのタグ付けされたリソースを返します。

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

出力:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
```

例 6: 指定されたフィルターに一致するすべてのタグ付けされたリソースを返します。

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

出力:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- API の詳細については、「 コマンドレットリファレンスGetResources」の「」を参照してください。 AWS Tools for PowerShell

Get-RGTTagKey

次の例は、Get-RGTTagKey を使用する方法を説明しています。

のツール PowerShell

例 1: 指定したリージョンのすべてのタグキーを返します。-Region パラメータが指定されていない場合、 コマンドレットはデフォルトのシェルリージョンまたは EC2 インスタンスメタデータからリージョンを推測しようとします。タグキーは特定の順序で返されないことに注意してください。

```
Get-RGTTagKey -region us-west-2
```

出力:

```
version
stage
```

- API の詳細については、「 コマンドレットリファレンスGetTagKeys」の「」を参照してください。 AWS Tools for PowerShell

Get-RGTTagValue

次の例は、Get-RGTTagValue を使用する方法を説明しています。

のツール PowerShell

例 1: リージョン内の指定されたタグの値を返します。-Region パラメータが指定されていない場合、 コマンドレットはデフォルトのシェルリージョンまたは EC2 インスタンスメタデータからリージョンを推測しようとします。

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

出力:

```
beta
```

- API の詳細については、「 コマンドレットリファレンスGetTagValues」の「」を参照してください。 AWS Tools for PowerShell

`Remove-RGTResourceTag`

次の例は、Remove-RGTResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: Amazon S3 バケットと Amazon DynamoDB テーブルからタグキー「stage」と「version」、および関連する値を削除します。 DynamoDB タグを削除するため、サービスが 1 回呼び出されます。タグを削除する前に、コマンドレットで確認を求められます。確認をバイパスするには、-Force パラメータを追加します。

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

例 2: Amazon S3 バケットと Amazon DynamoDB テーブルからタグキー「stage」と「version」、および関連する値を削除します。 DynamoDB コマンドレットにパイプされたリソース ARN ごとに 1 つずつ、サービスに対して 2 つの呼び出しが行われます。各呼び出しの前に、コマンドレットに確認を求めるプロンプトが表示されます。確認をバイパスするには、-Force パラメータを追加します。

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- API の詳細については、「 コマンドレットリファレンスUntagResources」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Route 53 の例 PowerShell

次のコード例は、Route 53 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

アクション

`Edit-R53ResourceRecordSet`

次の例は、`Edit-R53ResourceRecordSet` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、www.example.com の A レコードを作成し、test.example.com の A レコードを 192.0.2.3 から 192.0.2.1 に変更します。変更の TXT タイプのレコードの値は二重引用符で囲む必要があることに注意してください。詳細については、Amazon Route 53 のドキュメントを参照してください。Get-R53Change コマンドレットを使用してポーリングし、変更が完了したタイミングを判断できます。

```
$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
```

例 2: この例では、エイリアスリソースレコードセットを作成する方法を示します。Z222222222」は、エイリアスリソースレコードセットを作成する Amazon Route 53 ホストゾーンの ID です。example.com」は、エイリアスを作成するゾーン頂点で、www.example.com」は、エイリアスも作成するサブドメインです。Z1111111111111」はロードバランサーのホストゾーン ID の例で、example-load-balancer「-1111111111.us-east-1.elb.amazonaws.com」は Amazon Route 53 が example.com および www.example.com のクエリに応答するロードバランサードメイン名の例です。詳細については、Amazon Route 53 のドキュメントを参照してください。Get-R53Change コマンドレットを使用してポーリングし、変更が完了したタイミングを判断できます。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
```

```
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets, one
 for the zone apex, example.com, and one for www.example.com, that both point to
 example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

例 3: この例では、www.example.com 用に 2 つの A レコードを作成します。Amazon Route 53 は、4 分の 1 の時間 (1/(1+3)) で、最初のリソースレコードセット (192.0.2.9 および 192.0.2.10) の 2 つの値で www.example.com のクエリに応答します。Amazon Route 53 は 4 分の 3 の時間 (3/(1+3)) で、2 番目のリソースレコードセット (192.0.2.11 および 192.0.2.12) の 2 つの値で www.example.com のクエリに応答します。詳細については、Amazon Route 53 のドキュメントを参照してください。Get-R53Change コマンドレットを使用してポーリングし、変更が完了したタイミングを判断できます。

```
$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
```

例 4: この例では、example.com が加重エイリアスリソースレコードセットを作成するドメインであると仮定して、加重エイリアスリソースレコードセットを作成する方法を示します。 は、2 つの加重エイリアスリソースレコードセットを互いに SetIdentifier 区別します。Name 要素と Type 要素は両方のリソースレコードセットで同じ値を持つため、この要素は必須です。Z1111111111111および Z3333333333333 は、DNSName の値で指定された ELB ロードバランサーのホストゾーン IDs の例です。example-load-balancer2222222222.us-east-1.elb.amazonaws.com および example-load-balancer-4444444444.us-east-1.elb.amazonaws.com は、Amazon Route 53 が のクエリに応答する Elastic Load Balancing ドメインの例です。 DNSName example.com 詳細については、Amazon Route 53 のドキュメントを参照してください。Get-R53Change コマンドレットを使用してポーリングし、変更が完了したタイミングを判断できます。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
```

```
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first ELB
 domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

例 5: この例では、2 つのレイテンシーエイリアスリソースレコードセットを作成します。1 つは米国西部 (オレゴン) リージョン (us-west-2) の ELB ロードバランサー用、もう 1 つはアジアパシフィック (シンガポール) リージョン (ap-southeast-1) のロードバランサー用です。詳細については、Amazon Route 53 のドキュメントを参照してください。Get-R53Change コマンドレットを使用してポーリングし、変更が完了したタイミングを判断できます。

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
```

```
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
 sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
 region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- API の詳細については、「 コマンドレットリファレンスChangeResourceRecordSets」の「」を参照してください。 AWS Tools for PowerShell

Get-R53AccountLimit

次の例は、Get-R53AccountLimit を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在のアカウントを使用して作成できるホストゾーンの最大数を返します。

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

出力:

```
15
```

- API の詳細については、「 コマンドレットリファレンスGetAccountLimit」の「」を参照してください。 AWS Tools for PowerShell

Get-R53CheckerIpRanges

次の例は、Get-R53CheckerIpRanges を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Route53 ヘルスチェッカーCIDRs を返します。

```
Get-R53CheckerIpRanges
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetCheckerIpRanges」の「」を参照してください。 AWS Tools for PowerShell

Get-R53HostedZone

次の例は、Get-R53HostedZone を使用する方法を説明しています。

のツール PowerShell

例 1: ID Z1D633PJN98FT9 のホストゾーンの詳細を返します。

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- API の詳細については、「 コマンドレットリファレンスGetHostedZone」の「」を参照してください。 AWS Tools for PowerShell

Get-R53HostedZoneCount

次の例は、Get-R53HostedZoneCount を使用する方法を説明しています。

のツール PowerShell

例 1: 現在の のパブリックホストゾーンとプライベートホストゾーンの合計数を返します AWS アカウント。

```
Get-R53HostedZoneCount
```

- API の詳細については、「 コマンドレットリファレンスGetHostedZoneCount」の「」を参照してください。 AWS Tools for PowerShell

Get-R53HostedZoneLimit

次の例は、Get-R53HostedZoneLimit を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたホストゾーンで作成できるレコードの最大数の制限を返します。

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

出力:

```
5
```

- API の詳細については、「 コマンドレットリファレンスGetHostedZoneLimit」の「」を参照してください。 AWS Tools for PowerShell

Get-R53HostedZoneList

次の例は、Get-R53HostedZoneList を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのパブリックホストゾーンとプライベートホストゾーンを出力します。

```
Get-R53HostedZoneList
```

例 2: ID NZ8X2CISAMPLE を持つ再利用可能な委任セットに関連付けられているすべてのホストゾーンを出力します

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- API の詳細については、「 コマンドレットリファレンスListHostedZones」の「」を参照してください。 AWS Tools for PowerShell

Get-R53HostedZonesByName

次の例は、Get-R53HostedZonesByName を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのパブリックホストゾーンとプライベートホストゾーンをドメイン名で ASCII 順に返します。

```
Get-R53HostedZonesByName
```

例 2: パブリックホストゾーンとプライベートホストゾーンを、指定した DNS 名からドメイン名の ASCII 順に返します。

```
Get-R53HostedZonesByName -DnsName example2.com
```

例 3: この例は、最初に 1 つの項目を取得し、次にすべてのゾーンが返されるまで 2 つずつ反復処理することで、ホストゾーンを手動で列挙する方法を示しています。各呼び出しの後に**$AWSHistory**スタック内のサービスレスポンスにアタッチされたマーカープロパティを使用します。

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- API の詳細については、「 コマンドレットリファレンスListHostedZonesByName」の「」を参照してください。 AWS Tools for PowerShell

Get-R53QueryLoggingConfigList

次の例は、Get-R53QueryLoggingConfigList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、現在の に関連付けられている DNS クエリログ記録のすべての設定を返しますAWS アカウント。

```
Get-R53QueryLoggingConfigList
```

出力:

```
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:*
```

- API の詳細については、「 コマンドレットリファレンスListQueryLoggingConfigs」の「」を参照してください。 AWS Tools for PowerShell

Get-R53ReusableDelegationSet

次の例は、Get-R53ReusableDelegationSet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、委任セットに割り当てられた 4 つのネームサーバーを含む、指定された委任セットに関する情報を取得します。

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスGetReusableDelegationSet」の「」を参照してください。 AWS Tools for PowerShell

New-R53HostedZone

次の例は、New-R53HostedZone を使用する方法を説明しています。

のツール PowerShell

例 1: 再利用可能な委任セットに関連付けられたexample.com」という名前の新しいホストゾーンを作成します。オペレーションを 2 回実行するリスクなしに、必要に応じて再試行する必要があるリクエストを返すには、 CallerReference パラメータの値を指定する必要があることに注意してください。ホストゾーンは VPC で作成されているため、自動的にプライベートになるため、-HostedZoneConfig_PrivateZone parameter を設定しないでください。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateHostedZone」の「」を参照してください。 AWS Tools for PowerShell

New-R53QueryLoggingConfig

次の例は、New-R53QueryLoggingConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたホストゾーンの新しい Route53 DNS クエリログ記録設定を作成します。Amazon Route53 は、指定された Cloudwatch ロググループに DNS クエリログを発行します。

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

出力:

```
QueryLoggingConfig                      Location
------------------                      --------
```

```
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- API の詳細については、「 コマンドレットリファレンスCreateQueryLoggingConfig」の「」を参照してください。 AWS Tools for PowerShell

New-R53ReusableDelegationSet

次の例は、New-R53ReusableDelegationSet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、複数のホストゾーンで再利用できる 4 つのネームサーバーの再利用可能な委任セットを作成します。

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

出力:

```
DelegationSet                       Location
-------------                       --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- API の詳細については、「 コマンドレットリファレンスCreateReusableDelegationSet」の「」を参照してください。 AWS Tools for PowerShell

Register-R53VPCWithHostedZone

次の例は、Register-R53VPCWithHostedZone を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC をプライベートホストゾーンに関連付けます。

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

出力:

```
Id                    Status  SubmittedAt         Comment
```

```
--                      ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」のAssociateVPCWithHostedZone」を参照してください。

Remove-R53HostedZone

次の例は、Remove-R53HostedZone を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された ID のホストゾーンを削除します。-Force スイッチパラメータを追加しない限り、コマンドが続行する前に確認を求められます。

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- API の詳細については、「 コマンドレットリファレンスDeleteHostedZone」の「」を参照してください。 AWS Tools for PowerShell

Remove-R53QueryLoggingConfig

次の例は、Remove-R53QueryLoggingConfig を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、DNS クエリログ記録用に指定された設定を削除します。

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- API の詳細については、「 コマンドレットリファレンスDeleteQueryLoggingConfig」の「」を参照してください。 AWS Tools for PowerShell

Remove-R53ReusableDelegationSet

次の例は、Remove-R53ReusableDelegationSet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された再利用可能な委任セットを削除します。

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- API の詳細については、「 コマンドレットリファレンスDeleteReusableDelegationSet」の「」を参照してください。 AWS Tools for PowerShell

Unregister-R53VPCFromHostedZone

次の例は、Unregister-R53VPCFromHostedZone を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された VPC とプライベートホストゾーンの関連付けを解除します。

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

出力:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- API の詳細については、「 コマンドレットリファレンスDisassociateVPCFromHostedZone」を参照してください。 AWS Tools for PowerShell

Update-R53HostedZoneComment

次の例は、Update-R53HostedZoneComment を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定されたホストゾーンのコメントを更新します。

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

出力:

```
Id                    : /hostedzone/Z385PDS6AAAAAR
```

```
Name                   : example.com.
CallerReference        : C5B55555-7147-EF04-8341-69131E805C89
Config                 : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService          :
```

- API の詳細については、「 コマンドレットリファレンスUpdateHostedZoneComment」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon S3 の例 PowerShell

次のコード例は、Amazon S3 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Copy-S3Object`

次の例は、`Copy-S3Object` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」から同じバケットにコピーします。ただし、新しいキー「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
```

例 2: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からバケット「backup-files」にコピーします。ただし、キーは「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-copy.txt
 -DestinationBucket backup-files
```

例 3: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からダウンロードして、「local-sample.txt」という名前のローカルファイルに保存します。

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

例 4: 指定したファイルに 1 つのオブジェクトをダウンロードします。ダウンロードしたファイルは c:\downloads\data\archive.zip に保存されます。

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:\downloads
```

例 5: 指定した key prefix と一致するすべてのオブジェクトをローカルフォルダにダウンロードします。相対的なキー階層は、ダウンロードの場所全体のサブフォルダとして保存されます。

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- API の詳細については、「 コマンドレットリファレンスCopyObject」の「」を参照してください。 AWS Tools for PowerShell

Get-S3ACL

次の例は、Get-S3ACL を使用する方法を説明しています。

のツール PowerShell

例 1: コマンドは、S3 オブジェクトのオブジェクト所有者の詳細を取得します。

```
Get-S3ACL -BucketName 's3casetestbucket' -key 'initialize.ps1' -Select
 AccessControlList.Owner
```

出力:

```
DisplayName Id
```

```
---------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンスGetACL」を参照してください。

Get-S3Bucket

次の例は、Get-S3Bucket を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドはすべての S3 バケットを返します。

```
Get-S3Bucket
```

例 2: このコマンドは「test-files」という名前のバケットを返します。

```
Get-S3Bucket -BucketName test-files
```

- API の詳細については、「 コマンドレットリファレンスListBuckets」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketAccelerateConfiguration

次の例は、Get-S3BucketAccelerateConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: 指定したバケットで転送加速設定が有効になっている場合、このコマンドは Enabled の値を返します。

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、「 コマンドレットリファレンスGetBucketAccelerateConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketAnalyticsConfiguration

次の例は、Get-S3BucketAnalyticsConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターの詳細を返します。

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketAnalyticsConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketAnalyticsConfigurationList

次の例は、Get-S3BucketAnalyticsConfigurationList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの最初の 100 つの分析設定を返します。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- API の詳細については、「 コマンドレットリファレンスListBucketAnalyticsConfigurations」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketEncryption

次の例は、Get-S3BucketEncryption を使用する方法を説明しています。

のツール PowerShell

例 1: のコマンドは、指定したバケットに関連付けられたすべてのサーバー側暗号化ルールを返します。

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketEncryption」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketInventoryConfiguration

次の例は、Get-S3BucketInventoryConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testinventory」という名前のインベントリの詳細を返します。

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketInventoryConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketInventoryConfigurationList

次の例は、Get-S3BucketInventoryConfigurationList を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの最初の 100 つのインベントリ設定を返します。

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスListBucketInventoryConfigurations」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketLocation

次の例は、Get-S3BucketLocation を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、制約が存在する場合、バケット「s3testbucket」の場所の制約を返します。

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

出力:

```
Value
-----
ap-south-1
```

- API の詳細については、「 コマンドレットリファレンスGetBucketLocation」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketLogging

次の例は、Get-S3BucketLogging を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットのログ記録ステータスを返します。

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

出力:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- API の詳細については、「 コマンドレットリファレンスGetBucketLogging」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketMetricsConfiguration

次の例は、Get-S3BucketMetricsConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの「testfilter」という名前のメトリクスフィルターに関する詳細を返します。

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId 'testfilter'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketMetricsConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketNotification

次の例は、Get-S3BucketNotification を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したバケットの通知設定を取得します。

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

出力:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API の詳細については、「 コマンドレットリファレンスGetBucketNotification」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketPolicy

次の例は、Get-S3BucketPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを出力します。

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketPolicy」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketPolicyStatus

次の例は、Get-S3BucketPolicyStatus を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットのポリシーステータスを返し、バケットがパブリックかどうかを示します。

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketPolicyStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketReplication

次の例は、Get-S3BucketReplication を使用する方法を説明しています。

のツール PowerShell

例 1: mybucket」という名前のバケットに設定されているレプリケーション設定の情報を返します。

```
Get-S3BucketReplication -BucketName mybucket
```

- API の詳細については、「 コマンドレットリファレンスGetBucketReplication」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketRequestPayment

次の例は、Get-S3BucketRequestPayment を使用する方法を説明しています。

のツール PowerShell

例 1: 「mybucket」という名前のバケットのリクエストの支払い設定を返します。デフォルトでは、バケット所有者はバケットからのダウンロード料金を支払います。

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- API の詳細については、「 コマンドレットリファレンスGetBucketRequestPayment」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketTagging

次の例は、Get-S3BucketTagging を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットに関連付けられたすべてのタグを返します。

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketTagging」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketVersioning

次の例は、Get-S3BucketVersioning を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットに関するバージョニングステータスを返します。

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketVersioning」の「」を参照してください。 AWS Tools for PowerShell

Get-S3BucketWebsite

次の例は、Get-S3BucketWebsite を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトの設定の詳細を返します。

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetBucketWebsite」の「」を参照してください。 AWS Tools for PowerShell

Get-S3CORSConfiguration

次の例は、Get-S3CORSConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された S3 バケットに対応するすべての CORS 設定ルールを含むオブジェクトを返します。

```
Get-S3CORSConfiguration -BucketName 's3testbucket' -Select Configuration.Rules
```

出力:

```
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 : {}
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」のGetCORSConfiguration」を参照してください。

Get-S3LifecycleConfiguration

次の例は、Get-S3LifecycleConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、バケットのライフサイクル設定を取得します。

```
Get-S3LifecycleConfiguration -BucketName test-bla
```

出力:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- API の詳細については、「 コマンドレットリファレンスGetLifecycleConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3Object

次の例は、Get-S3Object を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、バケット「test-files」内のすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files
```

例 2: このコマンドは、バケット「test-files」内の項目「sample.txt」に関する情報を取得します。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

例 3: このコマンドは、バケット「test-files」からプレフィックス「sample」を持つすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- API の詳細については、「 コマンドレットリファレンスListObjects」の「」を参照してください。 AWS Tools for PowerShell

Get-S3ObjectLockConfiguration

次の例は、Get-S3ObjectLockConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: 指定した S3 バケットのオブジェクトロック設定が有効になっている場合、このコマンドは値「Enabled」を返しま。

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、「 コマンドレットリファレンスGetObjectLockConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Get-S3ObjectMetadata

次の例は、Get-S3ObjectMetadata を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された S3 バケット内のキーListTrusts「.txt」を持つオブジェクトのメタデータを返します。

```
Get-S3ObjectMetadata -BucketName 's3testbucket' -Key 'ListTrusts.txt'
```

出力:

```
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                                :
```

- API の詳細については、「 コマンドレットリファレンスGetObjectMetadata」の「」を参照してください。 AWS Tools for PowerShell

Get-S3ObjectRetention

次の例は、Get-S3ObjectRetention を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、オブジェクトが保持されるまでのモードと日付を返します。

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- API の詳細については、「 コマンドレットリファレンスGetObjectRetention」の「」を参照してください。 AWS Tools for PowerShell

Get-S3ObjectTagSet

次の例は、Get-S3ObjectTagSet を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定した S3 バケット上に存在するオブジェクトに関連付けられたタグを返します。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

出力:

```
Key  Value
---  -----
test value
```

- API の詳細については、「 コマンドレットリファレンスGetObjectTagging」の「」を参照してください。 AWS Tools for PowerShell

Get-S3PreSignedURL

次の例は、Get-S3PreSignedURL を使用する方法を説明しています。

のツール PowerShell

例 1: コマンドは、指定されたキーと有効期限の署名付き URL を返します。

```
Get-S3PreSignedURL -BucketName 's3testbucket' -Key 'testkey' -Expires '2023-11-16'
```

例 2: コマンドは、指定されたキーと有効期限を持つディレクトリバケットの署名付き URL を返します。

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName sampledirectorybucket--use1-az5--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- API の詳細については、「 コマンドレットリファレンス」のGetPreSigned「URL」を参照してください。 AWS Tools for PowerShell

Get-S3PublicAccessBlock

次の例は、Get-S3PublicAccessBlock を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットのパブリックアクセスブロック設定を返します。

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスGetPublicAccessBlock」の「」を参照してください。 AWS Tools for PowerShell

Get-S3Version

次の例は、Get-S3Version を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された S3 バケット内のすべてのバージョンのオブジェクトに関するメタデータを返します。

```
Get-S3Version -BucketName 's3testbucket'
```

出力:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : s3testbucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- API の詳細については、「 コマンドレットリファレンスListVersions」の「」を参照してください。 AWS Tools for PowerShell

New-S3Bucket

次の例は、New-S3Bucket を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、「sample-bucket」という名前の新しいプライベートバケットを作成します。

```
New-S3Bucket -BucketName sample-bucket
```

例 2: このコマンドは、読み取り/書き込みアクセス許可を持つ「sample-bucket」という名前の新しいバケットを作成します。

```
New-S3Bucket -BucketName sample-bucket -PublicReadWrite
```

例 3: このコマンドは、読み取り専用アクセス許可を持つ「sample-bucket」という名前の新しいバケットを作成します。

```
New-S3Bucket -BucketName sample-bucket -PublicReadOnly
```

例 4: このコマンドは、「samplebucket--use1-az5--x-s3」という名前の新しいディレクトリバケットを で作成します 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
```

- API の詳細については、「 コマンドレットリファレンスPutBucket」の「」を参照してください。 AWS Tools for PowerShell

`Read-S3Object`

次の例は、`Read-S3Object` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、バケット「test-files」から項目「sample.txt」を取得して、それを現在の場所にある「local-sample.txt」という名前のファイルに保存します。このコマンドを呼び出す前にファイル「local-sample.txt」が存在していなくても構いません。

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

例 2: このコマンドは、バケット「test-files」から仮想ディレクトリ「DIR」を取得し、それを現在の場所の「Local-DIR」という名前のフォルダに保存します。このコマンドを呼び出す前に、フォルダー「Local-DIR」が存在していなくても構いません。

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

例 3: バケット名に「config」を含むバケットから、キーが「.json」で終わるすべてのオブジェクトを、指定したフォルダ内のファイルにダウンロードします。オブジェクトキーはファイル名の設定に使用されます。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- API の詳細については、「 コマンドレットリファレンスGetObject」の「」を参照してください。 AWS Tools for PowerShell

`Remove-S3Bucket`

次の例は、`Remove-S3Bucket` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、バケット「test-files」からすべてのオブジェクトとオブジェクトバージョンを削除してから、バケットを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force スイッチを追加すると、確認メッセージが表示されなくなります。空でないバケットは削除できないことに注意が必要です。

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucket」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketAnalyticsConfiguration

次の例は、Remove-S3BucketAnalyticsConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターを削除します。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketAnalyticsConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketEncryption

次の例は、Remove-S3BucketEncryption を使用する方法を説明しています。

のツール PowerShell

例 1: これは、指定した S3 バケットで有効になっている暗号化を無効にします。

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketEncryption」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketInventoryConfiguration

次の例は、Remove-S3BucketInventoryConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された S3 バケットtestInventoryNameに対応する「」という名前のインベントリを削除します。

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketInventoryConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketMetricsConfiguration

次の例は、Remove-S3BucketMetricsConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testmetrics」という名前のメトリクスフィルターを削除します。

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketMetricsConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketPolicy

次の例は、Remove-S3BucketPolicy を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを削除します。

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketPolicy」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketReplication

次の例は、Remove-S3BucketReplication を使用する方法を説明しています。

のツール PowerShell

例 1: 「mybucket」という名前のバケットに関連付けられているレプリケーションの設定を削除します。このオペレーションには s3:DeleteReplicationConfiguration action のアクセス許可が必要であることに注意してください。オペレーションを続行する前に確認を求めるプロンプトが表示されます。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-S3BucketReplication -BucketName mybucket
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketReplication」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketTagging

次の例は、Remove-S3BucketTagging を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットに関連付けられたすべてのタグを削除します。

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketTagging」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3BucketWebsite

次の例は、Remove-S3BucketWebsite を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトホスティングのプロパティを無効にします。

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteBucketWebsite」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3CORSConfiguration

次の例は、Remove-S3CORSConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定された S3 バケットの CORS 設定を削除します。

```
Remove-S3CORSConfiguration -BucketName 's3testbucket'
```

出力:

```
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
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス」のDeleteCORSConfiguration」を参照してください。

Remove-S3LifecycleConfiguration

次の例は、Remove-S3LifecycleConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: コマンドは、指定された S3 バケットのすべてのライフサイクルルールを削除します。

```
Remove-S3LifecycleConfiguration -BucketName 's3testbucket'
```

- API の詳細については、「 コマンドレットリファレンスDeleteLifecycleConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3MultipartUpload

次の例は、Remove-S3MultipartUpload を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、5 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

例 2: このコマンドは、2014 年 1 月 2 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

例 3: このコマンドは、2014 年 1 月 2 日 10:45:37 より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02 10:45:37"
```

- API の詳細については、「 コマンドレットリファレンスAbortMultipartUpload」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3Object

次の例は、Remove-S3Object を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、バケット「test-files」からオブジェクト「sample.txt」を削除します。コマンドを実行する前に、確認を求めるプロンプトが表示されます。-Force スイッチを追加すると、確認メッセージが表示されなくなります。

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

例 2: このコマンドは、指定されたバージョンのオブジェクト「sample.txt」をバケット「test-files」から削除します。これは、バケットがオブジェクトバージョンを有効にするように設定されていることを前提としています。

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

例 3: このコマンドは、バケット「test-files」からオブジェクト「sample1.txt」、「sample2.txt」、「sample3.txt」を 1 回のバッチ操作で削除します。このサービスの応答では、削除の成功またはエラーのステータスを問わず、処理されたすべてのキーがリスト表示されます。サービスで処理できなかったキーのエラーのみを取得するには、 - ReportErrorsOnly パラメータを追加します (このパラメータはエイリアス -Quiet で指定することもできます。

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

例 4: この例では、 KeyCollection パラメータを持つインライン式を使用して、削除するオブジェクトのキーを取得します。Get-S3Object は Amazon.S3.Model.S3Object インスタンスのコレクションを返します。各インスタンスには、オブジェクトを識別する文字列型のキーメンバーがあります。

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-files" -
KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

例 5: この例では、バケット内のキープレフィックス「prefix/subprefix」を持つすべてのオブジェクトを取得して削除します。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションでは、コレクションを コマンドレットの -InputObject（ エイリアス -S3ObjectCollection) パラメータに渡して、サービスへの 1 回の呼び出しで削除をバッチとして実行することを検討してください。

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

例 6: この例では、削除マーカーを表す Amazon.S3.Model.S3 ObjectVersion インスタンスのコレクションを削除コマンドレットにパイプします。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションでは、コレクションを コマンドレットの -InputObject（ エイリアス -S3ObjectCollection) パラメータに渡して、サービスへの 1 回の呼び出しで削除をバッチとして実行することを検討してください。

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

例 7: このスクリプトは、 KeyAndVersionCollection パラメータで使用するオブジェクトの配列を構築して、オブジェクトのセット (この場合は削除マーカー) のバッチ削除を実行する方法を示しています。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスDeleteObjects」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3ObjectTagSet

次の例は、Remove-S3ObjectTagSet を使用する方法を説明しています。

のツール PowerShell

このコマンドは、指定した S3 バケット内のキー「testfile.txt」を持つオブジェクトに関連付けられたすべてのタグを削除します。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select '^Key'
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteObjectTagging」の「」を参照してください。 AWS Tools for PowerShell

Remove-S3PublicAccessBlock

次の例は、Remove-S3PublicAccessBlock を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットのブロックパブリックアクセス設定をオフにします。

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select '^BucketName'
```

出力:

```
s3testbucket
```

- API の詳細については、「 コマンドレットリファレンスDeletePublicAccessBlock」の「」を参照してください。 AWS Tools for PowerShell

Set-S3BucketEncryption

次の例は、Set-S3BucketEncryption を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットで Amazon S3 マネージドキー (SSE-S3) を使用したデフォルトの AES256 サーバー側暗号化を有効にします。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API の詳細については、「 コマンドレットリファレンスPutBucketEncryption」の「」を参照してください。 AWS Tools for PowerShell

Test-S3Bucket

次の例は、Test-S3Bucket を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、バケットが存在する場合は True、それ以外の場合は False を返します。このコマンドは、バケットがユーザーに属していない場合でも True を返します。

```
Test-S3Bucket -BucketName test-files
```

- API の詳細については、「 コマンドレットリファレンスTest-S3Bucket」の「」を参照してください。 AWS Tools for PowerShell

Write-S3BucketAccelerateConfiguration

次の例は、Write-S3BucketAccelerateConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットの転送加速を有効にします。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
```

```
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- API の詳細については、「 コマンドレットリファレンスPutBucketAccelerateConfiguration」の「」を参照してください。 AWS Tools for PowerShell

Write-S3BucketNotification

次の例は、Write-S3BucketNotification を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、S3 イベントの SNS トピック設定を設定し ObjectRemovedDelete 、指定された s3 バケットの通知を有効にします。

```
$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
```

例 2: この例では、Lambda 関数に送信する特定のバケット ObjectCreatedAll の の通知を有効にします。

```
$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
```

例 3: この例では、異なるキーサフィックスに基づいて 2 つの異なる Lambda 設定を作成し、両方を 1 つのコマンドで設定します。

```
#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
```

- API の詳細については、「 コマンドレットリファレンスPutBucketNotification」の「」を参照してください。 AWS Tools for PowerShell

Write-S3BucketReplication

次の例は、Write-S3BucketReplication を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、バケット「examplebucket」のキー名のプレフィックス「」で作成された新しいオブジェクトをTaxDocs「exampletargetbucket」バケットにレプリケーションできるようにする単一のルールを使用してレプリケーション設定を設定します。

```
$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
```

例 2: この例では、キー名のプレフィックスTaxDocs「」または「」のいずれかで作成された新しいオブジェクトを「exampletargetbucket」バケットにレプリケーションできるようにする複数のルールを使用してレプリケーション設定を設定しますOtherDocs。キーのプレフィックスの重複は許可されません。

```
$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
```

例 3: この例では、指定されたバケットのレプリケーション設定を更新して、バケット「exampletargetbucketTaxDocs」へのキー名のプレフィックス「」を持つオブジェクトのレプリケーションを制御するルールを無効にします。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスPutBucketReplication」の「」を参照してください。 AWS Tools for PowerShell

`Write-S3BucketRequestPayment`

次の例は、`Write-S3BucketRequestPayment` を使用する方法を説明しています。

のツール PowerShell

例 1: 「mybucket」という名前のバケットのリクエスト支払い設定を更新して、バケットからのダウンロードをリクエストしたユーザーにダウンロード料金が請求されるようにします。デフォルトでは、バケット所有者がダウンロード料金を支払います。リクエストの支払いをデフォルトに戻すには、 RequestPaymentConfiguration_Payer パラメータに 'BucketOwner' を使用します。

```
Write-S3BucketRequestPayment -BucketName mybucket -RequestPaymentConfiguration_Payer
 Requester
```

- API の詳細については、「 コマンドレットリファレンスPutBucketRequestPayment」の「」を参照してください。 AWS Tools for PowerShell

`Write-S3BucketTagging`

次の例は、`Write-S3BucketTagging` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、**`cloudtrail-test-2018`** という名前のバケットに 2 つのタグを適用します。1 つは Stage のキーと値が Test のタグで、もう 1 つはキーが Environment で、値は Alpha のタグです。タグがバケットに追加されたことを確認するには、**`Get-S3BucketTagging -BucketName bucket_name`** を実行します。結果では、最初のコマンドでバケットに適用したタグが表示されるはずです。**`Write-S3BucketTagging`** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。個別のタグを追加または削除するには、リソースグループとタグ付けの API コマンドレット、**`Add-RGTResourceTag`**、**`Remove-RGTResourceTag`** を実行します。または、 AWS マネジメントコンソールのタグエディタを使用して S3 バケットタグを管理します。

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

例 2: このコマンドは、バケット **`cloudtrail-test-2018`** を **`Write-S3BucketTagging`** コマンドレットにパイプします。これにより、Stage:Production と Department:Finance というタグがバケットに適用されます。**`Write-S3BucketTagging`** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- API の詳細については、「 コマンドレットリファレンスPutBucketTagging」の「」を参照してください。 AWS Tools for PowerShell

Write-S3BucketVersioning

次の例は、Write-S3BucketVersioning を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケットのバージョニングを有効にします。

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status Enabled
```

- API の詳細については、「 コマンドレットリファレンスPutBucketVersioning」の「」を参照してください。 AWS Tools for PowerShell

Write-S3BucketWebsite

次の例は、Write-S3BucketWebsite を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定したバケットのウェブサイトのホスティングを有効にして、インデックスドキュメントを「index.html」、エラードキュメントを「error.html」と指定します。

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API の詳細については、「 コマンドレットリファレンスPutBucketWebsite」の「」を参照してください。 AWS Tools for PowerShell

Write-S3LifecycleConfiguration

次の例は、Write-S3LifecycleConfiguration を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、$ で指定された設定を書き込み/置き換えますNewRule。この設定では、指定されたプレフィックスとタグ値を持つスコープオブジェクトを必ず制限します。

```
$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
```

例 2: この例では、フィルタリングを使用して複数のルールを設定します。$ArchiveRule Glacier に 30 日後にアーカイブするオブジェクトを に設定し、120 を に設定します DeepArchive。$ は、「py」プレフィックスと tag:key「archieved」が「yes」に設定されているオブジェクトについて、現在のバージョンと以前のバージョンの両方を 150 日以内にExpireRule 期限切れにします。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスPutLifecycleConfiguration」の「」を参照してください。 AWS Tools for PowerShell

`Write-S3Object`

次の例は、`Write-S3Object` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

例 2: このコマンドは、単一のファイル「sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。-Key パラメータを指定しない場合、ファイル名が S3 オブジェクトキーとして使用されます。

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

例 3: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「prefix/to/sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

例 4: このコマンドは、サブディレクトリ「Scripts」のすべてのファイルをバケット「test-files」にアップロードし、共通キープレフィックスSampleScripts「」を各オブジェクトに適用します。アップロードされた各ファイルにはSampleScripts「/filename」のキーがあり、「filename」は異なります。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

例 5: このコマンドは、ローカルディレクター「Scripts」のすべての *.ps1 ファイルをバケット「test-files」にアップロードし、共通キープレフィックスSampleScripts「」を各オブジェクトに適用します。アップロードされた各ファイルにはSampleScripts「/filename.ps1」のキーがあり、「filename」は異なります。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\ -
SearchPattern *.ps1
```

例 6: このコマンドは、「sample.txt」というキーを持つ、指定されたコンテンツ文字列を含む新しい S3 オブジェクトを作成します。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object contents"
```

例 7: このコマンドは、指定したファイル (ファイル名をキーとして使用) をアップロードして、指定したタグを新しいオブジェクトに適用します。

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

例 8: このコマンドは、指定したフォルダを再帰的にアップロードして、指定したタグをすべての新しいオブジェクトに適用します。

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse -
TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API の詳細については、「 コマンドレットリファレンスPutObject」の「」を参照してください。 AWS Tools for PowerShell

`Write-S3ObjectRetention`

次の例は、`Write-S3ObjectRetention` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、指定した S3 バケット内の「testfile.txt」オブジェクトの期限日「2019 年 12 月 31 日 00:00:00」までガバナンス保持モードを有効にします。

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API の詳細については、「 コマンドレットリファレンスPutObjectRetention」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した S3 Glacier の例 PowerShell

次のコード例は、S3 Glacier AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-GLCJob`

次の例は、`Get-GLCJob` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたジョブの詳細を返します。ジョブが正常に完了すると、Read-GC JobOutput コマンドレットを使用して、ジョブの内容 (アーカイブまたはインベントリリスト) をローカルファイルシステムに取得できます。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

出力:

```
Action                        : ArchiveRetrieval
ArchiveId                     : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash         : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes            : 38034480
Completed                     : False
CompletionDate                : 1/1/0001 12:00:00 AM
CreationDate                  : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes          : 0
JobDescription                :
JobId                         : op1x...JSbthM
JobOutputPath                 :
OutputLocation                :
RetrievalByteRange            : 0-38034479
SelectParameters              :
SHA256TreeHash                : 79f3ea754c02f58...dc57bf4395b
SNSTopic                      :
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- API の詳細については、「 コマンドレットリファレンスDescribeJob」の「」を参照してください。 AWS Tools for PowerShell

New-GLCVault

次の例は、New-GLCVault を使用する方法を説明しています。

のツール PowerShell

例 1: ユーザーのアカウントの新しいボールトを作成します。AccountId パラメータに値が指定されていないため、コマンドレットは現在のアカウントを示す「-」のデフォルトを使用します。

```
New-GLCVault -VaultName myvault
```

出力:

```
/01234567812/vaults/myvault
```

- API の詳細については、「 コマンドレットリファレンスCreateVault」の「」を参照してください。 AWS Tools for PowerShell

Read-GLCJobOutput

次の例は、Read-GLCJobOutput を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたジョブで取得がスケジュールされたアーカイブコンテンツをダウンロードし、ディスク上のファイルに保存します。ダウンロードによって、チェックサムが利用可能な場合は検証されます。必要に応じて、チェックサムは のようなサービスレスポンス履歴から取得できます (このコマンドレットが最後の実行であったと仮定します): **$AWSHistory.LastServiceResponse**。コマンドレットが最後に実行されなかった場合は、**$AWSHistory.Commands**コレクションを調べて関連するサービスレスポンスを取得します。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- API の詳細については、「 コマンドレットリファレンスGetJobOutput」の「」を参照してください。 AWS Tools for PowerShell

Start-GLCJob

次の例は、Start-GLCJob を使用する方法を説明しています。

のツール PowerShell

例 1: ユーザーが所有する指定されたボールトからアーカイブを取得するジョブを開始します。ジョブのステータスは、Get-GLCJob コマンドレットを使用して確認できます。ジョブが正常に完了すると、Read-GC JobOutput コマンドレットを使用して、ローカルファイルシステムへのアーカイブの内容を取得できます。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

出力:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- API の詳細については、「 コマンドレットリファレンスInitiateJob」の「」を参照してください。 AWS Tools for PowerShell

Write-GLCArchive

次の例は、Write-GLCArchive を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたボールトに 1 つのファイルをアップロードし、アーカイブ ID と計算されたチェックサムを返します。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

出力:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

例 2: フォルダ階層の内容をユーザーのアカウントの指定されたボールトにアップロードします。コマンドレットがアップロードしたファイルごとに、ファイル名、対応するアーカイブ ID、およびアーカイブの計算されたチェックサムが出力されます。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

出力:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin         qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin           39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin           vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
```

```
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- API の詳細については、「 コマンドレットリファレンスUploadArchive」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon SES の例 PowerShell

次のコード例は、Amazon SES AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。 Amazon SES

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。このリンクには GitHub、コンテキスト内でコードを設定および実行する方法の手順が記載されています。

トピック

- アクション

アクション

`Get-SESIdentity`

次の例は、`Get-SESIdentity` を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、検証ステータスに関係なく、特定の AWS アカウントのすべての ID (E メールアドレスとドメイン) を含むリストを返します。

```
Get-SESIdentity
```

- API の詳細については、「 コマンドレットリファレンスListIdentities」の「」を参照してください。 AWS Tools for PowerShell

Get-SESSendQuota

次の例は、Get-SESSendQuota を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、ユーザーの現在の送信制限を返します。

```
Get-SESSendQuota
```

- API の詳細については、「 コマンドレットリファレンスGetSendQuota」の「」を参照してください。 AWS Tools for PowerShell

Get-SESSendStatistic

次の例は、Get-SESSendStatistic を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、ユーザーの送信統計を返します。結果は、過去 2 週間の送信アクティビティを表すデータポイントのリストです。リスト内の各データポイントには、15 分間隔の統計が含まれます。

```
Get-SESSendStatistic
```

- API の詳細については、「 コマンドレットリファレンスGetSendStatistics」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon SNS の例 PowerShell

次のコード例は、Amazon SNS AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Publish-SNSMessage`

次の例は、Publish-SNSMessage を使用する方法を説明しています。

のツール PowerShell

例 1: この例は、インラインで MessageAttribute 宣言された 1 つのメッセージを発行する方法を示しています。

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

例 2: この例は、事前に複数の が MessageAttributes 宣言されたメッセージを発行しているところを示しています。

```
$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
```

- API の詳細については、「 コマンドレットリファレンス」の「パブリッシュ」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon SQS の例 PowerShell

次のコード例は、Amazon SQS AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Add-SQSPermission`

次の例は、`Add-SQSPermission` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された が指定されたキューからメッセージを送信 AWS アカウント できるようにします。

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- API の詳細については、「 コマンドレットリファレンスAddPermission」の「」を参照してください。 AWS Tools for PowerShell

Clear-SQSQueue

次の例は、Clear-SQSQueue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキューからすべてのメッセージを削除します。

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスPurgeQueue」の「」を参照してください。 AWS Tools for PowerShell

Edit-SQSMessageVisibility

次の例は、Edit-SQSMessageVisibility を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキュー内の指定された受信ハンドルを持つメッセージの可視性タイムアウトを 10 時間 (10 時間 x 60 分 x 60 秒 = 36000 秒) に変更します。

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- API の詳細については、「 コマンドレットリファレンスChangeMessageVisibility」の「」を参照してください。 AWS Tools for PowerShell

Edit-SQSMessageVisibilityBatch

次の例は、Edit-SQSMessageVisibilityBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したキューで指定した受信ハンドルを持つ 2 つのメッセージの可視性タイムアウトを変更します。最初のメッセージの可視性タイムアウトは 10 時間 (10 時間 x 60 分 x 60 秒 = 36000 秒) に変更されます。2 番目のメッセージの可視性タイムアウトは 5 時間 (5 時間 x 60 分 x 60 秒 = 18000 秒) に変更されます。

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
```

```
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

出力:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- API の詳細については、「 コマンドレットリファレンスChangeMessageVisibilityBatch」の「」を参照してください。 AWS Tools for PowerShell

Get-SQSDeadLetterSourceQueue

次の例は、Get-SQSDeadLetterSourceQueue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキューをデッドレターキューとして依存するキューの URLs を一覧表示します。

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

出力:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- API の詳細については、「 コマンドレットリファレンスListDeadLetterSourceQueues」の「」を参照してください。 AWS Tools for PowerShell

Get-SQSQueue

次の例は、Get-SQSQueue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのキューを一覧表示します。

```
Get-SQSQueue
```

出力:

```
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
```

例 2: この例では、指定した名前で始まるキューを一覧表示します。

```
Get-SQSQueue -QueueNamePrefix My
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListQueues」の「」を参照してください。 AWS Tools for PowerShell

Get-SQSQueueAttribute

次の例は、Get-SQSQueueAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したキューのすべての属性を一覧表示します。

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

出力:

```
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]...}
```

例 2: この例では、指定されたキューの指定された属性のみを個別に一覧表示します。

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

出力:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                :
 {"Version":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                       495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                       398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                       arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- API の詳細については、「 コマンドレットリファレンスGetQueueAttributes」の「」を参照してください。 AWS Tools for PowerShell

Get-SQSQueueUrl

次の例は、Get-SQSQueueUrl を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前のキューの URL を一覧表示します。

```
Get-SQSQueueUrl -QueueName MyQueue
```

出力:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスGetQueueUrl」の「」を参照してください。 AWS Tools for PowerShell

New-SQSQueue

次の例は、New-SQSQueue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された名前のキューを作成します。

```
New-SQSQueue -QueueName MyQueue
```

出力:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスCreateQueue」の「」を参照してください。 AWS Tools for PowerShell

Receive-SQSMessage

次の例は、Receive-SQSMessage を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したキューに対して受信される次の 10 件のメッセージまでの情報を一覧表示します。情報には、指定されたメッセージ属性が存在する場合、その値が含まれます。

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

出力:

```
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==
```

- API の詳細については、「 コマンドレットリファレンスReceiveMessage」の「」を参照してください。 AWS Tools for PowerShell

Remove-SQSMessage

次の例は、Remove-SQSMessage を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された受信ハンドルを持つメッセージを指定されたキューから削除します。

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- API の詳細については、「 コマンドレットリファレンスDeleteMessage」の「」を参照してください。 AWS Tools for PowerShell

Remove-SQSMessageBatch

次の例は、Remove-SQSMessageBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された受信ハンドルを持つ 2 つのメッセージを指定されたキューから削除します。

```
$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="
```

```
Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

出力:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- API の詳細については、「 コマンドレットリファレンスDeleteMessageBatch」の「」を参照してください。 AWS Tools for PowerShell

Remove-SQSPermission

次の例は、Remove-SQSPermission を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定したラベルのアクセス許可設定を指定したキューから削除します。

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスRemovePermission」の「」を参照してください。 AWS Tools for PowerShell

Remove-SQSQueue

次の例は、Remove-SQSQueue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定されたキューを削除します。

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスDeleteQueue」の「」を参照してください。 AWS Tools for PowerShell

Send-SQSMessage

次の例は、Send-SQSMessage を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された属性とメッセージ本文を持つメッセージを指定されたキューに送信し、メッセージ配信が 10 秒間遅延します。

```
$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
```

出力:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- API の詳細については、「 コマンドレットリファレンスSendMessage」の「」を参照してください。 AWS Tools for PowerShell

Send-SQSMessageBatch

次の例は、Send-SQSMessageBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された属性とメッセージ本文を持つ 2 つのメッセージを指定されたキューに送信します。配信は、最初のメッセージでは 15 秒間、2 番目のメッセージでは 10 秒間遅延します。

```
$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
```

出力:

```
Failed    Successful
```

```
------    ----------

{}        {FirstMessage, SecondMessage}
```

- API の詳細については、「 コマンドレットリファレンスSendMessageBatch」の「」を参照してください。 AWS Tools for PowerShell

Set-SQSQueueAttribute

次の例は、Set-SQSQueueAttribute を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、キューを SNS トピックにサブスクライブするポリシーを設定する方法を示します。メッセージがトピックに発行されると、サブスクライブされたキューにメッセージが送信されます。

```
# 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 }
```

例 2: この例では、指定されたキューに指定された属性を設定します。

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API の詳細については、「 コマンドレットリファレンスSetQueueAttributes」の「」を参照してください。 AWS Tools for PowerShell

AWS STS Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS STS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`Convert-STSAuthorizationMessage`

次の例は、`Convert-STSAuthorizationMessage` を使用する方法を説明しています。

のツール PowerShell

例 1: リクエストへの応答として返された、指定されたエンコード済みメッセージコンテンツに含まれる追加情報をデコードします。承認ステータスの詳細は、アクションをリクエストしたユーザーが見てはならない特権情報である可能性があるため、追加情報はエンコードされます。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- API の詳細については、「 コマンドレットリファレンスDecodeAuthorizationMessage」の「」を参照してください。 AWS Tools for PowerShell

Get-STSFederationToken

次の例は、Get-STSFederationToken を使用する方法を説明しています。

のツール PowerShell

例 1: 「Bob」をフェデレーションユーザーの名前として使用して、1 時間有効なフェデレーショントークンをリクエストします。この名前は、リソースベースのポリシー (Amazon S3 バケットポリシーなど) のフェデレーションユーザー名を参照するために使用できます。提供される JSON 形式の IAM ポリシーを使用して、IAM ユーザーが利用できるアクセス権限の範囲を絞り込みます。指定されたポリシーでは、リクエストしたユーザーに付与されたアクセス権限よりも多くのアクセス権限を付与することはできません。フェデレーションユーザーの最終的なアクセス権限は、渡されたポリシーと IAM ユーザーポリシーの共通部分に基づいて最も制限の厳しい一式となります。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- API の詳細については、「 コマンドレットリファレンスGetFederationToken」の「」を参照してください。 AWS Tools for PowerShell

Get-STSSessionToken

次の例は、Get-STSSessionToken を使用する方法を説明しています。

のツール PowerShell

例 1: 一定期間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。一時的な認証情報をリクエストするために使用される認証情報は、現在の

シェルのデフォルトから推測されます。他の認証情報を指定するには、- ProfileName または - AccessKey/- SecretKey パラメータを使用します。

```
Get-STSSessionToken
```

出力:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

例 2: 1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。リクエストを行うために使用される認証情報は、指定されたプロファイルから取得されます。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

出力:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

例 3: プロファイル「myprofilename」で認証情報が指定されているアカウントに関連付けられた MFA デバイスの識別番号とデバイスから提供された値を使用して、1 時間有効な一時的な認証情報を含む **Amazon.RuntimeAWSCredentials** インスタンスを返します。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

出力:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
```

```
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- API の詳細については、「 コマンドレットリファレンスGetSessionToken」の「」を参照してください。 AWS Tools for PowerShell

Use-STSRole

次の例は、Use-STSRole を使用する方法を説明しています。

のツール PowerShell

例 1: リクエスト元のユーザーが通常アクセスできない AWS リソースにアクセスするために 1 時間使用できる一時的な認証情報 (アクセスキー、シークレットキー、セッショントークン) のセットを返します。返される認証情報には、引き受けているロールのアクセスポリシーと提供されたポリシーで許可されている権限があります (提供されたポリシーを使用して、引き受けているロールのアクセスポリシーで定義されている権限を超える権限を付与することはできません)。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

例 2: 引き受けているロールのアクセスポリシーで定義されているのと同じ権限を持つ、1 時間有効の一時的な認証情報を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

例 3: コマンドレットの実行に使用されるユーザー認証情報に関連付けられている MFA からシリアル番号と生成されたトークンを提供する一時的な認証情報一式を返します。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

例 4: 顧客アカウントで定義されているロールを引き受けた一時的な認証情報一式を返します。サードパーティーが引き受けることができるロールごとに、カスタマーアカウントは、ロールを引き受けるたびに ExternalId パラメータで渡す必要がある識別子を使用してロールを作成する必要があります。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- API の詳細については、「 コマンドレットリファレンスAssumeRole」の「」を参照してください。 AWS Tools for PowerShell

Use-STSWebIdentityRole

次の例は、Use-STSWebIdentityRole を使用する方法を説明しています。

のツール PowerShell

例 1: Login with Amazon ID プロバイダーで認証されたユーザーの一時的な認証情報一式 (1 時間有効) を返します。認証情報は、ロール ARN によって識別されるロールに関連付けられたアクセスポリシーを引き受けます。オプションで、JSON ポリシーを-Policy パラメータに渡して、アクセス権限をさらに絞り込むことができます (ロールに関連付けられている権限で使用可能な権限よりも多くの権限を付与することはできません)。に指定された値は、ID プロバイダーによって返された一意のユーザー識別子WebIdentityToken です。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- API の詳細については、「 コマンドレットリファレンスAssumeRoleWithWebIdentity」の「」を参照してください。 AWS Tools for PowerShell

AWS Support Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Support。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

Add-ASACommunicationToCase

次の例は、Add-ASACommunicationToCase を使用する方法を説明しています。

のツール PowerShell

例 1: 指定したケースに E メール通信の本文を追加します。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

例 2: 指定したケースに E メール通信の本文と、E メールの CC 行に含まれる 1 つ以上の E メールアドレスを追加します。

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- API の詳細については、「 コマンドレットリファレンスAddCommunicationToCase」の「」を参照してください。 AWS Tools for PowerShell

Get-ASACase

次の例は、Get-ASACase を使用する方法を説明しています。

のツール PowerShell

例 1: すべてのサポートケースの詳細を返します。

```
Get-ASACase
```

例 2: 指定された日時以降のすべてのサポートケースの詳細を返します。

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

例 3: 解決されたサポートケースを含め、最初の 10 件のサポートケースの詳細を返します。

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

例 4: 指定された 1 つのサポートケースの詳細を返します。

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

例 5: 指定されたサポートケースの詳細を返します。

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

例 6: 手動ページングを使用してすべてのサポートケースを返します。ケースは 20 個のバッチで取得されます。

```
$nextToken = $null
do {
  Get-ASACase -NextToken $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスDescribeCases」の「」を参照してください。 AWS Tools for PowerShell

Get-ASACommunication

次の例は、Get-ASACommunication を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたケースのすべての通信を返します。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

例 2: 指定されたケースについて、2012 年 1 月 1 日の深夜 UTC 以降のすべての通信を返します。

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

例 3: 2012 年 1 月 1 日の午前 0 時 UTC 以降の、指定されたケースのすべての通信を、手動ページングを使用して返します。通信は 20 個のバッチで取得されます。

```
$nextToken = $null
do {
  Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken
 $nextToken -MaxResult 20
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API の詳細については、「 コマンドレットリファレンスDescribeCommunications」の「」を参照してください。 AWS Tools for PowerShell

Get-ASAService

次の例は、Get-ASAService を使用する方法を説明しています。

のツール PowerShell

例 1: 使用可能なすべてのサービスコード、名前、カテゴリを返します。

```
Get-ASAService
```

例 2: 指定されたコードを持つサービスの名前とカテゴリを返します。

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

例 3: 指定されたサービスコードの名前とカテゴリを返します。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

例 4: 指定されたサービスコードの名前とカテゴリ (日本語) を返します。現在、英語 (「en」) と日本語 (「ja」) の言語コードがサポートされています。

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- API の詳細については、「 コマンドレットリファレンスDescribeServices」の「」を参照してください。 AWS Tools for PowerShell

Get-ASASeverityLevel

次の例は、Get-ASASeverityLevel を使用する方法を説明しています。

のツール PowerShell

例 1: AWS サポートケースに割り当てることができる重要度レベルのリストを返します。

```
Get-ASASeverityLevel
```

例 2: AWS サポートケースに割り当てることができる重要度レベルのリストを返します。レベルの名前は日本語で返されます。

```
Get-ASASeverityLevel -Language "ja"
```

- API の詳細については、「 コマンドレットリファレンスDescribeSeverityLevels」の「」を参照してください。 AWS Tools for PowerShell

Get-ASATrustedAdvisorCheck

次の例は、Get-ASATrustedAdvisorCheck を使用する方法を説明しています。

のツール PowerShell

例 1: Trusted Advisor チェックのコレクションを返します。英語出力には「en」、日本語出力には「ja」のいずれかを使用できる言語パラメータを指定する必要があります。

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrustedAdvisorChecks」の「」を参照してください。 AWS Tools for PowerShell

Get-ASATrustedAdvisorCheckRefreshStatus

次の例は、Get-ASATrustedAdvisorCheckRefreshStatus を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたチェックの更新リクエストの現在のステータスを返します。Request-ASA を使用して、チェックのステータス情報の更新をリクエストTrustedAdvisorCheckRefresh できます。

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrustedAdvisorCheckRefreshStatuses」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASATrustedAdvisorCheckResult`

次の例は、`Get-ASATrustedAdvisorCheckResult` を使用する方法を説明しています。

のツール PowerShell

例 1: Trusted Advisor チェックの結果を返します。利用可能な Trusted Advisor チェックのリストは、Get-ASA を使用して取得できますTrustedAdvisorChecks。出力は、チェックの全体的なステータス、チェックが最後に実行されたタイムスタンプ、および特定のチェックの一意のチェック ID です。結果を日本語で出力するには、-Language "ja" パラメータを追加します。

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrustedAdvisorCheckResult」の「」を参照してください。 AWS Tools for PowerShell

`Get-ASATrustedAdvisorCheckSummary`

次の例は、`Get-ASATrustedAdvisorCheckSummary` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された Trusted Advisor チェックの最新の概要を返します。

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

例 2: 指定された Trusted Advisor チェックの最新の概要を返します。

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- API の詳細については、「 コマンドレットリファレンスDescribeTrustedAdvisorCheckSummaries」の「」を参照してください。 AWS Tools for PowerShell

New-ASACase

次の例は、New-ASACase を使用する方法を説明しています。

のツール PowerShell

例 1: AWS サポートセンターで新しいケースを作成します。- ServiceCode および - CategoryCode パラメータの値は、Get-ASAService コマンドレットを使用して取得できます。SeverityCode パラメータの値は、Get-ASA SeverityLevel コマンドレットを使用して取得できます。IssueType パラメータ値は、「customer-service」または「technical」のいずれかです。成功すると、 AWS サポートケース番号が出力されます。デフォルトでは、ケースは英語で処理され、日本語を使用するには、-Language "ja" パラメータを追加します。-ServiceCode、-CategoryCode、-Subject、-CommunicationBody parameters は必須です。

```
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"
```

- API の詳細については、「 コマンドレットリファレンスCreateCase」の「」を参照してください。 AWS Tools for PowerShell

Request-ASATrustedAdvisorCheckRefresh

次の例は、Request-ASATrustedAdvisorCheckRefresh を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された Trusted Advisor チェックの更新をリクエストします。

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- API の詳細については、「 コマンドレットリファレンスRefreshTrustedAdvisorCheck」の「」を参照してください。 AWS Tools for PowerShell

Resolve-ASACase

次の例は、Resolve-ASACase を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたケースの初期状態と、解決のための呼び出しが完了した後の現在の状態を返します。

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- API の詳細については、「 コマンドレットリファレンスResolveCase」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Systems Manager の例 PowerShell

次のコード例は、Systems Manager AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

Add-SSMResourceTag

次の例は、Add-SSMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しいタグを使用してメンテナンスウィンドウを更新します。コマンドが成功した場合、出力はありません。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

例 2: PowerShell バージョン 2 では、New-Object を使用して各タグを作成する必要があります。コマンドが成功した場合、出力はありません。

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- API の詳細については、「 コマンドレットリファレンスAddTagsToResource」の「」を参照してください。 AWS Tools for PowerShell

Edit-SSMDocumentPermission

次の例は、Edit-SSMDocumentPermission を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントのすべてのアカウントに「共有」アクセス許可を追加します。コマンドが成功した場合、出力はありません。

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

例 2: この例では、ドキュメントの特定のアカウントに「共有」アクセス許可を追加します。コマンドが成功した場合、出力はありません。

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- API の詳細については、「 コマンドレットリファレンスModifyDocumentPermission」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMActivation

次の例は、Get-SSMActivation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントのアクティベーションに関する詳細情報を示します。

```
Get-SSMActivation
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeActivations」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAssociation

次の例は、Get-SSMAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスとドキュメントの関連付けを記述します。

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAssociation」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAssociationExecution

次の例は、Get-SSMAssociationExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された関連付け ID の実行を返します。

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAssociationExecutions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAssociationExecutionTarget

次の例は、Get-SSMAssociationExecutionTarget を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、関連付けの実行ターゲットの一部であるリソース ID とその実行ステータスを表示します。

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

出力:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

例 2: このコマンドは、コマンドドキュメントが関連付けられている昨日以降の特定のオートメーションにおける、特定の実行をチェックします。さらに、関連付けの実行が失敗したかどうかを確認し、失敗した場合は、実行のコマンド呼び出しの詳細情報とインスタンス ID が表示されます。

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.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
    }
}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAssociationExecutionTargets」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAssociationList

次の例は、Get-SSMAssociationList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスのすべての関連付けを一覧表示します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

出力:

```
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}
```

例 2: この例では、設定ドキュメントのすべての関連付けを一覧表示します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

出力:

```
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}
```

例 3: PowerShell バージョン 2 では、New-Object を使用して各フィルターを作成する必要があります。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスListAssociations」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAssociationVersionList

次の例は、Get-SSMAssociationVersionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、指定された関連付けのすべてのバージョンを取得します。

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスListAssociationVersions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAutomationExecution

次の例は、Get-SSMAutomationExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オートメーションの実行に関する詳細を表示します。

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

出力:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU
                            gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUGOAZG-xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,
```

```
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

例 2: この例では、指定されたオートメーションの実行 ID におけるステップの詳細情報を一覧表示します。

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

出力:

```
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   {}
```

- API の詳細については、「 コマンドレットリファレンスGetAutomationExecution」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAutomationExecutionList

次の例は、Get-SSMAutomationExecutionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントに関連付けられているすべてのアクティブなオートメーションの実行と、終了したオートメーションの実行を記述します。

```
Get-SSMAutomationExecutionList
```

出力:

```
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]]}
```

例 2: この例では、「成功 AutomationExecutionStatus 」以外の実行の ExecutionID 、ドキュメント、実行開始/終了タイムスタンプを表示します。

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

出力:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
--------------------                ------------
 ------------------------ ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi                    Failed
                   4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- API の詳細については、「 コマンドレットリファレンスDescribeAutomationExecutions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAutomationStepExecution

次の例は、Get-SSMAutomationStepExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オートメーションワークフローにおけるすべてのアクティブなステップの実行と、終了したステップの実行に関する情報を表示します。

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeAutomationStepExecutions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMAvailablePatch

次の例は、Get-SSMAvailablePatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、Windows Server 2012 で利用でき、MSRC 重要度が「緊急」のすべてのパッチを取得します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

出力:

```
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
...
```

例 2: PowerShell バージョン 2 では、New-Object を使用して各フィルターを作成する必要があります。

```
$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
```

例 3: この例では、過去 20 日間にリリースされ、 WindowsServer2019 年に一致する製品に適用可能なすべての更新を取得します。

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

出力:

```
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)
```

- API の詳細については、「 コマンドレットリファレンスDescribeAvailablePatches」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMCommand

次の例は、Get-SSMCommand を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リクエストされたすべてのコマンドを一覧表示します。

```
Get-SSMCommand
```

出力:

```
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            : {}
```

例 2: この例では、特定のコマンドのステータスを取得します。

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

例 3: この例では、2019-04-01T00:00:00Z より後に呼び出されたすべての SSM コマンドを取得します。

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListCommands」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMCommandInvocation

次の例は、Get-SSMCommandInvocation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、コマンドのすべての呼び出しを一覧表示します。

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

出力:

```
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        :
```

例 2: この例では、コマンド ID e1eb2e3c-ed4c-5123-45c1-234f5612345f の呼び出し CommandPlugins を一覧表示します。

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

出力:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- API の詳細については、「 コマンドレットリファレンスListCommandInvocations」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMCommandInvocationDetail

次の例は、Get-SSMCommandInvocationDetail を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスで実行されたコマンドの詳細情報を表示します。

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetCommandInvocation」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMComplianceItemList

次の例は、Get-SSMComplianceItemList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、任意のリソース ID とタイプのコンプライアンス項目リストを一覧表示し、コンプライアンスタイプを「関連付け」でフィルタリングします。

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

出力:

```
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            :
```

- API の詳細については、「 コマンドレットリファレンスListComplianceItems」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMComplianceSummaryList

次の例は、Get-SSMComplianceSummaryList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのコンプライアンスタイプの準拠リソースと非準拠リソースの集計カウントを返します。

```
Get-SSMComplianceSummaryList
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスListComplianceSummaries」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMConnectionStatus

次の例は、Get-SSMConnectionStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスが接続され、Session Manager 接続を受信する準備ができているかどうかを判断するため、インスタンスの Session Manager 接続ステータスを取得します。

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

出力:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- API の詳細については、「 コマンドレットリファレンスGetConnectionStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDefaultPatchBaseline

次の例は、Get-SSMDefaultPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、デフォルトのパッチベースラインを表示します。

```
Get-SSMDefaultPatchBaseline
```

出力:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- API の詳細については、「 コマンドレットリファレンスGetDefaultPatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDeployablePatchSnapshotForInstance

次の例は、Get-SSMDeployablePatchSnapshotForInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスで使用されるパッチベースラインの現在のスナップショットを表示します。このコマンドは、インスタンス認証情報を使用してインスタンスから実行する必要があります。この例では、インスタンス認証情報が使用されるようにするため、Credentials パラメータに **Amazon.Runtime.InstanceProfileAWSCredentials** オブジェクトを渡します。

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

出力:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
```

```
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

例 2: この例は、 全体を取得する方法を示しています SnapshotDownloadUrl。このコマンドは、インスタンス認証情報を使用してインスタンスから実行する必要があります。この例では、インスタンス認証情報を確実に使用できるように、**Amazon.Runtime.InstanceProfileAWSCredentials** オブジェクトを使用するように PowerShell セッションを設定します。

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

出力:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- API の詳細については、「 コマンドレットリファレンスGetDeployablePatchSnapshotForInstance」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDocument

次の例は、Get-SSMDocument を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントのコンテンツを返します。

```
Get-SSMDocument -Name "RunShellScript"
```

出力:

```
Content
-------
{...
```

例 2: この例では、ドキュメントの完全なコンテンツを表示します。

```
(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 }}"
         }
      }
   ]
}
```

- API の詳細については、「 コマンドレットリファレンスGetDocument」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDocumentDescription

次の例は、Get-SSMDocumentDescription を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントに関する情報を返します。

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeDocument」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDocumentList

次の例は、Get-SSMDocumentList を使用する方法を説明しています。

のツール PowerShell

例 1: アカウント内のすべての設定ドキュメントを一覧表示します。

```
Get-SSMDocumentList
```

出力:

```
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
...
```

例 2: この例では、名前が「プラットフォーム」と一致するすべてのオートメーションドキュメントを取得します。

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

出力:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- API の詳細については、「 コマンドレットリファレンスListDocuments」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDocumentPermission

次の例は、Get-SSMDocumentPermission を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントのすべてのバージョンを一覧表示します。

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

出力:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- API の詳細については、「 コマンドレットリファレンスDescribeDocumentPermission」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMDocumentVersionList

次の例は、Get-SSMDocumentVersionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントのアクセス許可リストを返します。

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

出力:

```
all
```

- API の詳細については、「 コマンドレットリファレンスListDocumentVersions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMEffectiveInstanceAssociationList

次の例は、Get-SSMEffectiveInstanceAssociationList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスの有効な関連付けを記述します。

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

出力:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

例 2: この例では、インスタンスの有効な関連付けの内容を記述します。

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

出力:

```
{
    "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 }}"
                }
            ]
        }
    }
}
```

- API の詳細については、「 コマンドレットリファレンスDescribeEffectiveInstanceAssociations」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMEffectivePatchesForPatchBaseline

次の例は、Get-SSMEffectivePatchesForPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのパッチベースラインを一覧表示します。結果の最大表示件数は 1 です。

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

出力:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

例 2: この例では、すべてのパッチベースラインのパッチステータスを表示します。結果の最大表示件数は 1 です。

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

出力:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- API の詳細については、「 コマンドレットリファレンスDescribeEffectivePatchesForPatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInstanceAssociationsStatus

次の例は、Get-SSMInstanceAssociationsStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスの関連付けの詳細情報を表示します。

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

出力:

```
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
```

例 2: この例では、指定されたインスタンス ID におけるインスタンスの関連付けのステータスを確認し、それらの関連付けの実行ステータスを表示します。

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstanceAssociationsStatus」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInstanceInformation

次の例は、Get-SSMInstanceInformation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、各インスタンスの詳細情報を表示します。

```
Get-SSMInstanceInformation
```

出力:

```
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
```

例 2: この例では、 -Filter パラメータを使用して、 **AgentVersion**が **us-east-1**のリージョンにある AWS Systems Manager インスタンスのみに結果をフィルタリングする方法を示します**2.2.800.0**。API InstanceInformation リファレンストピック (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type-InstanceInformation-ActivationId) に有効な -Filter キー値のリストがあります。

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

出力:

```
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
```

例 3: この例では、 InstanceInformationFilterList パラメータを使用して、 **Windows**または **us-east-1PlatformTypes**のリージョンにある AWS Systems Manager インスタンスのみに結果をフィルタリングする方法を示します**Linux**。有効な InstanceInformationFilterList キー値のリストは、 InstanceInformationFilter API リファレンストピック (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html) にあります。

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

出力:

```
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
```

例 4: この例では、ssm マネージドインスタンスを一覧表示し InstanceId、、 PingStatus、LastPingDateTime PlatformName を csv ファイルにエクスポートします。

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstanceInformation」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInstancePatch

次の例は、Get-SSMInstancePatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスのパッチコンプライアンスの詳細を取得します。

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstancePatches」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInstancePatchState

次の例は、Get-SSMInstancePatchState を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスのパッチの概要状態を取得します。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

例 2: この例では、2 つのインスタンスにおけるパッチの概要状態を取得します。

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstancePatchStates」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInstancePatchStatesForPatchGroup

次の例は、Get-SSMInstancePatchStatesForPatchGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループにおけるインスタンスごとのパッチの概要状態を取得します。

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- API の詳細については、「 コマンドレットリファレンスDescribeInstancePatchStatesForPatchGroup」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInventory

次の例は、Get-SSMInventory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インベントリのカスタムメタデータを取得します。

```
Get-SSMInventory
```

出力:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- API の詳細については、「 コマンドレットリファレンスGetInventory」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInventoryEntriesList

次の例は、Get-SSMInventoryEntriesList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスのすべてのカスタムインベントリエントリを一覧表示します。

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

出力:

```
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
```

例 2: この例では詳細を一覧表示します。

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

出力:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- API の詳細については、「 コマンドレットリファレンスListInventoryEntries」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMInventoryEntryList

次の例は、Get-SSMInventoryEntryList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスの**AWS:Network**型インベントリエントリを取得します。

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンス」の「Get-SSMInventoryEntryList」を参照してください。 AWS Tools for PowerShell

Get-SSMInventorySchema

次の例は、Get-SSMInventorySchema を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントのインベントリタイプ名のリストを返します。

```
Get-SSMInventorySchema
```

- API の詳細については、「 コマンドレットリファレンスGetInventorySchema」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMLatestEC2Image

次の例は、Get-SSMLatestEC2Image を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、最新のすべての Windows AMIs。

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

出力:

```
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
...
```

例 2: この例では、us-west-2 リージョンの特定の Amazon Linux イメージの AMI ID を取得します。

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

出力:

```
ami-09b92cd132204c704
```

例 3: この例では、指定されたワイルドカード式に一致する最新の Windows AMIs をすべて一覧表示します。

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

出力:

```
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
...
```

- API の詳細については、「 コマンドレットリファレンスGet-SSMLatestEC2Image」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindow

次の例は、Get-SSMMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの詳細を取得します。

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowExecution

次の例は、Get-SSMMaintenanceWindowExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの実行の一部として実行されるタスクに関する情報を一覧表示します。

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスGetMaintenanceWindowExecution」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowExecutionList

次の例は、Get-SSMMaintenanceWindowExecutionList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウにおけるすべての実行を一覧表示します。

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

出力:

```
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
```

例 2: この例では、指定されたメンテナンスウィンドウにおける指定された日付より前のすべての実行を一覧表示します。

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

例 3: この例では、指定されたメンテナンスウィンドウにおける指定された日付より後のすべての実行を一覧表示します。

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindowExecutions」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowExecutionTask

次の例は、Get-SSMMaintenanceWindowExecutionTask を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの実行の一部であったタスクに関する情報を一覧表示します。

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

出力:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsMana
                   meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- API の詳細については、「 コマンドレットリファレンスGetMaintenanceWindowExecutionTask」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowExecutionTaskInvocationList

次の例は、Get-SSMMaintenanceWindowExecutionTaskInvocationList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの実行の一部として実行される、タスクの呼び出しを一覧表示します。

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

出力:

```
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    :
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindowExecutionTaskInvocations」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowExecutionTaskList

次の例は、Get-SSMMaintenanceWindowExecutionTaskList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの実行に関連するタスクを一覧表示します。

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindowExecutionTasks」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowList

次の例は、Get-SSMMaintenanceWindowList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントにおけるすべてのメンテナンスウィンドウを一覧表示します。

```
Get-SSMMaintenanceWindowList
```

出力:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindows」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowTarget

次の例は、Get-SSMMaintenanceWindowTarget を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウのすべてのターゲットを一覧表示します。

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindowTargets」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMMaintenanceWindowTaskList

次の例は、Get-SSMMaintenanceWindowTaskList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウのすべてのタスクを一覧表示します。

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeMaintenanceWindowTasks」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMParameterHistory

次の例は、Get-SSMParameterHistory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パラメータの値の履歴を一覧表示します。

```
Get-SSMParameterHistory -Name "Welcome"
```

出力:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- API の詳細については、「 コマンドレットリファレンスGetParameterHistory」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMParameterList

次の例は、Get-SSMParameterList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのパラメータを一覧表示します。

```
Get-SSMParameterList
```

出力:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- API の詳細については、「 コマンドレットリファレンスDescribeParameters」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMParameterValue

次の例は、Get-SSMParameterValue を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パラメータの値を一覧表示します。

```
Get-SSMParameterValue -Name "Welcome"
```

出力:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

例 2: この例では、値の詳細を一覧表示します。

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

出力:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- API の詳細については、「 コマンドレットリファレンスGetParameters」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMPatchBaseline

次の例は、Get-SSMPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、すべてのパッチベースラインを一覧表示します。

```
Get-SSMPatchBaseline
```

出力:

```
BaselineDescription                                                 BaselineId
                                                  BaselineName
------------------                                                  ----------
                                                  ------------
Default Patch Baseline Provided by AWS.                             arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
                                                  Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
                                                  Production-B...
```

例 2: この例では、 によって提供されるすべてのパッチベースラインを一覧表示します AWS。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

出力:

```
Get-SSMPatchBaseline -Filter $filter1
```

例 3: この例では、所有者しているすべてのパッチベースラインを一覧表示します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

出力:

```
Get-SSMPatchBaseline -Filter $filter1
```

例 4: PowerShell バージョン 2 では、New-Object を使用して各タグを作成する必要があります。

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"
```

```
Get-SSMPatchBaseline -Filter $filter1
```

出力:

```
BaselineDescription                     BaselineId
                          BaselineName              DefaultBaselin

                                                    e
------------------                      ----------
                          ------------              --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- API の詳細については、「 コマンドレットリファレンスDescribePatchBaselines」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMPatchBaselineDetail

次の例は、Get-SSMPatchBaselineDetail を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチベースラインの詳細を表示します。

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

出力:

```
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 : {}
```

- API の詳細については、「 コマンドレットリファレンスGetPatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMPatchBaselineForPatchGroup

次の例は、Get-SSMPatchBaselineForPatchGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループのパッチベースラインを表示します。

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

出力:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API の詳細については、「 コマンドレットリファレンスGetPatchBaselineForPatchGroup」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMPatchGroup

次の例は、Get-SSMPatchGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループの登録を一覧表示します。

```
Get-SSMPatchGroup
```

出力:

```
BaselineIdentity                                                PatchGroup
----------------                                                ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- API の詳細については、「 コマンドレットリファレンスDescribePatchGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMPatchGroupState

次の例は、Get-SSMPatchGroupState を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループのパッチコンプライアンスの概要を取得します。

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

出力:

```
Instances                           : 4
InstancesWithFailedPatches          : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches       : 3
InstancesWithMissingPatches         : 0
InstancesWithNotApplicablePatches   : 0
```

- API の詳細については、「 コマンドレットリファレンスDescribePatchGroupState」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMResourceComplianceSummaryList

次の例は、Get-SSMResourceComplianceSummaryList を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、リソースレベルの概要数を取得します。概要には、「Windows10」に一致する製品の準拠ステータスと非準拠ステータス、および詳細なコンプライアンス項目の重要度数に関する情報が含まれます。パラメータが指定されておらず、この値が有効でない場合、MaxResult デフォルトは 100 であるため、 MaxResult パラメータが追加され、値は 50 に設定されます。

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- API の詳細については、「 コマンドレットリファレンスListResourceComplianceSummaries」の「」を参照してください。 AWS Tools for PowerShell

Get-SSMResourceTag

次の例は、Get-SSMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウのタグを一覧表示します。

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

出力:

```
Key   Value
---   -----
Stack Production
```

- API の詳細については、「 コマンドレットリファレンスListTagsForResource」の「」を参照してください。 AWS Tools for PowerShell

New-SSMActivation

次の例は、New-SSMActivation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、マネージドインスタンスを作成します。

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

出力:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- API の詳細については、「 コマンドレットリファレンスCreateActivation」の「」を参照してください。 AWS Tools for PowerShell

New-SSMAssociation

次の例は、New-SSMAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンス ID を使用して、設定ドキュメントをインスタンスに関連付けます。

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

出力:

```
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 :
```

例 2: この例では、ターゲットを使用して、設定ドキュメントをインスタンスに関連付けます。

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

出力:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

例 3: この例では、ターゲットとパラメータを使用して、設定ドキュメントをインスタンスに関連付けます。

```
$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
```

出力:

```
Name                 : Configure-CloudWatch
InstanceId           :
Date                 : 5/17/2018 3:17:44 PM
Status.Name          :
Status.Date          :
Status.Message       :
Status.AdditionalInfo :
```

例 4: この例では、**AWS-GatherSoftwareInventory** を使用して、リージョン内におけるすべてのインスタンスとの関連付けを作成します。また、収集するパラメータにカスタムファイルとレジストリの場所を指定します。

```
$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)"
```

出力:

```
Name                 : AWS-GatherSoftwareInventory
InstanceId           :
Date                 : 6/9/2019 8:57:56 AM
Status.Name          :
```

```
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API の詳細については、「 コマンドレットリファレンスCreateAssociation」の「」を参照してください。 AWS Tools for PowerShell

New-SSMAssociationFromBatch

次の例は、New-SSMAssociationFromBatch を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、設定ドキュメントを複数のインスタンスに関連付けます。出力では、成功したオペレーションと失敗したオペレーションのリストが返されます (該当する場合)。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

出力:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

例 2: この例では、成功したオペレーションの詳細を表示します。

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- API の詳細については、「 コマンドレットリファレンスCreateAssociationBatch」の「」を参照してください。 AWS Tools for PowerShell

New-SSMDocument

次の例は、New-SSMDocument を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アカウントにドキュメントを作成します。ドキュメントは JSON 形式である必要があります。設定ドキュメントの記述については、「SSM API Reference」の「Configuration Document」を参照してください。

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateDocument」の「」を参照してください。 AWS Tools for PowerShell

New-SSMMaintenanceWindow

次の例は、New-SSMMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、そのウィンドウは、毎週火曜日の午後 4 時に 4 時間実行され (カットオフは 1 時間)、関連付けられていないターゲットを許可する、指定された名前の新しいメンテナンスウィンドウを作成します。

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

出力:

```
mw-03eb53e1ea7383998
```

- API の詳細については、「 コマンドレットリファレンスCreateMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

New-SSMPatchBaseline

次の例は、New-SSMPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、本番環境で Windows Server 2019 を実行しているマネージドインスタンスに対して、Microsoft からリリースされてから 7 日後にパッチを承認するパッチベースラインを作成します。

```
$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
```

出力:

```
pb-0z4z6221c4296b23z
```

- API の詳細については、「 コマンドレットリファレンスCreatePatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Register-SSMDefaultPatchBaseline

次の例は、Register-SSMDefaultPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチベースラインをデフォルトのパッチベースラインとして登録します。

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

出力:

```
pb-03da896ca3b68b639
```

- API の詳細については、「 コマンドレットリファレンスRegisterDefaultPatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Register-SSMPatchBaselineForPatchGroup

次の例は、Register-SSMPatchBaselineForPatchGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループのパッチベースラインを登録します。

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

出力:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- API の詳細については、「 コマンドレットリファレンスRegisterPatchBaselineForPatchGroup」の「」を参照してください。 AWS Tools for PowerShell

`Register-SSMTargetWithMaintenanceWindow`

次の例は、Register-SSMTargetWithMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスをメンテナンスウィンドウに登録します。

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

出力:

```
d8e47760-23ed-46a5-9f28-927337725398
```

例 2: この例では、複数のインスタンスをメンテナンスウィンドウに登録します。

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

出力:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

例 3: この例では、EC2 タグを使用して、インスタンスをメンテナンスウィンドウに登録します。

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

出力:

```
2994977e-aefb-4a71-beac-df620352f184
```

- API の詳細については、「 コマンドレットリファレンスRegisterTargetWithMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Register-SSMTaskWithMaintenanceWindow

次の例は、Register-SSMTaskWithMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンス ID を使用して、タスクをメンテナンスウィンドウに登録します。出力はタスク ID です。

```
$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
```

出力:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

例 2: この例では、ターゲット ID を使用して、タスクをメンテナンスウィンドウに登録します。出力はタスク ID です。

```
$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
```

出力:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

例 3: この例では、run コマンドドキュメント **AWS-RunPowerShellScript** のパラメータオブジェクトを作成し、ターゲット ID を使用して任意のメンテナンスウィンドウを持つタスクを作成します。返される出力はタスク ID です。

```
$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
```

出力:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

例 4: この例では、 という名前のドキュメントを使用して AWS Systems Manager Automation タスクを登録します**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"
```

- API の詳細については、「 コマンドレットリファレンスRegisterTaskWithMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMActivation

次の例は、Remove-SSMActivation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、アクティベーションを削除します。コマンドが成功した場合、出力はありません。

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- API の詳細については、「 コマンドレットリファレンスDeleteActivation」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMAssociation

次の例は、Remove-SSMAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスとドキュメント間の関連付けを削除します。コマンドが成功した場合、出力はありません。

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- API の詳細については、「 コマンドレットリファレンスDeleteAssociation」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMDocument

次の例は、Remove-SSMDocument を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ドキュメントを削除します。コマンドが成功した場合、出力はありません。

```
Remove-SSMDocument -Name "RunShellScript"
```

- API の詳細については、「 コマンドレットリファレンスDeleteDocument」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMMaintenanceWindow

次の例は、Remove-SSMMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウを削除します。

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

出力:

```
mw-06d59c1a07c022145
```

- API の詳細については、「 コマンドレットリファレンスDeleteMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMParameter

次の例は、Remove-SSMParameter を使用する方法を説明しています。

のツール PowerShell

例 1: この例ではパラメータを削除します。コマンドが成功した場合、出力はありません。

```
Remove-SSMParameter -Name "helloWorld"
```

- API の詳細については、「 コマンドレットリファレンスDeleteParameter」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMPatchBaseline

次の例は、Remove-SSMPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチベースラインを削除します。

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

出力:

```
pb-045f10b4f382baeda
```

- API の詳細については、「 コマンドレットリファレンスDeletePatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

Remove-SSMResourceTag

次の例は、Remove-SSMResourceTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウからタグを削除します。コマンドが成功した場合、出力はありません。

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- API の詳細については、「 コマンドレットリファレンスRemoveTagsFromResource」の「」を参照してください。 AWS Tools for PowerShell

Send-SSMCommand

次の例は、Send-SSMCommand を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ターゲットインスタンスで echo コマンドを実行します。

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

出力:

```
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}
```

例 2: この例は、ネストされたパラメータを受け入れるコマンドを実行する方法を示しています。

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- API の詳細については、「 コマンドレットリファレンスSendCommand」の「」を参照してください。 AWS Tools for PowerShell

Start-SSMAutomationExecution

次の例は、Start-SSMAutomationExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オートメーションロール、AMI ソース ID、および Amazon EC2 インスタンスロールを指定するドキュメントを実行します。

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

出力:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- API の詳細については、「 コマンドレットリファレンスStartAutomationExecution」の「」を参照してください。 AWS Tools for PowerShell

Stop-SSMAutomationExecution

次の例は、Stop-SSMAutomationExecution を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、オートメーションの実行を停止します。コマンドが成功した場合、出力はありません。

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- API の詳細については、「 コマンドレットリファレンスStopAutomationExecution」の「」を参照してください。 AWS Tools for PowerShell

Stop-SSMCommand

次の例は、Stop-SSMCommand を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、コマンドのキャンセルを試みます。オペレーションが成功した場合、出力はありません。

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- API の詳細については、「 コマンドレットリファレンスCancelCommand」の「」を参照してください。 AWS Tools for PowerShell

Unregister-SSMManagedInstance

次の例は、Unregister-SSMManagedInstance を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、マネージドインスタンスを登録解除します。コマンドが成功した場合、出力はありません。

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- API の詳細については、「 コマンドレットリファレンスDeregisterManagedInstance」の「」を参照してください。 AWS Tools for PowerShell

Unregister-SSMPatchBaselineForPatchGroup

次の例は、Unregister-SSMPatchBaselineForPatchGroup を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、パッチグループをパッチベースラインから登録解除します。

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

出力:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API の詳細については、「 コマンドレットリファレンスDeregisterPatchBaselineForPatchGroup」の「」を参照してください。 AWS Tools for PowerShell

Unregister-SSMTargetFromMaintenanceWindow

次の例は、Unregister-SSMTargetFromMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウからターゲットを削除します。

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

出力:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- API の詳細については、「 コマンドレットリファレンスDeregisterTargetFromMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Unregister-SSMTaskFromMaintenanceWindow

次の例は、Unregister-SSMTaskFromMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウからタスクを削除します。

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

出力:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- API の詳細については、「 コマンドレットリファレンスDeregisterTaskFromMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMAssociation

次の例は、Update-SSMAssociation を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、新しいドキュメントバージョンとの関連付けを更新します。

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

出力:

```
Name                  : AWS-UpdateSSMAgent
```

```
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API の詳細については、「 コマンドレットリファレンスUpdateAssociation」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMAssociationStatus

次の例は、Update-SSMAssociationStatus を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、インスタンスと設定ドキュメント間の関連付けのステータスを更新します。

```
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"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateAssociationStatus」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMDocument

次の例は、Update-SSMDocument を使用する方法を説明しています。

のツール PowerShell

例 1: これは、指定した JSON ファイルの更新内容を含むドキュメントの新しいバージョンを作成するためのものです。ドキュメントは JSON 形式である必要があります。ドキュメントバージョンは、「Get-SSMDocumentVersionList」コマンドレットで取得できます。

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスUpdateDocument」の「」を参照してください。 AWS Tools for PowerShell

`Update-SSMDocumentDefaultVersion`

次の例は、`Update-SSMDocumentDefaultVersion` を使用する方法を説明しています。

のツール PowerShell

例 1: ここではドキュメントのデフォルトバージョンを更新します。「Get-SSMDocumentVersionList」コマンドレットを使用して、使用可能なドキュメントバージョンを取得できます。

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

出力:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- API の詳細については、「 コマンドレットリファレンスUpdateDocumentDefaultVersion」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMMaintenanceWindow

次の例は、Update-SSMMaintenanceWindow を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、メンテナンスウィンドウの名前を更新します。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

出力:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

例 2: この例では、メンテナンスウィンドウを有効にします。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

出力:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
```

```
Name                        : My-Renamed-MW
Schedule                    : cron(0 */30 * * * ? *)
WindowId                    : mw-03eb9db42890fb82d
```

例 3: この例では、メンテナンスウィンドウを無効にします。

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

出力:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- API の詳細については、「 コマンドレットリファレンスUpdateMaintenanceWindow」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMManagedInstanceRole

次の例は、Update-SSMManagedInstanceRole を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、マネージドインスタンスのロールを更新します。コマンドが成功した場合、出力はありません。

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- API の詳細については、「 コマンドレットリファレンスUpdateManagedInstanceRole」の「」を参照してください。 AWS Tools for PowerShell

Update-SSMPatchBaseline

次の例は、Update-SSMPatchBaseline を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、2 つのパッチを拒否済み、1 つのパッチを承認済みとして既存のパッチベースラインに追加します。

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

出力:

```
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}
```

- API の詳細については、「 コマンドレットリファレンスUpdatePatchBaseline」の「」を参照してください。 AWS Tools for PowerShell

`Write-SSMComplianceItem`

次の例は、`Write-SSMComplianceItem` を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、任意のマネージドインスタンスのカスタムコンプライアンス項目を書き込みます。

```
$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"
```

- API の詳細については、「 コマンドレットリファレンスPutComplianceItems」の「」を参照してください。 AWS Tools for PowerShell

Write-SSMInventory

次の例は、Write-SSMInventory を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、ラックの場所情報をインスタンスに割り当てます。コマンドが成功した場合、出力はありません。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスPutInventory」の「」を参照してください。 AWS Tools for PowerShell

Write-SSMParameter

次の例は、Write-SSMParameter を使用する方法を説明しています。

のツール PowerShell

例 1: この例ではパラメータを作成します。コマンドが成功した場合、出力はありません。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

例 2: この例ではパラメータを変更します。コマンドが成功した場合、出力はありません。

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- API の詳細については、「 コマンドレットリファレンスPutParameter」の「」を参照してください。 AWS Tools for PowerShell

Tools for を使用した Amazon Translate の例 PowerShell

次のコード例は、Amazon Translate AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

`ConvertTo-TRNTargetLanguage`

次の例は、`ConvertTo-TRNTargetLanguage` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された英語テキストをフランス語に変換します。変換するテキストは、-Text パラメータとして渡すこともできます。

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- API の詳細については、「 コマンドレットリファレンスTranslateText」の「」を参照してください。 AWS Tools for PowerShell

AWS WAFV2 Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS WAFV2。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

New-WAF2WebACL

次の例は、New-WAF2WebACL を使用する方法を説明しています。

のツール PowerShell

例 1: このコマンドは、「waf-test」という名前の新しいウェブ ACL を作成します。サービス API ドキュメントでは、DefaultAction「」は必須プロパティです。したがって、'-DefaultAction_Allow' および/または '-DefaultAction_Block' の値を指定する必要があります。'-DefaultAction_Allow' と '-DefaultAction_Block' は必須プロパティではないため、上記の例に示すように、値 '@{}' をプレースホルダーとして使用できます。

```
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 @{}
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスCreateWebAcl」の「」を参照してください。 AWS Tools for PowerShell

WorkSpaces Tools for を使用した の例 PowerShell

次のコード例は、 AWS Tools for PowerShell で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています WorkSpaces。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

各例には、 へのリンクが含まれています。ここでは GitHub、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- アクション

アクション

Approve-WKSIpRule

次の例は、Approve-WKSIpRule を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、既存の IP グループにルールを追加します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスAuthorizeIpRules」の「」を参照してください。 AWS Tools for PowerShell

Copy-WKSWorkspaceImage

次の例は、Copy-WKSWorkspaceImage を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、「」という名前の us-west-2 から現在のリージョンに、指定された ID のワークスペースイメージをコピーします。CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

出力:

```
wsi-456abaqfe
```

- API の詳細については、「 コマンドレットリファレンスCopyWorkspaceImage」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSClientProperty

次の例は、Edit-WKSClientProperty を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、Workspaces クライアントの再接続を有効にします

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- API の詳細については、「 コマンドレットリファレンスModifyClientProperties」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSSelfServicePermission

次の例は、Edit-WKSSelfServicePermission を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、セルフサービスのアクセス許可を有効にして、指定されたディレクトリのコンピューティングタイプを変更し、ボリュームサイズを増やします。

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- API の詳細については、「 コマンドレットリファレンスModifySelfservicePermissions」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSWorkspaceAccessProperty

次の例は、Edit-WKSWorkspaceAccessProperty を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、指定したディレクトリの Android および Chrome OS で Workspace アクセスを有効にします。

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- API の詳細については、「 コマンドレットリファレンスModifyWorkspaceAccessProperties」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSWorkspaceCreationProperty

次の例は、Edit-WKSWorkspaceCreationProperty を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、Workspace の作成中にインターネットアクセスとメンテナンスモードをデフォルト値として true にすることができます。

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- API の詳細については、「 コマンドレットリファレンスModifyWorkspaceCreationProperties」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSWorkspaceProperty

次の例は、Edit-WKSWorkspaceProperty を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、指定した Workspace の Workspace 実行モードプロパティを自動停止に変更します。

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- API の詳細については、「 コマンドレットリファレンスModifyWorkspaceProperties」の「」を参照してください。 AWS Tools for PowerShell

Edit-WKSWorkspaceState

次の例は、Edit-WKSWorkspaceState を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された Workspace の状態を Available に変更します。

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- API の詳細については、「 コマンドレットリファレンスModifyWorkspaceState」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSClientProperty

次の例は、Get-WKSClientProperty を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定されたディレクトリの Workspace クライアントのクライアントプロパティを取得します。

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- API の詳細については、「 コマンドレットリファレンスDescribeClientProperties」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSIpGroup

次の例は、Get-WKSIpGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定されたリージョン内の指定された IP グループの詳細を取得します。

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

出力:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- API の詳細については、「 コマンドレットリファレンスDescribeIpGroups」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSTag

次の例は、Get-WKSTag を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された Workspace のタグを取得します

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

出力:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- API の詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspace

次の例は、Get-WKSWorkspace を使用する方法を説明しています。

のツール PowerShell

例 1: すべての の詳細をパイプライン WorkSpaces に取得します。

```
Get-WKSWorkspace
```

出力:

```
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
```

例 2: このコマンドは、 **us-west-2** リージョンのワークスペース**WorkSpaceProperties**の の子プロパティの値を表示します。の子プロパティの詳細について は**WorkSpaceProperties**、https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html を参照してください。

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

出力:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

例 3: このコマンドは、 **us-west-2** リージョンのワークスペース**WorkSpaceProperties**の **RootVolumeSizeGib**の子プロパティの値を表示します。GiB 単位のルートボリュームサイズは 80 です。

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

出力:

```
        80
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspaceBundle

次の例は、Get-WKSWorkspaceBundle を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、現在のリージョンのすべての Workspace バンドルの詳細を取得します。

```
Get-WKSWorkspaceBundle
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspaceBundles」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspaceDirectory

次の例は、Get-WKSWorkspaceDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、登録されたディレクトリのディレクトリの詳細を一覧表示します。

```
Get-WKSWorkspaceDirectory
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspaceDirectories」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspaceImage

次の例は、Get-WKSWorkspaceImage を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、リージョン内のすべてのイメージの詳細をすべて取得します。

```
Get-WKSWorkspaceImage
```

出力:

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspaceImages」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspaceSnapshot

次の例は、Get-WKSWorkspaceSnapshot を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された Workspace 用に作成された最新のスナップショットのタイムスタンプを示しています。

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

出力:

```
RebuildSnapshots                    RestoreSnapshots
----------------                    ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspaceSnapshots」の「」を参照してください。 AWS Tools for PowerShell

Get-WKSWorkspacesConnectionStatus

次の例は、Get-WKSWorkspacesConnectionStatus を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された Workspace の接続ステータスを取得します。

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- API の詳細については、「 コマンドレットリファレンスDescribeWorkspacesConnectionStatus」の「」を参照してください。 AWS Tools for PowerShell

New-WKSIpGroup

次の例は、New-WKSIpGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、 という名前の空の Ip グループを作成します。 FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

出力:

```
wsipg-w45rty4ty
```

- API の詳細については、「 コマンドレットリファレンスCreateIpGroup」の「」を参照してください。 AWS Tools for PowerShell

New-WKSTag

次の例は、New-WKSTag を使用する方法を説明しています。

のツール PowerShell

例 1: この例では、 という名前のワークスペースに新しいタグを追加しますws-wsname。タグのキーは「Name」で、キー値は です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
```

例 2: この例では、 という名前のワークスペースに複数のタグを追加しますws-wsname。1 つのタグには「Name」のキーと のキー値AWS_Workspaceがあり、もう 1 つのタグには「Stage」のタグキーと「Test」のキー値があります。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスCreateTags」の「」を参照してください。 AWS Tools for PowerShell

New-WKSWorkspace

次の例は、New-WKSWorkspace を使用する方法を説明しています。

のツール PowerShell

例 1: 指定されたバンドル、ディレクトリ、およびユーザーの WorkSpace を作成します。

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

例 2: この例では、複数の を作成します。 WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- API の詳細については、「 コマンドレットリファレンスCreateWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

Register-WKSIpGroup

次の例は、Register-WKSIpGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された IP グループを指定されたディレクトリに登録します

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- API の詳細については、「 コマンドレットリファレンスAssociateIpGroups」の「」を参照してください。 AWS Tools for PowerShell

Register-WKSWorkspaceDirectory

次の例は、Register-WKSWorkspaceDirectory を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、Workspaces Service に指定されたディレクトリを登録します。

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- API の詳細については、「 コマンドレットリファレンスRegisterWorkspaceDirectory」の「」を参照してください。 AWS Tools for PowerShell

Remove-WKSIpGroup

次の例は、Remove-WKSIpGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定された IP グループを削除します

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteIpGroup」の「」を参照してください。 AWS Tools for PowerShell

Remove-WKSTag

次の例は、Remove-WKSTag を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルでは、Workspace に関連付けられたタグを削除します。

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

出力:

```
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
```

- API の詳細については、「 コマンドレットリファレンスDeleteTags」の「」を参照してください。 AWS Tools for PowerShell

Remove-WKSWorkspace

次の例は、Remove-WKSWorkspace を使用する方法を説明しています。

のツール PowerShell

例 1: 複数の を終了します WorkSpaces。-Force スイッチを使用すると、コマンドレットが確認を求めるプロンプトが表示されなくなります。

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

例 2: すべての のコレクションを取得し WorkSpaces 、IDs を Remove-WKSWorkspace の WorkSpaceId パラメータにパイプし、すべての を終了します WorkSpaces。コマンドレット WorkSpace は、それぞれが終了する前にプロンプトを表示します。確認プロンプトを抑制するには、-Force スイッチを追加します。

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

例 3: この例では、終了する を定義する TerminateRequest オブジェクト WorkSpaces を渡す方法を示します。コマンドレットは、-Force スイッチパラメータも指定されていない限り、先に進む前に確認を求められます。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスTerminateWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

Reset-WKSWorkspace

次の例は、Reset-WKSWorkspace を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された を再構築します WorkSpace。

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

例 2: すべての のコレクションを取得し WorkSpaces 、IDs を Reset-WKSWorkspace の WorkSpaceId パラメータ WorkSpaces にパイプして、 を再構築します。

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- API の詳細については、「 コマンドレットリファレンスRebuildWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

`Restart-WKSWorkspace`

次の例は、`Restart-WKSWorkspace` を使用する方法を説明しています。

のツール PowerShell

例 1: 指定された を再起動します WorkSpace。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

例 2: 複数の を再起動します WorkSpaces。

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

例 3: すべての のコレクションを取得し WorkSpaces 、IDs を Restart-WKSWorkspace の WorkSpaceId パラメータ WorkSpaces にパイプして、 を再起動します。

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- API の詳細については、「 コマンドレットリファレンスRebootWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

`Stop-WKSWorkspace`

次の例は、`Stop-WKSWorkspace` を使用する方法を説明しています。

のツール PowerShell

例 1: 複数の を停止します WorkSpaces。

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

例 2: すべての のコレクションを取得し WorkSpaces 、 IDs を Stop-WKSWorkspace の WorkSpaceId パラメータ WorkSpaces にパイプして、 を停止させます。

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

例 3: この例では、停止する を定義する StopRequest オブジェクト WorkSpaces を渡す方法を示します。

```
$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
```

- API の詳細については、「 コマンドレットリファレンスStopWorkspaces」の「」を参照してください。 AWS Tools for PowerShell

Unregister-WKSIpGroup

次の例は、Unregister-WKSIpGroup を使用する方法を説明しています。

のツール PowerShell

例 1: このサンプルは、指定されたディレクトリから指定された IP グループを登録解除します。

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- API の詳細については、「 コマンドレットリファレンスDisassociateIpGroups」の「」を参照してください。 AWS Tools for PowerShell

この AWS 製品またはサービスのセキュリティ

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。 AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、 AWS とユーザー間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ — AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャを保護し AWS 、安全に使用できるサービスを提供します。当社のセキュリティ責任は において最優先事項であり AWS、当社のセキュリティの有効性は、 AWS コンプライアンスプログラム の一環として、サードパーティーの監査人によって定期的にテストおよび検証されています。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、およびお客様のデータの機密性、組織の要件、適用可能な法律や規制などのその他の要因によって決まります。

この AWS 製品またはサービスは、サポートしている特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となる のサービスを参照してください。

トピック

- この AWS 製品またはサービスのデータ保護
- ID とアクセス管理
- この AWS 製品またはサービスのコンプライアンス検証
- Tools for PowerShell で最小 TLS バージョンを適用する方法
- Tools for のセキュリティに関するその他の考慮事項 PowerShell

この AWS 製品またはサービスのデータ保護

AWS 責任共有モデル は、このAWSの製品またはサービスのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護するがあります。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバ

シーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、「AWS セキュリティブログ」に投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみを各ユーザーに付与できます。また、次の方法でデータを保護することをおすすめします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須です。TLS 1.3 が推奨されます。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。
- AWS のサービス内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API により AWS にアクセスするときに FIPS 140-2 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

顧客の E メールアドレスなどの機密情報や重要情報は、タグや Name フィールドなどの自由形式のフィールドに入力しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK を使用して、この AWS 製品またはサービス、あるいはその他の AWS のサービスを使用する場合も同様です。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

データの暗号化

セキュリティで保護されたサービスの重要な特徴として、情報はアクティブに使用されていないときに暗号化されます。

保管時の暗号化

AWS Tools for PowerShell は、ユーザーに代わって AWS のサービスとやり取りするために必要な認証情報を除き、それ自体にカスタマーデータを保存することはありません。

AWS Tools for PowerShell を使用して、カスタマーデータをローカルコンピュータに転送して保存する AWS のサービスを呼び出す場合は、そのサービスのユーザーガイドの「セキュリティ & コンプライアンス」の章で、データの保存、保護、および暗号化の方法を参照してください。

転送時の暗号化

デフォルトでは、AWS Tools for PowerShell や AWS のサービスのエンドポイントを実行しているクライアントコンピュータから転送されるすべてのデータは、HTTPS/TLS 接続を介した送信により、すべてが暗号化されます。

HTTPS/TLS の使用を有効にするために必要な操作はありません。常に有効になっています。

ID とアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰をわ認証 (サインイン) し、誰に AWS リソースの使用を許可する (権限を持たせる) かを制御します。IAM は、無料で使用できる AWS のサービスです。

トピック

- 対象者
- アイデンティティによる認証
- ポリシーを使用したアクセス権の管理
- AWS のサービスと IAM の連携の仕組み
- AWS ID とアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の用途は、AWS で行う作業によって異なります。

サービスユーザー – ジョブを実行するために AWS のサービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。作業を実行するためにさらに多くの AWS 機能を使用するとき、追加の権限が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。AWS の機能にアクセスできない場合は、「AWS ID とアクセスのトラブルシューティング」を参照するか、使用している AWS のサービスのユーザーガイドを参照してください。

サービス管理者 - 社内の AWS リソースを担当している場合は、通常、AWS への完全なアクセスがあります。サービスのユーザーがどの AWS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。AWS で IAM を利用する方法の詳細については、使用している AWS のサービスのユーザーガイドを参照してください。

IAM 管理者 - 管理者は、AWS へのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS ID ベースのポリシーの例を表示するには、使用している AWS のサービスのユーザーガイドを参照してください。。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。

ID ソースから提供された認証情報を使用して、フェデレーティッドアイデンティティとして AWS にサインインできます。 AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「AWS アカウント にサインインする方法」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では、アカウントのセキュリティ強化のために多要素認証 (MFA) の使用をお勧めしています。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、そのアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッド ID

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスにアクセスすることを要求します。

フェデレーテッドアイデンティティ は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS のサービス にアクセスするユーザーです。フェデレーテッドアイデンティティ が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンター でユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM アイデンティティセンター の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM アイデンティティセンター?」(IAM アイデンティティセンター とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の権限を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、『IAM ユーザーガイド』の「IAM ロールの使用」を参照してください。

一時的な認証情報を持った IAM ロールは、以下の状況で役立ちます。

- フェデレーションユーザーユーザーアクセス – フェデレーションアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー向けロールの作成」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできま

す。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、『IAM ユーザーガイド』の「AWS のサービス に権限を委任するロールの作成」を参照してください。
  - サービスリンクロール - サービスリンクロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセス権を管理するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらの権限を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、『IAM ユーザーガイド』の「JSON ポリシー概要」を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Simple Storage Service (Amazon S3)、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- 権限の境界 - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。権限の境界の詳細については、『IAM ユーザーガイド』の「IAM エンティティの権限の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大権限を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数

の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」をご参照ください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、『IAM ユーザーガイド』の「Policy evaluation logic (ポリシーの評価ロジック)を参照してください。

AWS のサービスと IAM の連携の仕組み

AWS のサービスが IAM のほとんどの機能と連携する仕組みの概要については、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

特定の AWS のサービスで IAM を使用する方法については、該当するサービスのユーザーガイドでセキュリティに関するセクションを参照してください。

AWS ID とアクセスのトラブルシューティング

以下の情報は、AWS と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- AWS でアクションを実行する権限がない
- iam:PassRole を実行する権限がない
- 自分の AWS アカウント 以外のユーザーに AWS リソースへのアクセスを許可したい

AWS でアクションを実行する権限がない

あるアクションを実行する権限がないというエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次の例は、mateojackson という IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細を表示しようとしたとき、架空の awes:*GetWidget* アクセス許可がない場合に発生するエラーを示しています。

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

この場合、awes:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスリンクロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

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

この場合、Mary のポリシーを更新して、Mary に iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント 以外のユーザーに AWS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外のユーザーが、リソースへのアクセスに使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- AWS がこれらの機能をサポートしているかどうかを確認するには、「AWS のサービスと IAM の連携の仕組み」をご参照ください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を付与する方法については、「IAM ユーザーガイド」の「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに付与する」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「第三者が所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

この AWS 製品またはサービスのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS「サービスセキュリティドキュメント」ページとAWS 、 AWS コンプライアンスプログラム によるコンプライアンスの取り組みの対象となるサービスを参照してください。

Tools for PowerShell で最小 TLS バージョンを適用する方法

AWS サービスと通信する際のセキュリティを強化するには、Tools for PowerShell で適切な TLS バージョンを使用するように設定する必要があります。その方法については、「AWS SDK for .NET 開発者ガイド」の「最小 TLS バージョンの適用」を参照してください。

Tools for のセキュリティに関するその他の考慮事項 PowerShell

このトピックには、前のセクションで説明したセキュリティトピックに加えて、セキュリティに関する考慮事項が含まれています。

機密情報のログ記録

このツールの一部のオペレーションは、環境変数からの情報など、機密と見なされる可能性のある情報を返す場合があります。特定のシナリオでは、この情報が公開されるとセキュリティリスクが発生する可能性があります。例えば、その情報は継続的インテグレーションと継続的デプロイ (CI/CD) ログに含まれる可能性があります。したがって、ログの一部としてこのような出力を含める場合は を確認し、不要な場合は出力を抑制することが重要です。機密データの保護の詳細については、「」を参照してくださいこの AWS 製品またはサービスのデータ保護。

以下のベストプラクティスを考慮します。

- サーバーレスリソースの機密値を保存するために環境変数を使用しないでください。代わりに、サーバーレスコードをプログラムでシークレットストアからシークレットを取得します (例：AWS Secrets Manager ）。
- ビルドログの内容を確認して、機密情報が含まれていないことを確認します。/dev/null へのパイプ処理や、出力をバッシュまたは PowerShell変数としてキャプチャしてコマンド出力を抑制するなどのアプローチを検討してください。
- ログへのアクセスを検討し、ユースケースに合わせてアクセスの範囲を適切に設定します。

Tools for PowerShell 用のコマンドレットリファレンス

Tools for PowerShell には、AWS のサービスにアクセスするために使用できるコマンドレットが用意されています。使用できるコマンドレットについては、「AWS Tools for PowerShell コマンドレットリファレンス」を参照してください。

ドキュメント履歴

このトピックでは、 AWS Tools for PowerShellのドキュメントに対する重要な変更について説明します。

また、お客様からのフィードバックに応じて、定期的にドキュメントを更新します。トピックに関するフィードバックを送信するには、「このページは役に立ちましたか?」の横にある [フィードバック] ボタンを使用します。各ページの下部にあります。

の変更と更新の詳細については AWS Tools for PowerShell、「 リリースノート」を参照してください。

変更	説明	日付
コードの例	コマンドレットの例に関する章を追加しました。	2024 年 4 月 17 日
セキュリティに関するその他の考慮事項	機密データの潜在的な口グ記録に関する情報が含まれました。	2024 年 4 月 16 日
によるツール認証の設定 AWS	での SSO のサポートに関する情報を追加しました AWS Tools for PowerShell。	2024 年 3 月 15 日
Tools for のコマンドレットリファレンス PowerShell	Tools for PowerShell cmdlet リファレンスへのリンクを含むセクションを追加しました。	2023 年 11 月 17 日
IAM ベストプラクティスの更新をさらに追加	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。	2023 年 10 月 12 日
Windows でのインストール	廃止された MSI PowerShell を使用した Tools for Windows	2023 年 9 月 25 日

	のインストールに関する情報を削除しました。	
IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。	2023 年 9 月 8 日
パイプライン処理と $AWSHistory	`Set-AWSHistoryConfiguration` コマンドレットに `IncludeSensitiveData` パラメータを追加しました。	2023 年 3 月 9 日
コマンドレットでの ClientConfig パラメータの使用	ClientConfig パラメータのサポートに関する情報を追加しました。	2022 年 10 月 28 日
Windows を使用して Amazon EC2 インスタンスを起動する PowerShell	EC2-Classic の廃止に関する注記を追加しました。	2022 年 7 月 26 日
AWS Tools for PowerShell バージョン 4	バージョン 4 に関する情報を追加しました。この情報には、Windows と Linux/macOS の両方へのインストール手順、バージョン 3 との相違点と新機能を示す 移行トピックが含まれます。	2019 年 11 月 21 日

AWS Tools for PowerShell 3.3.563	`AWS.Tools.Common` モジュールのプレビューバージョンをインストールして使用する方法についての情報を追加しました。この新しいモジュールは、古いモノリシックパッケージを 1 つの共有モジュールと AWS サービスごとに 1 つのモジュールに分割します。	2019 年 10 月 18 日
AWS Tools for PowerShell 3.3.343.0	「 の使用 AWS Tools for PowerShell PowerShell 」セクションに、Tools for PowerShell Core デベロッパーが AWS Lambda 関数を構築 AWS Lambda するための情報を追加しました。	2018 年 9 月 11 日
AWS Tools for Windows PowerShell 3.1.31.0	「開始方法」セクションに新しいコマンドレットに関する情報を追加しました。これらのコマンドレットでは、Security Assertion Markup Language (SAML) を使用してユーザーのフェデレーティッド ID を設定できます。	2015 年 12 月 1 日
AWS Tools for Windows PowerShell 2.3.19	ユーザーが目的のコマンドレットをより簡単に見つけるのに役立つ新しい`Get-AWSCmdletName` コマンドレットに関する情報を AWS Cmdlets Discovery and Aliases セクションに追加しました。	2015 年 2 月 5 日

AWS Tools for Windows PowerShell 1.1.1.0	コマンドレットからのコレクション出力は常にパイプラインに列挙されます PowerShell。ページング可能なサービス呼び出しの自動サポート。新しい $AWSHistory shell 変数は、サービス応答とオプションでサービスリクエストを収集します。 AWSRegion インスタンスは、パイプライン化を支援する SystemName ために、 ではなくリージョンフィールドを使用します。Remove-S3Bucket は DeleteObjects スイッチオプションをサポートしています。Set- のユーザビリティの問題を修正しましたAWSCredentials。認証情報とリージョンデータを取得した場所からAWSDefaults レポートを初期化します。Stop-EC2Instance は、Amazon.EC2.Model.Reservation インスタンスを入力として受け入れます。汎用 List<T> パラメータタイプは、配列タイプ (T[]) に置き換えられました。リソースを削除または終了するコマンドレットは、削除する前に確認プロンプトを表示します。Write-S3Object は、Amazon S3 にアップロードするインラインテキストコンテンツをサポートしています。	2013 年 5 月 15 日

AWS Tools for Windows PowerShell 1.0.1.0	Tools for Windows PowerShell モジュールのインストール場所が変更され、Windows PowerShell バージョン 3 を使用する環境で自動ロードを利用できるようになりました。モジュールとサポートファイルは、`AWS ToolsPowerShell` の下の `AWSPowerShell` サブフォルダにインストールされるようになりました。`AWS ToolsPowerShell` フォルダに存在する旧バージョンのファイルは、インストーラによって自動的に削除されます。Windows `PSModulePath` 用 PowerShell (すべてのバージョン) は、このリリースで更新され、モジュールの親フォルダ () が含まれていますAWS `ToolsPowerShell` 。Windows PowerShell バージョン 2 を使用するシステムの場合、スタートメニューショートカットが更新され、新しい場所からモジュールがインポートされ、 が実行されますInitialize-AWSDefaults 。Windows PowerShell バージョン 3 を使用するシステムの場合、スタートメニューショートカットが更新され、 `Import-Module` コマンドが削除され、 だけ	2012 年 12 月 21 日

が残りますInitialize-AWSDefaults 。Import-Module AWSPowerShell.psd1 ファイルの を実行するようにプロファイルを編集 PowerShellした場合は、ファイルの新しい場所を指すようにプロファイルを更新する必要があります (PowerShell バージョン 3 を使用している場合は、不要になったImport-Module ステートメントを削除してください)。これらの変更の結果、 の実行時に Tools for Windows PowerShell モジュールが使用可能なモジュールとして一覧表示されるようになりましたGet-Module -ListAvailable 。さらに、Windows PowerShell バージョン 3 のユーザーの場合、モジュールによってエクスポートされたコマンドレットを実行すると、Import-Module 最初に を使用することなく、現在の PowerShell シェルにモジュールが自動的にロードされます。これにより、スクリプトの実行を許可しない実行ポリシーのあるシステムで、コマンドレットをインタラクティブに使用できるようになりました。

AWS Tools for Windows PowerShell 1.0.0.0	初回リリース	2012 年 12 月 6 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。