사용 설명서

AWS Tools for PowerShell (버전 4)

AWS Tools for PowerShell (버전 4): 사용 설명서

Table of Contents

AWS Tools for PowerShell V4가 유지 관리 모드로 전환되었습니다.

AWS Tools for PowerShell V5로 마이그레이션하는 것이 좋습니다. 마이그레이션 방법에 대한 자세한 내용과 정보는 유지 관리 모드 공지를 참조하세요.

AWS Tools for PowerShell이란 무엇입니까?

AWS Tools for PowerShell 는에서 노출되는 기능을 기반으로 구축된 PowerShell 모듈 세트입니다 AWS SDK for .NET. 를 AWS Tools for PowerShell 사용하면 PowerShell 명령줄에서 AWS 리소스에 대한 작업을 스크립팅할 수 있습니다.

cmdlet은 다양한 AWS 서비스 HTTP 쿼리 APIs를 사용하여 구현되더라도 파라미터를 지정하고 결과를 처리하기 위한 관용적인 PowerShell 환경을 제공합니다. 예를 들어의 cmdlet은 PowerShell 파이프라이닝을 AWS Tools for PowerShell 지원합니다. 즉, PowerShell 객체를 cmdlet 안팎으로 파이프할 수 있습니다.

AWS Tools for PowerShell 는 AWS Identity and Access Management (IAM) 인프라에 대한 지원을 포함하여 자격 증명을 처리할 수 있는 유연한 방법입니다. IAM 사용자 자격 증명, 임시 보안 토큰 및 IAM 역할과 함께 도구를 사용할 수 있습니다.

는 SDK에서 지원하는 것과 동일한 서비스 및 AWS 리전 세트를 AWS Tools for PowerShell 지원합니다. Windows, Linux 또는 macOS 운영 체제를 실행하는 AWS Tools for PowerShell 컴퓨터에를 설치할 수 있습니다.

> **Note**
>
> AWS Tools for PowerShell 버전 4(V4)는 AWS Tools for PowerShell 버전 3.3에 대한 이전 버전과 호환되는 업데이트입니다. 기존 cmdlet 동작을 유지하면서 기능이 상당히 향상되었습니다. V4로 업그레이드한 후에도 기존 스크립트가 계속 작동하지만 업그레이드하기 전에 철저히 테스트하는 것이 좋습니다. V4의 변경 사항에 대한 자세한 내용은 AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션 섹션을 참조하세요.

AWS Tools for PowerShell 는 다음과 같은 세 가지 패키지로 사용할 수 있습니다.

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

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 메이저 버전 및 기본 종속성의 유지 관리 및 지원에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 다음 내용을 참조하세요.

- AWS SDKs 및 도구 유지 관리 정책
- AWS SDKs 및 도구 버전 지원 매트릭스

`AWS.Tools` - 모듈화된 버전의 AWS Tools for PowerShell

이 버전의 AWS Tools for PowerShell 는 프로덕션 환경에서 PowerShell을 실행하는 모든 컴퓨터에 권장되는 버전입니다. 모듈화되었으므로 사용하려는 서비스에 대한 모듈만 다운로드하고 로드해야 합니다. 이렇게 하면 다운로드 시간 및 메모리 사용량을 줄일 수 있으며 대부분의 경우 수동으로 우선 `Import-Module`을 호출할 필요 없이 `AWS.Tools` cmdlet을 자동으로 가져올 수 있습니다.

이 버전은의 최신 버전 AWS Tools for PowerShell 이며 Windows, Linux 및 macOS를 포함하여 지원되는 모든 운영 체제에서 실행됩니다. 이 패키지는 , `AWS.Tools.Installer`, 등 각 AWS 서비스에 대해 하나의 설치 모듈`AWS.Tools.Common`, `AWS.Tools.EC2`, 하나의 공통 모듈`AWS.Tools.S3`, `AWS.Tools.IdentityManagement`, 하나의 모듈을 제공합니다.

`AWS.Tools.Installer` 모듈은 각 AWS 서비스에 대한 모듈을 설치, 업데이트 및 제거할 수 있는 cmdlet을 제공합니다. 이 모듈의 cmdlet은 사용할 모듈을 지원하는 데 필요한 모든 종속 모듈이 있는지를 자동으로 확인합니다.

이 `AWS.Tools.Common` 모듈은 서비스에 한정되지 않은 구성 및 인증을 위한 cmdlet을 제공합니다. AWS 서비스에 cmdlet을 사용하려면 명령을 실행하면 됩니다. PowerShell은 cmdlet을 실행하려는 AWS 서비스의 `AWS.Tools.Common` 모듈과 모듈을 자동으로 가져옵니다. `AWS.Tools.Installer` 모듈을 사용하여 서비스 모듈을 설치하는 경우 이 모듈은 자동으로 설치됩니다.

실행 중인 컴퓨터에이 버전의 AWS Tools for PowerShell 를 설치할 수 있습니다.

- Windows, Linux 또는 macOS의 PowerShell Core 6.0 이상
- Windows의 Windows PowerShell 5.1 이상(.NET Framework 4.7.2 이상 포함)

이 안내서에서는 이 버전만 지정해야 할 때 이를 모듈 이름 *AWS.Tools*으로 지칭하고 있습니다.

AWSPowerShell.NetCore -의 단일 모듈 버전 AWS Tools for PowerShell

이 버전은 모든 AWS 서비스에 대한 지원이 포함된 단일 대형 모듈로 구성됩니다. 이 모듈을 사용하려면 먼저 수동으로 가져와야 합니다.

실행 중인 컴퓨터에이 버전의 AWS Tools for PowerShell 를 설치할 수 있습니다.

- Windows, Linux 또는 macOS의 PowerShell Core 6.0 이상
- Windows의 Windows PowerShell 3.0 이상(.NET Framework 4.7.2 이상 포함)

이 안내서에서는 이 버전만 지정해야 할 때 이를 모듈 이름 AWSPowerShell.NetCore로 지칭하고 있습니다.

AWSPowerShell - Windows PowerShell의 단일 모듈 버전

이 버전의 AWS Tools for PowerShell 는 Windows PowerShell 버전 2.0~5.1을 실행하는 Windows 컴퓨터에서만 호환되고 설치 가능합니다. PowerShell Core 6.0 이상이나 다른 운영 체제(Linux 또는 macOS)와는 호환되지 않습니다. 이 버전은 모든 AWS 서비스에 대한 지원이 포함된 단일 대형 모듈로 구성됩니다.

이 안내서에서는 이 버전만 지정해야 할 때 이를 모듈 이름 AWSPowerShell로 지칭하고 있습니다.

이 설명서의 사용법

이 안내서는 다음과 같은 주요 단원으로 구성되어 있습니다.

설치AWS Tools for PowerShell

이 단원에서는를 설치하는 방법을 설명합니다 AWS Tools for PowerShell. 여기에는 아직 계정이 없는 AWS 경우에 가입하는 방법과 cmdlet을 실행하는 데 사용할 수 있는 IAM 사용자를 생성하는 방법이 포함됩니다.

AWS Tools for Windows PowerShell 시작

이 섹션에서는 자격 증명 및 AWS 리전 지정 AWS Tools for PowerShell, 특정 서비스에 대한 cmdlet 찾기, cmdlet에 대한 별칭 사용 등 사용의 기본 사항에 대해 설명합니다.

에서 AWS 서비스 작업 AWS Tools for PowerShell

이 단원에는를 사용하여 가장 일반적인 AWS 작업 중 일부를 수행하는 AWS Tools for PowerShell 방법에 대한 정보가 포함되어 있습니다.

이 섹션의 추가 주제

- 수정 기록
- AWS Tools for PowerShell의 새로운 기능

수정 기록

다양한 릴리스에서 변경된 내용을 찾으려면 다음을 참조하세요.

- 변경 로그
- AWS Tools for PowerShell의 새로운 기능
- 문서 기록

AWS Tools for PowerShell의 새로운 기능

AWS Tools for PowerShell과 관련된 새로운 개발에 대한 개괄적인 정보는 https://aws.amazon.com/powershell/ 제품 페이지와 변경 로그를 참조하세요.

다음은 Tools for PowerShell의 새로운 기능입니다.

2025년 9월 17일: AWS Tools for PowerShell의 버전 4에 대한 지원 종료 예정

AWS Tools for PowerShell의 이 버전(V4)에 대한 지원 종료가 발표되었습니다. V4 스크립트를 마이그레이션하고 중단을 방지하려면 V5 마이그레이션 가이드를 참조하세요. 자세한 내용은 블로그 게시물 AWS Tools for PowerShell v4에 대한 지원 종료 발표를 참조하세요.

2025년 6월 23일: AWS Tools for PowerShell의 버전 5

AWS Tools for PowerShell의 버전 5(V5)를 정식 버전으로 사용할 수 있습니다. 자세한 내용은 AWS Tools for PowerShell 사용 설명서(V5), 특히 V5로 마이그레이션 주제를 참조하세요.

2025년 2월 10일: 관찰성을 위한 GA 릴리스

관찰성은 시스템의 현재 상태를 내보내는 데이터에서 추론할 수 있는 범위입니다. 원격 측정 공급자의 구현을 포함하여 관찰성이 Tools for PowerShell에 추가되었습니다. 자세한 내용은 이 가이드의 관찰성 및 블로그 게시물 AWS .NET OpenTelemetry 라이브러리의 정식 출시 발표를 참조하세요.

2025년 1월 15일: 무결성 보호를 위한 새로운 기본 동작

AWS Tools for PowerShell의 버전 4.1.737부터 도구는 업로드에 대한 CRC32 체크섬을 자동으로 계산하여 기본 무결성 보호를 제공합니다. 자세한 내용은 https://github.com/aws/aws-tools-for-powershell/issues/370의 GitHub 공지를 참조하세요. 또한 도구는 외부에서 설정할 수 있는 데이터 무결성 보호에 대한 전역 설정을 제공하며, AWS SDK 및 도구 참조 안내서의 데이터 무결성 보호에서 확인할 수 있습니다.

2024년 11월 18일: 버전 5용 미리 보기 1 릴리스

AWS Tools for PowerShell 버전 5의 미리 보기 1은 2024년 11월 18일에 릴리스되었습니다. 이 미리 보기에 대한 자세한 내용은 블로그 게시물 AWS Tools for PowerShell V5의 미리 보기 1을 참조하세요.

설치AWS Tools for PowerShell

AWS Tools for PowerShell cmdlet을 성공적으로 설치하고 사용하려면 다음 항목의 단계를 참조하십시오.

주제

- Windows에 AWS Tools for PowerShell 설치
- Linux 또는 macOS AWS Tools for PowerShell 에 설치
- AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션

Windows에 AWS Tools for PowerShell 설치

Windows 기반 컴퓨터는 다음과 같은 AWS Tools for PowerShell 패키지 옵션을 실행할 수 있습니다.

- **`AWS.Tools`** - 의 모듈화된 버전입니다..AWS Tools for PowerShell 각 AWS 서비스는 공유 지원 모듈 `AWS.Tools.Common` 및 `AWS.Tools.Installer`을 갖춘 자체의 개별적인 소형 모듈에서 지원됩니다.
- AWSPowerShell.NetCore - AWS Tools for PowerShell의 단일 대형 모듈 버전입니다. 모든 AWS 서비스는 이 단일 대형 모듈에서 지원됩니다.

  > **Note**
  >
  > 단일 모듈이 너무 커서 AWS Lambda 함수와 함께 사용하지 못할 수 있다는 점에 유의하세요. 대신 위에 나온 모듈화된 버전을 사용합니다.

- AWSPowerShell - AWS Tools for PowerShell의.레거시 Windows용 단일 대형 모듈 버전입니다. 모든 AWS 서비스는 이 단일 대형 모듈에서 지원됩니다.

실행 중인 Windows 릴리스 및 에디션에 따라 선택하는 패키지가 다릅니다.

> **Note**
>
> AWS Tools for PowerShell는 모든 Windows 기반 Amazon Machine Images(AMI)에 기본으로 설치됩니다. 설치되는 옵션은 AMI에 따라 다릅니다. 대부분의 AMI에 AWSPowerShell 모

> 듈이 있지만, 일부 AMI에는 다른 옵션이 있을 수 있습니다. 예를 들어 Windows Server 2025용 Amazon EC2 AMI는 모듈식 `AWS.Tools` 옵션을 사용합니다.

AWS Tools for PowerShell을 설정하려면 다음과 같은 상위 수준 작업이 필요합니다(이 항목에서 자세히 설명).

1. 사용자 환경에 적합한 AWS Tools for PowerShell 패키지 옵션을 설치합니다.
2. `Get-ExecutionPolicy` cmdlet을 실행하여 스크립트 실행이 활성화되었는지 확인합니다.
3. AWS Tools for PowerShell 모듈을 PowerShell 세션으로 가져옵니다.

사전 조건

PowerShell Core를 포함한 PowerShell의 최신 버전은 Microsoft 웹 사이트의 PowerShell의 다양한 버전 설치에서 다운로드할 수 있습니다.

Windows에 **`AWS.Tools`** 설치

Windows PowerShell 버전 5.1, 또는 PowerShell Core 6.0 이상이 설치된 Windows를 실행하는 컴퓨터에 AWS Tools for PowerShell의 모듈화된 버전을 설치할 수 있습니다. PowerShell Core를 설치하는 방법에 대한 자세한 내용은 Microsoft 웹 사이트에서 PowerShell의 다양한 버전 설치를 참조하십시오.

다음 세 가지 방법 중 하나로 `AWS.Tools`를 설치할 수 있습니다.

- `AWS.Tools.Installer` 모듈에서 cmdlet을 사용합니다. 이 모듈은 다른 `AWS.Tools` 모듈의 설치 및 업데이트를 간소화합니다. `AWS.Tools.Installer`는 `PowerShellGet`의 업데이트된 버전을 요청하고 이 버전을 자동으로 다운로드하고 설치합니다. `AWS.Tools.Installer`는 자동으로 모듈 버전의 동기화를 유지합니다. 한 모듈의 최신 버전을 설치하거나 업데이트하면 `AWS.Tools.Installer`의 cmdlet은 다른 모든 `AWS.Tools` 모듈을 동일한 버전으로 자동으로 업데이트합니다.

  이 방법은 다음 절차에 설명되어 있습니다.

- `AWS.Tools.zip`에서 모듈을 다운로드하고 모듈 폴더 중 하나에 압축을 해제합니다. `PSModulePath` 환경 변수의 값을 표시하여 모듈 폴더를 찾을 수 있습니다.

> **Warning**
>
> ZIP 파일을 다운로드한 후 콘텐츠를 추출하기 전에 차단을 해제해야 할 수 있습니다. 이는 일반적으로 파일의 속성을 열고, 일반 탭을 확인하고, 있는 경우 차단 해제 확인란을 선택하여 수행됩니다.
> ZIP 파일을 차단 해제해야 하지만 차단하지 않으면 'Import-Module : Could not load file or assembly'와 유사한 오류가 발생할 수 있습니다.

- `Install-Module` cmdlet을 사용하여 PowerShell Gallery에서 각 서비스 모듈을 설치합니다.

`AWS.Tools.Installer` 모듈을 사용하여 Windows에 `AWS.Tools`를 설치하려면

1. PowerShell 세션을 시작합니다.

   > **Note**
   >
   > 수행해야 하는 작업에서 요구하는 경우를 제외하고 승격된 권한을 가진 관리자로 PowerShell을 실행하지 않는 것이 좋습니다. 이는 잠재적 보안 위험 때문이며 최소 권한의 원칙과 일치하지 않습니다.

2. 모듈화된 `AWS.Tools` 패키지를 설치하려면 다음 명령을 실행합니다.

```
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
```

   저장소를 "신뢰할 수 없음"이라는 알림을 받은 경우에도 설치를 원하는지 묻는 메시지가 표시됩니다. PowerShell에서 모듈을 설치할 수 있도록 하려면 **y**를 입력합니다. 메시지가 표시되지 않도록 하고 저장소를 신뢰하지 않은 상태에서 모듈을 설치하려면 `-Force` 파라미터로 명령을 실행할 수 있습니다.

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

3. 이제 AWS cmdlet을 통해 사용하려는 각 `Install-AWSToolsModule` 서비스에 대해 모듈을 설치할 수 있습니다. 예를 들어 다음 명령은 Amazon EC2 및 Amazon S3 모듈을 설치합니다. 이 명령은 지정된 모듈이 작동하는 데 필요한 모든 종속 모듈도 설치합니다. 예를 들어 첫 번째 `AWS.Tools` 서비스 모듈을 설치하면 `AWS.Tools.Common`도 설치됩니다. 이는 모든 AWS 서비스 모듈에 필요한 공유 모듈입니다. 또한 이전 버전의 모듈을 제거하고 다른 모듈을 동일한 최신 버전으로 업데이트합니다.

```
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**
>
> 이 `Install-AWSToolsModule` cmdlet은 이름이 `PSRepository`인 `PSGallery`(https://www.powershellgallery.com/)에서 요청된 모든 모듈을 다운로드하고 신뢰할 수 있는 소스로 간주합니다. `Get-PSRepository -Name PSGallery` 명령을 사용하여 이 `PSRepository`에 대해 자세히 알아볼 수 있습니다.

이전 명령을 실행하면 기본적으로 `%USERPROFILE%\Documents\WindowsPowerShell\Modules` 폴더에 모듈이 설치됩니다. 컴퓨터의 모든 사용자에 대해 AWS Tools for PowerShell

를 설치하려면 관리자로 시작했던 PowerShell 세션에서 다음 명령을 실행해야 합니다. 예를 들어 다음 명령은 IAM 모듈을 모든 사용자가 액세스할 수 있는 `%ProgramFiles%\WindowsPowerShell\Modules` 폴더에 설치합니다.

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

다른 모듈을 설치하려면 PowerShell Gallery에 있는 것과 같이 적절한 모듈 이름을 사용하여 유사한 명령을 실행합니다.

윈도우에 AWSPowerShell.NetCore 설치

PowerShell 버전 3~5.1, 또는 PowerShell Core 6.0 이상이 설치된 Windows를 실행하는 컴퓨터에 AWSPowerShell.NetCore를 설치할 수 있습니다. PowerShell Core 설치 방법에 대한 자세한 내용은 Microsoft PowerShell 웹 사이트의 다양한 버전의 PowerShell 설치를 참조하십시오.

AwspowerShell.NetCore는 두 가지 방법 중에 선택해서 설치할 수 있습니다.

- AWSPowerShell.NetCore.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `PSModulePath` 환경 변수의 값을 표시하여 모듈 디렉토리를 찾을 수 있습니다.

  > **Warning**
  >
  > ZIP 파일을 다운로드한 후 콘텐츠를 추출하기 전에 차단을 해제해야 할 수 있습니다. 이는 일반적으로 파일의 속성을 열고, 일반 탭을 확인하고, 있는 경우 차단 해제 확인란을 선택하여 수행됩니다.
  > ZIP 파일을 차단 해제해야 하지만 차단하지 않으면 'Import-Module : Could not load file or assembly'와 유사한 오류가 발생할 수 있습니다.

- 다음 절차에 설명된 대로 `Install-Module` cmdlet을 사용하여 PowerShell Gallery에서 설치합니다.

Install-Module cmdlet을 사용하여 PowerShell Gallery에서 AWSPowerShell.NetCore를 설치하려면 다음을 수행합니다

PowerShell Gallery에서 AWSPowerShell.NetCore를 설치하려면 컴퓨터에서 PowerShell 5.0 이상을 실행하고 있거나 PowerShell 3 이상에서 PowerShellGet을 실행하고 있어야 합니다. 다음 명령을 실행합니다.

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

PowerShell을 관리자로 실행하는 경우 이전 명령은 컴퓨터의 모든 사용자에 대해 AWS Tools for PowerShell을 설치합니다. 관리자 권한이 없는 표준 사용자로서 PowerShell을 실행하는 경우 동일한 명령이 현재 사용자에 대해서만 AWS Tools for PowerShell를 설치합니다.

해당 사용자에게 관리자 권한이 있는 경우 현재 사용자에 대해서만 설치하려면 다음과 같이 `-Scope CurrentUser` 매개 변수 세트를 사용하여 명령을 실행합니다.

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

모듈에서 cmdlet을 처음 실행할 때 일반적으로 PowerShell 3.0 이상의 릴리스는 PowerShell 세션에 모듈을 로드하지만 AWSPowerShell.NetCore 모듈은 이 기능을 지원하기에 너무 큽니다. 대신 다음 명령을 실행하여 AWSPowerShell.NetCore Core 모듈을 PowerShell 세션에 명시적으로 로드해야 합니다.

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

AWSPowerShell.NetCore 모듈을 PowerShell 세션에 자동으로 로드하려면 해당 명령을 PowerShell 프로필에 추가합니다. PowerShell 프로필 편집에 대한 자세한 내용은 PowerShell 설명서의 프로필 소개를 참조하십시오.

Windows PowerShell에 AWSPowerShell 설치

다음 두 가지 방법 중 하나로 AWS Tools for Windows PowerShell를 설치할 수 있습니다.

- AWSPowerShell.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `PSModulePath` 환경 변수의 값을 표시하여 모듈 디렉토리를 찾을 수 있습니다.

  > **Warning**
  >
  > ZIP 파일을 다운로드한 후 콘텐츠를 추출하기 전에 차단을 해제해야 할 수 있습니다. 이는 일반적으로 파일의 속성을 열고, 일반 탭을 확인하고, 있는 경우 차단 해제 확인란을 선택하여 수행됩니다.
  >
  > ZIP 파일을 차단 해제해야 하지만 차단하지 않으면 'Import-Module : Could not load file or assembly'와 유사한 오류가 발생할 수 있습니다.

- 다음 절차에 설명된 대로 `Install-Module` cmdlet을 사용하여 PowerShell Gallery에서 설치합니다.

Install-Module cmdlet을 사용하여 PowerShell Gallery에서 AWSPowerShell을 설치하려면 다음을 수행합니다

PowerShell 5.0 이상을 실행하고 있거나 PowerShell 3 이상에서 PowerShellGet을 설치했으면 PowerShell Gallery에서 AWSPowerShell을 설치할 수 있습니다. 다음 명령을 실행하여 Microsoft의 PowerShell Gallery에서 AWSPowerShell을 설치하고 업데이트할 수 있습니다.

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

AWSPowerShell 모듈을 PowerShell 세션에 자동으로 로드하려면 이전 `import-module` cmdlet을 PowerShell 프로필에 추가합니다. PowerShell 프로필 편집에 대한 자세한 내용은 PowerShell 설명서의 프로필 소개를 참조하십시오.

> **Note**
>
> Tools for Windows PowerShell은 모든 Windows 기반의 Amazon Machine Image(AMI)에 기본적으로 설치됩니다.

스크립트 실행 활성화

AWS Tools for PowerShell 모듈을 로드하려면 PowerShell 스크립트 실행을 활성화해야 합니다. 스크립트 실행을 활성화하려면 `Set-ExecutionPolicy` cmdlet을 실행하여 `RemoteSigned` 정책을 설정합니다. 자세한 내용은 Microsoft Technet 웹 사이트의 실행 정책 소개를 참조하십시오.

> **Note**
>
> 이는 Windows를 실행하는 컴퓨터에만 적용되는 요구 사항입니다. 다른 운영 체제에는 `ExecutionPolicy` 보안 제한이 없습니다.

스크립트 실행을 활성화하려면

1. 실행 정책을 설정하려면 관리자 권한이 필요합니다. 관리자 권한이 있는 사용자로 로그인하지 않은 경우, 관리자 권한으로 PowerShell 세션을 엽니다. 시작을 선택한 다음 All Programs(모든 프로그램)를 선택합니다. Accessories(액세서리)를 선택한 다음 Windows PowerShell을 선택합니다. Windows PowerShell을 마우스 오른쪽 버튼으로 클릭한 후 컨텍스트 메뉴에서 Run as administrator(관리자 권한으로 실행)를 선택합니다.

2. 명령 프롬프트에서 다음을 입력합니다.

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

> **Note**
>
> 64비트 시스템에서는 PowerShell의 32비트 버전인 Windows PowerShell(x86)에 대해 이 단계를 별도로 수행해야 합니다.

실행 정책이 올바르게 설정되지 않은 경우 프로필과 같은 스크립트를 실행하려고 할 때마다 PowerShell에서 다음과 같은 오류가 표시됩니다.

```
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 설치 관리자는 `AWSPowerShell` 모듈을 포함하는 디렉터리의 위치를 포함하도록 PSModulePath를 자동으로 업데이트합니다.

`PSModulePath`가 AWS 모듈의 디렉터리 위치를 포함하므로 `Get-Module -ListAvailable` cmdlet이 이 모듈을 표시합니다.

```
PS > Get-Module -ListAvailable

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

버전 관리

AWS는 새로운 AWS Tools for PowerShell 서비스 및 기능을 지원하기 위해 정기적으로 AWS의 새 버전을 릴리스합니다. 현재 설치된 도구의 버전을 확인하려면 Get-AWSPowerShellVersion cmdlet을 실행합니다.

예:

```
PS > Get-AWSPowerShellVersion

AWS Tools for PowerShell
Version 4.1.849
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.7.402.75
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/
CHANGELOG.ALL.md

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

또한 `-ListServiceVersionInfo` 파라미터를 Get-AWSPowerShellVersion 명령에 추가하여 도구의 현재 버전에서 지원되는 AWS 서비스의 목록을 볼 수 있습니다. 모듈화된 `AWS.Tools.*` 옵션을 사용하는 경우 현재 가져온 모듈만 표시됩니다.

예:

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

Service                           Noun Prefix Module Name                       SDK

 Assembly

 Version
-------                           ----------- -----------
 -----------
```

```
AWS IAM Access Analyzer            IAMAA       AWS.Tools.AccessAnalyzer
 3.7.400.33
AWS Account                        ACCT        AWS.Tools.Account
 3.7.400.33
AWS Certificate Manager Private... PCA         AWS.Tools.ACMPCA
 3.7.400.34
AWS Amplify                        AMP         AWS.Tools.Amplify
 3.7.401.28
Amplify Backend                    AMPB        AWS.Tools.AmplifyBackend
 3.7.400.33
...
```

실행 중인 PowerShell 버전을 확인하려면 `$PSVersionTable`을 입력하여 $PSVersionTable 자동 변수의 내용을 봅니다.

예:

```
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
```

Windows에서 AWS Tools for PowerShell 업데이트

AWS Tools for PowerShell의 업데이트된 버전이 릴리스됨에 따라 로컬에서 실행 중인 버전을 정기적으로 업데이트해야 합니다.

모듈화된 **AWS.Tools** 모듈 업데이트

`AWS.Tools` 모듈을 최신 버전으로 업그레이드하려면 다음 명령을 실행합니다.

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

이 명령은 현재 설치된 모든 AWS.Tools 모듈을 업데이트하고 성공적으로 업데이트한 후 설치되어 있는 다른 버전을 제거합니다.

> **Note**
>
> 이 Update-AWSToolsModule cmdlet은 이름이 PSRepository인 PSGallery(https://www.powershellgallery.com/)의 모든 모듈을 다운로드하고 신뢰할 수 있는 소스로 간주합니다. Get-PSRepository -Name PSGallery 명령을 사용하여 이 PSRepository에 대해 자세히 알아볼 수 있습니다.

Tools for PowerShell Core 업데이트

Get-AWSPowerShellVersion cmdlet을 실행하여 실행 중인 버전을 확인한 후, PowerShell Gallery 웹 사이트에서 제공하는 Tools for Windows PowerShell 버전과 비교합니다. 2~3주마다 확인하는 것이 좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 제공되는 버전으로 업데이트한 후에만 사용할 수 있습니다.

AWSPowerShell.NetCore의 새 릴리스를 설치하기 전에 기존 모듈을 제거합니다. 기존 패키지를 제거하기 전에 열려 있는 PowerShell 세션을 모두 닫습니다. 다음 명령을 실행하여 패키지를 제거합니다.

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

패키지를 제거한 후 다음 명령을 실행하여 업데이트 된 모듈을 설치합니다.

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

설치 후 Import-Module AWSPowerShell.NetCore 명령을 실행하여 업데이트된 cmdlet을 PowerShell 세션에 로드합니다.

Tools for Windows PowerShell 업데이트

Get-AWSPowerShellVersion cmdlet을 실행하여 실행 중인 버전을 확인한 후, PowerShell Gallery 웹 사이트에서 제공하는 Tools for Windows PowerShell 버전과 비교합니다. 2~3주마다 확인하는 것이 좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 제공되는 버전으로 업데이트한 후에만 사용할 수 있습니다.

- Install-Module cmdlet을 사용하여 설치한 경우 다음 명령을 실행합니다.

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

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

- 다운로드한 ZIP 파일을 사용하여 설치한 경우:
  1. Tools for PowerShell에서 최신 버전을 다운로드합니다. 다운로드된 파일 이름의 패키지 버전 번호를 `Get-AWSPowerShellVersion` cmdlet을 실행할 때 얻을 수 있는 버전 번호와 비교합니다.
  2. 다운로드 버전이 설치한 버전보다 번호가 높은 경우 모든 Tools for Windows PowerShell 콘솔을 닫습니다.
  3. Tools for Windows PowerShell 최신 버전을 설치합니다.

설치 후 `Import-Module AWSPowerShell`를 실행하여 업데이트된 cmdlet을 PowerShell 세션에 로드합니다. 또는 시작 메뉴에서 사용자 지정 AWS Tools for PowerShell 콘솔을 실행합니다.

Linux 또는 macOS AWS Tools for PowerShell 에 설치

이 주제에서는 Linux 또는 macOS에를 설치하는 방법에 AWS Tools for PowerShell 대한 지침을 제공합니다.

설정 개요

Linux 또는 macOS 컴퓨터에 AWS Tools for PowerShell 를 설치하려면 다음 두 가지 패키지 옵션 중에서 선택할 수 있습니다.

- **`AWS.Tools`** - 모듈화된 버전입니다 AWS Tools for PowerShell. 각 AWS 서비스는 공유 지원 모듈와 함께 자체 개별 소형 모듈에서 지원됩니다`AWS.Tools.Common`.
- AWSPowerShell.NetCore -의 단일 대형 모듈 버전입니다 AWS Tools for PowerShell. 모든 AWS 서비스는이 단일 대형 모듈에서 지원됩니다.

> **Note**
>
> 단일 모듈이 너무 커서 AWS Lambda 함수와 함께 사용하지 못할 수 있다는 점에 유의하세요. 대신 위에 나온 모듈화된 버전을 사용합니다.

Linux 또는 macOS를 실행하는 컴퓨터에서 이러한 작업을 설정하려면 다음 작업이 필요합니다(이 항목의 뒷부분에 자세히 설명).

1. 지원되는 시스템에 PowerShell Core 6.0 이상을 설치합니다.

2. PowerShell Core를 설치한 후 시스템 셸에서 pwsh를 실행하여 PowerShell을 시작합니다.
3. AWS.Tools 또는 AWSPowerShell.NetCore 중 하나를 설치합니다.
4. 적절한 Import-Module cmdlet을 실행하여 모듈을 PowerShell 세션으로 가져옵니다.
5. Initialize-AWSDefaultConfiguration cmdlet을 실행하여 AWS 자격 증명을 제공합니다.

사전 조건

를 실행하려면 AWS Tools for PowerShell Core컴퓨터에서 PowerShell Core 6.0 이상을 실행해야 합니다.

- 지원되는 Linux 플랫폼 목록 및 Linux 기반 컴퓨터에 PowerShell 최신 버전을 설치하는 방법에 대한 자세한 내용은 Microsoft 웹 사이트의 Linux에 PowerShell 설치를 참조하세요. Arch, Kali, Raspbian 등과 같은 Linux 기반의 일부 운영 체제는 공식적으로 지원되지 않지만 다양한 수준의 커뮤니티 지원이 이루어지고 있습니다.
- 지원되는 macOS 버전 및 macOS에 PowerShell 최신 버전을 설치하는 방법에 대한 자세한 내용은 Microsoft 웹 사이트의 macOS에 PowerShell 설치를 참조하세요.

Linux 또는 macOS에 **AWS.Tools** 설치

PowerShell Core 6.0 이상을 실행하는 AWS Tools for PowerShell 컴퓨터에 모듈화된 버전의를 설치할 수 있습니다. PowerShell Core 설치 방법에 대한 자세한 내용은 Microsoft PowerShell 웹 사이트의 다양한 버전의 PowerShell 설치를 참조하십시오.

다음 세 가지 방법 중 하나로 AWS.Tools를 설치할 수 있습니다.

- AWS.Tools.Installer 모듈에서 cmdlet을 사용합니다. 이 모듈은 다른 AWS.Tools 모듈의 설치 및 업데이트를 간소화합니다. AWS.Tools.Installer는 PowerShellGet의 업데이트된 버전을 요청하고 이 버전을 자동으로 다운로드하고 설치합니다. AWS.Tools.Installer는 자동으로 모듈 버전의 동기화를 유지합니다. 한 모듈의 최신 버전을 설치하거나 업데이트하면 AWS.Tools.Installer의 cmdlet은 다른 모든 AWS.Tools 모듈을 동일한 버전으로 자동으로 업데이트합니다.

  이 방법은 다음 절차에 설명되어 있습니다.
- AWS.Tools.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. $Env:PSModulePath 변수의 값을 인쇄하여 모듈 디렉터리를 찾을 수 있습니다.
- Install-Module cmdlet을 사용하여 PowerShell Gallery에서 각 서비스 모듈을 설치합니다.

AWS.Tools.Installer 모듈을 사용하여 Linux 또는 macOS에 **AWS.Tools**를 설치하려면

1. 다음 명령을 실행하여 PowerShell Core 세션을 시작합니다.

```
$ pwsh
```

> **Note**
>
> 수행해야 하는 작업에서 요구하는 경우를 제외하고 승격된 권한을 가진 관리자로 PowerShell을 실행하지 않는 것이 좋습니다. 이는 잠재적 보안 위험 때문이며 최소 권한의 원칙과 일치하지 않습니다.

2. AWS.Tools.Installer 모듈을 사용하여 모듈화된 AWS.Tools 패키지를 설치하려면 다음 명령을 실행합니다.

```
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
```

저장소가 '신뢰할 수 없음'이라는 알림이 표시되면 어쨌든 설치를 원하는지 묻는 메시지가 표시됩니다. PowerShell에서 모듈을 설치할 수 있도록 하려면 **y**를 입력합니다. 메시지가 표시되지 않도록 하고 저장소를 신뢰하지 않은 상태에서 모듈을 설치하려면 다음 명령을 실행할 수 있습니다.

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

3. 이제 사용하려는 각 서비스에 대해 모듈을 설치할 수 있습니다. 예를 들어 다음 명령은 Amazon EC2 및 Amazon S3 모듈을 설치합니다. 이 명령은 지정된 모듈이 작동하는 데 필요한 모든 종속 모듈도 설치합니다. 예를 들어 첫 번째 AWS.Tools 서비스 모듈을 설치하면 AWS.Tools.Common도 설치됩니다. 이는 모든 AWS 서비스 모듈에 필요한 공유 모듈입니다. 또한 이전 버전의 모듈을 제거하고 다른 모듈을 동일한 최신 버전으로 업데이트합니다.

```
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**
>
> 이 `Install-AWSToolsModule` cmdlet은 이름이 `PSRepository`인 `PSGallery`(https://www.powershellgallery.com/)에서 요청된 모든 모듈을 다운로드하고 해당 리포지토리를 신뢰할 수 있는 소스로 간주합니다. `Get-PSRepository -Name PSGallery` 명령을 사용하여 이 `PSRepository`에 대해 자세히 알아볼 수 있습니다.

이전 명령은 모듈을 시스템의 기본 디렉터리에 설치합니다. 실제 디렉터리는 운영 체제 배포판 및 버전과 설치한 PowerShell 버전에 따라 달라집니다. 예를 들어, RHEL과 유사한 시스템에 PowerShell 7을 설치한 경우 기본 모듈은 `/opt/microsoft/powershell/7/Modules`(또는 `$PSHOME/Modules`)에 있고 사용자 모듈은 대부분 `~/.local/share/powershell/Modules`에 있습니다. 자세한 내용은 Microsoft PowerShell 웹 사이트의 Install PowerShell on Linux를 참조하세요. 모듈이 설치된 위치를 확인하려면 다음 명령을 실행합니다.

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

다른 모듈을 설치하려면 PowerShell Gallery에 있는 것과 같이 적절한 모듈 이름을 사용하여 유사한 명령을 실행합니다.

Linux 또는 macOS에 AWSPowerShell.NetCore 설치

AWSPowerShell.NetCore의 최신 릴리스로 업그레이드하려면 Linux 또는 macOS AWS Tools for PowerShell 에서 업데이트의 지침을 따릅니다. 먼저 이전 버전의 AWSPowerShell.NetCore를 제거합니다.

AwspowerShell.NetCore는 두 가지 방법 중에 선택해서 설치할 수 있습니다.

- AWSPowerShell.NetCore.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `$Env:PSModulePath` 변수의 값을 인쇄하여 모듈 디렉터리를 찾을 수 있습니다.
- 다음 절차에 설명된 대로 `Install-Module` cmdlet을 사용하여 PowerShell Gallery에서 설치합니다.

Install-Module cmdlet을 사용하여 Linux 또는 macOS에 AWSPowerShell.NetCore를 설치하려면

다음 명령을 실행하여 PowerShell Core 세션을 시작합니다.

```
$ pwsh
```

> **Note**
>
> 승격된 관리자 권한으로 PowerShell 실행하기 위해 `sudo pwsh`를 실행해서 PowerShell을 시작하지는 마십시오. 이는 잠재적 보안 위험 때문이며 최소 권한의 원칙과 일치하지 않습니다.

PowerShell Gallery에서 AWSPowerShell.NetCore 단일 모듈 패키지를 설치하려면 다음 명령을 실행합니다.

```
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
```

저장소가 '신뢰할 수 없음'이라는 알림이 표시되면 어쨌든 설치를 원하는지 묻는 메시지가 표시됩니다. PowerShell에서 모듈을 설치할 수 있도록 하려면 **y**를 입력합니다. 저장소를 신뢰하지 않은 상태에서 메시지가 표시되지 않도록 하려면 다음 명령을 실행할 수 있습니다.

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

컴퓨터의 모든 사용자 AWS Tools for PowerShell 에 대해를 설치하지 않는 한이 명령을 루트로 실행할 필요가 없습니다. 이렇게 하려면 sudo pwsh에서 시작한 PowerShell 세션에서 다음 명령을 실행합니다.

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

스크립트 실행

이 Set-ExecutionPolicy 명령은 Windows 이외의 시스템에서는 사용할 수 없습니다. Get-ExecutionPolicy를 실행할 수 있으며, 이는 Windows 기반이 아닌 시스템에서 실행하는 PowerShell Core의 기본 실행 정책이 Unrestricted임을 보여줍니다. 자세한 내용은 Microsoft Technet 웹 사이트의 실행 정책 소개를 참조하십시오.

에는 AWS 모듈 디렉터리의 위치가 PSModulePath 포함되어 있으므로 Get-Module -ListAvailable cmdlet에는 설치한 모듈이 표시됩니다.

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
```

를 사용하도록 PowerShell 콘솔 구성 AWS Tools for PowerShell Core (AWSPowerShell.NetCore만 해당)

PowerShell Core는 일반적으로 모듈에서 cmdlet을 실행할 때마다 모듈을 자동으로 로드합니다. 그러나 이것은 크기가 큰 AWSPowerShell.NetCore에서는 효과적이지 않습니다. AWSPowerShell.NetCore cmdlet 실행을 시작하려면 먼저 `Import-Module AWSPowerShell.NetCore` 명령을 실행해야 합니다. 이는 `AWS.Tools` 모듈의 cmdlet에서는 필요하지 않습니다.

PowerShell 세션 초기화

를 설치한 후 Linux 기반 또는 macOS 기반 시스템에서 PowerShell을 시작하는 AWS Tools for PowerShell경우 Initialize-AWSDefaultConfiguration을 실행하여 사용할 AWS 액세스 키를 지정해야 합니다. `Initialize-AWSDefaultConfiguration`에 대한 자세한 정보는 AWS 자격 증명 사용 섹션을 참조하세요.

> **Note**
>
> 의 이전(3.3.96.0 이전) 릴리스에서는 AWS Tools for PowerShell이 cmdlet의 이름이 로 지정되었습니다`Initialize-AWSDefaults`.

버전 관리

AWS 는 새로운 AWS 서비스와 기능을 지원하기 위해의 새 버전을 AWS Tools for PowerShell 정기적으로 릴리스합니다. 설치 AWS Tools for PowerShell 한의 버전을 확인하려면 Get-AWSPowerShellVersion cmdlet을 실행합니다.

예제:

```
PS > Get-AWSPowerShellVersion

AWS Tools for PowerShell
Version 4.1.849
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET
Core Runtime Version 3.7.402.75
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/v4.1/changelogs/
CHANGELOG.ALL.md

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

현재 버전의 도구에서 지원되는 AWS 서비스 목록을 보려면 Get-AWSPowerShellVersion cmdlet에 `-ListServiceVersionInfo` 파라미터를 추가합니다.

실행 중인 PowerShell 버전을 확인하려면 `$PSVersionTable`을 입력하여 `$PSVersionTable` 자동 변수의 내용을 봅니다.

예제:

```
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 또는 macOS AWS Tools for PowerShell 에서 업데이트

정기적으로의 업데이트된 버전이 AWS Tools for PowerShell 릴리스되면 로컬에서 실행 중인 버전을 업데이트해야 합니다.

모듈화된 **`AWS.Tools`** 모듈 업데이트

`AWS.Tools` 모듈을 최신 버전으로 업그레이드하려면 다음 명령을 실행합니다.

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

이 명령은 현재 설치된 모든 AWS.Tools 모듈을 업데이트하고 성공적으로 업데이트된 모듈에 대해서는 이전 버전을 제거합니다.

> **Note**
>
> 이 Update-AWSToolsModule cmdlet은 이름이 PSRepository인 PSGallery(https://www.powershellgallery.com/)의 모든 모듈을 다운로드하고 신뢰할 수 있는 소스로 간주합니다. Get-PSRepository -Name PSGallery 명령을 사용하여 이 PSRepository에 대해 자세히 알아볼 수 있습니다.

Tools for PowerShell Core 업데이트

Get-AWSPowerShellVersion cmdlet을 실행하여 실행 중인 버전을 확인한 후, PowerShell Gallery 웹 사이트에서 제공하는 Tools for Windows PowerShell 버전과 비교합니다. 2~3주마다 확인하는 것이 좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 포함된 버전으로 업데이트한 후에만 사용할 수 있습니다.

AWSPowerShell.NetCore의 새 릴리스를 설치하기 전에 기존 모듈을 제거합니다. 기존 패키지를 제거하기 전에 열려 있는 PowerShell 세션을 모두 닫습니다. 다음 명령을 실행하여 패키지를 제거합니다.

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

패키지를 제거한 후 다음 명령을 실행하여 업데이트 된 모듈을 설치합니다.

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

설치 후 Import-Module AWSPowerShell.NetCore 명령을 실행하여 업데이트된 cmdlet을 PowerShell 세션에 로드합니다.

관련 정보

- AWS Tools for Windows PowerShell 시작
- 에서 AWS 서비스 작업 AWS Tools for PowerShell

AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션

AWS Tools for PowerShell 버전 4는 AWS Tools for PowerShell 버전 3.3에 대한 이전 버전과 호환되는 업데이트입니다. 기존 cmdlet 동작을 유지하면서 기능이 상당히 향상되었습니다.

새 버전으로 업그레이드한 후에도 기존 스크립트가 계속 작동하지만 프로덕션 환경을 업그레이드하기 전에 철저히 테스트하는 것이 좋습니다.

이 단원에서는 변경 사항에 대해 설명하고 이러한 변경 사항이 스크립트에 어떤 영향을 미칠 수 있는지에 대해 다룹니다.

새롭게 완전 모듈화된 **AWS.Tools** 버전

AWSPowerShell.NetCore 및 AWSPowerShell 패키지는 '모놀리식'이었습니다. 즉, 모든 AWS 서비스가 동일한 모듈에서 지원되어 매우 크고 새로운 AWS 서비스와 기능이 추가될 때마다 더 커집니다. 새 `AWS.Tools` 패키지는 사용하는 AWS 서비스에 필요한 모듈만 다운로드하고 설치할 수 있는 유연성을 제공하는 더 작은 모듈로 나뉩니다. 패키지에는 다른 모든 모듈에서 필요로 하는 공유 `AWS.Tools.Common` 모듈과 필요에 따라 모듈의 설치, 업데이트 및 제거를 간소화하는 `AWS.Tools.Installer` 모듈이 포함되어 있습니다.

또한 `Import-module`을 우선 호출하지 않고도 첫 번째 호출 시 cmdlet을 자동으로 가져올 수 있습니다. 하지만 cmdlet을 호출하기 전에 연결된 .NET 객체와 상호 작용하려면 여전히 `Import-Module`을 호출하여 관련 .NET 유형에 대해 PowerShell에 알려야 합니다.

예를 들어 다음 명령에는 `Amazon.EC2.Model.Filter`에 대한 참조가 있습니다. 이 유형의 참조는 자동 가져오기를 트리거할 수 없으므로 우선 `Import-Module`을 호출해야 합니다. 그렇지 않으면 명령이 실패합니다.

```
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
```

새로운 **Get-AWSService** cmdlet

모듈 AWS.Tools 모음에서 각 AWS 서비스의 모듈 이름을 쉽게 찾을 수 있도록 Get-AWSService cmdlet을 사용할 수 있습니다.

```
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
  ...
```

Cmdlet에서 반환된 객체를 제어하는 새로운 **-Select** 파라미터

버전 4에서 대부분의 cmdlet은 새로운 -Select 파라미터를 지원합니다. 각 cmdlet은 AWS SDK for .NET를 사용하여 AWS 서비스 API를 호출합니다. 그런 다음 AWS Tools for PowerShell 클라이언트는 응답을 PowerShell 스크립트에서 사용할 수 있는 객체로 변환하고 다른 명령으로 파이프합니다. 원래 응답에 필요한 것보다 많은 필드 또는 속성이 최종 PowerShell 객체에 포함되어 있는 경우도 있고, 기본적으로 존재하지 않는 원래 응답의 필드 또는 속성을 이 객체에 포함시켜야 할 경우도 있을 수 있습니다. -Select 파라미터를 사용하면 cmdlet에서 반환된 .NET 객체에 포함시킬 사항을 지정할 수 있습니다.

예를 들어 Get-S3Object cmdlet은 Amazon S3 SDK 작업 ListObjects를 호출합니다. 이 작업은 ListObjectsResponse 객체를 반환합니다. 하지만 기본적으로 Get-S3Object cmdlet은 SDK 응답의 S3Objects 요소만 PowerShell 사용자에게 반환합니다. 다음 예제에서 이 객체는 2개 원소가 있는 배열입니다.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket

ETag : "01234567890123456789012345678901111"
BucketName : amzn-s3-demo-bucket
Key : file1.txt
LastModified : 9/30/2019 1:31:40 PM
```

```
Owner : Amazon.S3.Model.Owner
Size : 568
StorageClass : STANDARD

ETag : "01234567890123456789012345678902222"
BucketName : amzn-s3-demo-bucket
Key : file2.txt
LastModified : 7/15/2019 9:36:54 AM
Owner : Amazon.S3.Model.Owner
Size : 392
StorageClass : STANDARD
```

AWS Tools for PowerShell 버전 4에서는 SDK API 호출에서 반환되는 전체 .NET 응답 객체를 반환-Select *하도록를 지정할 수 있습니다.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select *
  IsTruncated    : False
  NextMarker     :
  S3Objects      : {file1.txt, file2.txt}
  Name           : amzn-s3-demo-bucket
  Prefix         :
  MaxKeys        : 1000
  CommonPrefixes : {}
  Delimiter      :
```

또한 원하는 특정 중첩 속성에 대한 경로를 지정할 수도 있습니다. 다음 예제에서는 S3Objects 배열에 있는 각 원소의 Key 속성만 반환합니다.

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

특정 상황에서는 cmdlet 파라미터를 반환하는 것이 유용할 수 있습니다. -Select ^ParameterName을 사용해 이 작업을 수행할 수 있습니다. 이 기능은 -PassThru 파라미터(여전히 사용 가능하지만 더 이상 사용되지 않음)를 대체합니다.

```
PS > Get-S3Object -BucketName amzn-s3-demo-bucket -Select S3Objects.Key |
>> Write-S3ObjectTagSet -Select ^Key -BucketName amzn-s3-demo-bucket -Tagging_TagSet
 @{ Key='key'; Value='value'}
  file1.txt
  file2.txt
```

각 cmdlet에 대한 참조 항목에서 -Select 파라미터를 지원하는지 여부를 식별합니다.

출력 항목 수에 대한 보다 일관된 제한

이전 버전의 에서는 -MaxItems 파라미터를 사용하여 최종 출력에서 반환되는 최대 객체 수를 지정할 수 AWS Tools for PowerShell 있었습니다.

이 동작은 AWS.Tools에서 제거됩니다.

이 동작은 AWSPowerShell.NetCore 및 AWSPowerShell에서 더 이상 사용되지 않으며 향후 릴리스의 해당 버전에서 제거됩니다.

기본 서비스 API가 MaxItems 파라미터를 지원하는 경우, 계속 사용할 수 있으며 API에서 지정한 대로 작동합니다. 하지만 cmdlet의 출력에서 반환되는 항목 수를 제한하는 동작을 더 이상 수행하지 않습니다.

최종 출력에서 반환되는 항목 수를 제한하려면 출력을 Select-Object cmdlet으로 파이핑하고 -First *n* 파라미터를 지정합니다. 여기서 *n*은 최종 출력에 포함할 최대 항목 수입니다.

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

모든 AWS 서비스가 -MaxItems 동일한 방식으로 지원되는 것은 아니므로 이로 인해 불일치와 때때로 발생한 예상치 못한 결과가 제거됩니다. 또한 -MaxItems가 새로운 -Select 파라미터와 결합되면 때때로 혼란스러운 결과가 발생할 수 있습니다.

더욱 사용하기 쉬워진 Stream 파라미터

Stream 또는 byte[] 유형의 파라미터는 이제 string, string[] 또는 FileInfo 값을 수락합니다.

예를 들어 다음 예제 중 하나를 사용할 수 있습니다.

```
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 는 UTF-8 인코딩을 byte[] 사용하여 모든 문자열을 로 변환합니다.

속성 이름별로 파이프 확장

이제 모든 파라미터에 대한 속성 이름을 지정하여 파이프라인 입력을 전달할 수 있으므로 보다 일관된 사용자 경험을 제공할 수 있습니다.

다음 예제에서는 대상 cmdlet의 파라미터 이름과 일치하는 이름을 가진 속성이 있는 사용자 지정 객체를 만듭니다. cmdlet이 실행되면 이러한 속성을 자동으로 해당 파라미터로 사용합니다.

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

> **Note**
>
> 일부 속성은의 이전 버전에서 이를 지원했습니다 AWS Tools for PowerShell. 버전 4에서는 모든 파라미터에 대해 이를 활성화함으로써 일관성을 개선했습니다.

정적 공통 파라미터

버전 4.0의 일관성을 개선하기 위해 AWS Tools for PowerShell모든 파라미터는 정적입니다.

이전 버전에서는 AWS Tools for PowerShell,, AccessKeySecretKey ProfileName또는와 같은 일부 공통 파라미터가 동적Region이었던 반면 다른 모든 파라미터는 정적이었습니다. 이 경우 PowerShell은 동적 파라미터보다 먼저 정적 파라미터를 바인딩하기 때문에 문제가 발생할 수 있습니다. 예를 들어 다음 명령을 실행했다고 가정해보겠습니다.

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

PowerShell의 이전 버전에서는 값 us-west-2를 -Region 동적 파라미터가 아니라 -RegionName 정적 파라미터에 바인딩했습니다. 이러한 수행은 사용자를 혼란스럽게 할 수 있습니다.

AWS.Tools 필수 파라미터 선언 및 강제 시행

AWS.Tools.* 모듈은 이제 필수 cmdlet 파라미터를 선언하고 강제 시행합니다. AWS 서비스에서 API의 파라미터가 필요하다고 선언한 경우, cmdlet 파라미터를 지정하지 않으면 PowerShell은 해당 cmdlet 파라미터를 묻는 메시지를 표시합니다. 이것은 AWS.Tools에만 적용됩니다. 이전 버전과의 호환성을 보장하기 위해 AWSPowerShell.NetCore 또는 AWSPowerShell에는 적용되지 않습니다.

모든 파라미터에 Null 값 사용 가능

이제 값 유형 파라미터(숫자 및 날짜)에 $null을 지정할 수 있습니다. 이 변경 사항은 기존 스크립트에 영향을 주지 않습니다. 이렇게 하면 필수 파라미터에 대해 표시되는 메시지를 우회할 수 있습니다. 필수 파라미터는 AWS.Tools에서만 강제 시행됩니다.

버전 4를 사용하여 다음 예제를 실행하면 각 필수 파라미터에 "값"을 제공하기 때문에 클라이언트 측 검증을 실질적으로 우회할 수 있습니다. 그러나 Amazon EC2 API 서비스 호출은 AWS 서비스에 여전히 해당 정보가 필요하기 때문에 실패합니다.

```
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
```

더 이상 사용되지 않는 기능 제거

다음 기능은의 이전 릴리스에서 더 이상 사용되지 AWS Tools for PowerShell 않으며 버전 4에서 제거됩니다.

- Stop-EC2Instance cmdlet에서 -Terminate 파라미터가 제거되었습니다. 대신 Remove-EC2Instance을 사용하세요.
- Clear-AWSCredential cmdlet에서 -ProfileName 파라미터가 제거되었습니다. 대신 Remove-AWSCredentialProfile을 사용하세요.

- cmdlet `Import-EC2Instance` 및 `Import-EC2Volume`이 제거되었습니다.

AWS Tools for Windows PowerShell 시작

이 섹션의 일부 주제에서는 도구를 설치한 후 Tools for Windows PowerShell 사용의 기본 사항에 대해 설명합니다. 예를 들면 Tools for Windows PowerShell에서 AWS와 상호 작용할 때 사용해야 할 보안 인증 정보와 AWS 리전을 지정하는 방법을 설명합니다.

이 섹션의 다른 주제에서는 도구, 환경 및 프로젝트를 구성할 수 있는 고급 방법에 대한 정보를 제공합니다.

주제

- AWS를 사용하여 도구 인증 구성
- AWS 리전 지정
- 를 사용하여 페더레이션 ID 구성 AWS Tools for PowerShell
- Cmdlet 검색 및 별칭
- AWS Tools for PowerShell의 파이프라이닝, 출력 및 반복
- 보안 인증 정보 및 프로파일 확인
- 사용자 및 역할에 대한 추가 정보
- 레거시 보안 인증 사용

AWS를 사용하여 도구 인증 구성

AWS 서비스로 개발할 때는 코드가 AWS에서 인증되는 방법을 설정해야 합니다. 환경 및 사용 가능한 AWS 액세스에 따라 AWS 리소스에 대한 프로그래밍 방식의 액세스를 구성할 수 있는 다양한 방법이 있습니다.

PowerShell용 도구에 대한 다양한 인증 방법을 확인하려면 AWS SDK 및 도구 참조 가이드에서 인증 및 액세스를 참조하세요.

이 주제에서는 새 사용자가 로컬에서 개발 중이고, 고용주로부터 인증 방법을 받지 않았으며, 임시 보안 인증 정보를 얻기 위해 AWS IAM Identity Center를 사용한다고 가정합니다. 사용자 환경이 이러한 가정에 해당하지 않는 경우 이 주제의 일부 정보가 사용자에게 적용되지 않거나 일부 정보가 이미 제공되었을 수 있습니다.

이 환경을 구성하려면 몇 가지 단계를 수행해야 하며, 요약하면 다음과 같습니다.

1. IAM Identity Center 사용 및 구성
2. PowerShell용 도구가 IAM Identity Center를 사용하도록 구성합니다.
3. AWS 액세스 포털 세션 시작

IAM Identity Center 사용 및 구성

AWS IAM Identity Center를 사용하려면 먼저 사용하도록 설정하고 구성해야 합니다. PowerShell에서 이 작업을 수행하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 IAM Identity Center 인증 항목에 있는 1단계를 참조하세요. 특히 IAM Identity Center를 통한 액세스가 설정되어 있지 않습니다 아래에 있는 필요한 지침을 따르세요.

PowerShell용 도구가 IAM Identity Center를 사용하도록 구성합니다.

> **Note**
>
> Tools for PowerShell의 버전 4.1.538부터 SSO 자격 증명을 구성하고 AWS 액세스 포털 세션을 시작하는 데 권장되는 방법은 이 주제에 설명된 대로 `Initialize-AWSSSOConfiguration` 및 `Invoke-AWSSSOLogin` cmdlet을 사용하는 것입니다. 해당 버전의 Tools for PowerShell(또는 이후 버전)에 액세스할 수 없거나 이러한 cmdlet을 사용할 수 없는 경우에도 AWS CLI를 사용하여 이러한 작업을 수행할 수 있습니다. 방법을 알아보려면 포털 로그인에 AWS CLI 사용 섹션을 참조하세요.

다음 절차에서는 Tools for PowerShell이 임시 자격 증명을 얻는 데 사용하는 SSO 정보로 공유된 AWS `config` 파일을 업데이트합니다. 이 절차를 수행하면 AWS 액세스 포털 세션도 시작됩니다. 공유된 `config` 파일에 이미 SSO 정보가 있고 Tools for PowerShell을 사용하여 액세스 포털 세션을 시작하는 방법만 확인하려면 이 주제의 다음 섹션인 AWS 액세스 포털 세션 시작 섹션을 참조하세요.

1. 아직 설치하지 않은 경우 PowerShell을 열고 일반 cmdlet을 포함하여 운영 체제 및 환경에 맞게 AWS Tools for PowerShell을 설치합니다. 이를 위한 자세한 방법은 설치AWS Tools for PowerShell 섹션을 참조하세요.

   예를 들어 Windows에 Tools for PowerShell의 모듈화된 버전을 설치하는 경우 다음과 유사한 명령을 실행할 가능성이 높습니다.

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

2. 다음 명령을 실행합니다. 예제 속성 값을 IAM Identity Center 구성의 값으로 바꿉니다. 이러한 속성과 해당 속성을 찾는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 IAM Identity Center 자격 증명 공급자 설정을 참조하세요.

```
$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
```

또는 cmdlet 자체, `Initialize-AWSSSOConfiguration`을 사용하면 Tools for PowerShell에서 속성 값을 입력하라는 메시지를 표시합니다.

특정 속성 값에 대한 고려 사항:

- 지침에 따라 IAM Identity Center를 활성화하고 구성한 경우 `-RoleName`의 값은 `PowerUserAccess`일 수 있습니다. 그러나 PowerShell 작업 전용으로 IAM Identity Center 권한 집합을 생성한 경우 이를 대신 사용합니다.
- IAM Identity Center를 구성한 AWS 리전을 사용해야 합니다.

3. 이제 PowerShell용 도구에서 참조할 수 있는 구성 값 집합이 있는 `my-sso-profile`이라는 프로필이 포함된 공유 AWS `config` 파일입니다. 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 공유 파일의 위치를 참조하세요.

PowerShell용 도구는 AWS에 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 얻습니다. `sso_role_name` 값은 IAM 신원 센터 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용되는 AWS 서비스 서비스에 대한 액세스를 허용해야 합니다.

다음 샘플은 위에 표시된 명령을 사용하여 생성된 프로필을 보여줍니다. 일부 속성 값과 순서는 실제 프로필에서 다를 수 있습니다. 프로필의 `sso-session` 속성은 AWS 액세스 포털 세션을 시작하는 설정이 포함된 `my-sso-session`이라는 섹션을 참조합니다.

```
[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. 이미 활성 AWS 액세스 포털 세션이 있는 경우 Tools for PowerShell에서 이미 로그인되어 있음을 알립니다.

   그렇지 않은 경우 Tools for PowerShell은 기본 웹 브라우저에서 SSO 권한 부여 페이지를 자동으로 열려고 시도합니다. 브라우저의 프롬프트를 따르세요. 여기에는 SSO 권한 부여 코드, 사용자 이름 및 암호, AWS IAM Identity Center 계정 및 권한 집합에 액세스할 수 있는 권한이 포함될 수 있습니다.

   Tools for PowerShell에서 SSO 로그인에 성공했음을 알려줍니다.

AWS 액세스 포털 세션 시작

AWS 서비스에 액세스하는 명령을 실행하기 전에 Tools for PowerShell에서 IAM Identity Center 인증을 사용하여 보안 인증을 확인하려면 활성 AWS 액세스 포털 세션이 있어야 합니다. AWS 액세스 포털에 로그인하려면 PowerShell에서 다음 명령을 실행합니다. 여기서 `-ProfileName my-sso-profile`은 이 주제의 이전 섹션에 있는 절차를 따랐을 때 공유 `config` 파일에 생성된 프로필의 이름입니다.

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

이미 활성 AWS 액세스 포털 세션이 있는 경우 Tools for PowerShell에서 이미 로그인되어 있음을 알립니다.

그렇지 않은 경우 Tools for PowerShell은 기본 웹 브라우저에서 SSO 권한 부여 페이지를 자동으로 열려고 시도합니다. 브라우저의 프롬프트를 따르세요. 여기에는 SSO 권한 부여 코드, 사용자 이름 및 암호, AWS IAM Identity Center 계정 및 권한 집합에 액세스할 수 있는 권한이 포함될 수 있습니다.

Tools for PowerShell에서 SSO 로그인에 성공했음을 알려줍니다.

이미 활성 세션이 있는지 테스트하려면 필요에 따라 `AWS.Tools.SecurityToken` 모듈을 설치하거나 가져온 후 다음 명령을 실행합니다.

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

`Get-STSCallerIdentity` cmdlet에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고합니다.

예시

다음은 PowerShell용 도구와 함께 IAM Identity Center를 사용하는 방법의 예제입니다. 이 예제에서는 다음을 가정합니다.

- 이 주제의 앞부분에서 설명한 대로 IAM Identity Center를 사용하도록 설정하고 구성했습니다. SSO 속성은 이 주제의 앞부분에서 구성한 `my-sso-profile` 프로필에 있습니다.
- `Initialize-AWSSSOConfiguration` 또는 `Invoke-AWSSSOLogin` cmdlet을 통해 로그인하는 경우 해당 사용자는 최소한 Amazon S3에 대한 읽기 전용 권한이 있습니다.
- 해당 사용자는 일부 S3 버킷을 볼 수 있습니다.

필요에 따라 `AWS.Tools.S3` 모듈을 설치하거나 가져온 후, 다음 PowerShell 명령을 사용하여 S3 버킷 목록을 표시합니다.

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

추가 정보

- 프로필 및 환경 변수 사용과 같은 PowerShell용 도구 인증에 대한 자세한 옵션은 AWS SDK 및 도구 참조 가이드의 구성 장을 참조하세요.
- 일부 명령은 AWS 리전을 지정해야 합니다. `-Region` cmdlet 옵션, `[default]` 프로필, `AWS_REGION` 환경 변수 등 여러 가지 방법으로 이를 수행할 수 있습니다. 자세한 정보는 이 가이드의 AWS 리전 지정 섹션과 AWS SDK 및 도구 참조 가이드의 AWS 리전을 참조하세요.
- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 IAM의 보안 모범 사례를 참조하세요.
- 단기 AWS 보안 인증 정보를 만들려면 IAM 사용 설명서에서 임시 보안 인증 정보를 참조하세요.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 표준화된 보안 인증 공급자를 참조하세요.

주제

- 포털 로그인에 AWS CLI 사용

포털 로그인에 AWS CLI 사용

Tools for PowerShell의 버전 4.1.538부터 SSO 자격 증명을 구성하고 AWS 액세스 포털 세션을 시작하는 데 권장되는 방법은 AWS를 사용하여 도구 인증 구성에 설명된 대로 `Initialize-AWSSSOConfiguration` 및 `Invoke-AWSSSOLogin` cmdlet을 사용하는 것입니다. 해당 버전의 Tools for PowerShell(또는 이후 버전)에 액세스할 수 없거나 이러한 cmdlet을 사용할 수 없는 경우에도 AWS CLI를 사용하여 이러한 작업을 수행할 수 있습니다.

AWS CLI를 통해 Tools for PowerShell이 IAM Identity Center를 사용하도록 구성합니다.

아직 활성화하지 않은 경우 계속하기 전에 IAM Identity Center를 활성화하고 구성해야 합니다.

AWS CLI를 통해 IAM Identity Center를 사용하도록 Tools for PowerShell을 구성하는 방법에 대한 정보는 AWS SDK 및 도구 참조 가이드의 IAM Identity Center 인증에 대한 항목의 2단계에 나와 있습니다. 이 구성을 완료하면 시스템에 다음 요소가 포함되어야 합니다.

- AWS CLI는 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 사용합니다.
- PowerShell용 도구에서 참조할 수 있는 구성 값 집합이 있는 `[default]` 프로필이 포함된 공유 AWS `config` 파일입니다. 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 공유 파일의 위치를 참조하세요. PowerShell용 도구는 AWS에 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 얻습니다. `sso_role_name` 값은 IAM 신원 센터 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용되는 AWS 서비스 서비스에 대한 액세스를 허용해야 합니다.

  다음 샘플 `config` 파일은 SSO 토큰 공급자 구성으로 설정된 `[default]` 프로필을 보여줍니다. 프로필의 `sso_session` 설정은 이름이 지정된 `sso-session` 섹션을 참조합니다. `sso-session` 섹션에는 AWS 액세스 포털 세션을 시작하기 위한 설정이 포함되어 있습니다.

```
[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 확인이 작동하려면 PowerShell 세션에 다음 모듈이 설치되어 있고 가져와야 합니다.
>
> - `AWS.Tools.SSO`
> - `AWS.Tools.SSOOIDC`
>
> 이전 버전의 Tools for PowerShell을 사용하고 있으며 이러한 모듈이 없는 경우 'Assembly AWSSDK.SSOOIDC could not be found...'와 유사한 오류가 발생합니다.

AWS 액세스 포털 세션 시작

AWS 서비스에 액세스하는 명령을 실행하기 전에 Tools for Windows PowerShell에서 IAM Identity Center 인증을 사용하여 보안 인증을 확인하려면 활성 AWS 액세스 포털 세션이 있어야 합니다. 구성된 세션 길이에 따라 결국 액세스가 만료되고 Windows PowerShell용 도구에 인증 오류가 발생합니다. AWS 액세스 포털에 로그인하려면 AWS CLI에서 다음 명령을 실행합니다.

```
aws sso login
```

`[default]` 프로필을 사용하므로 `--profile` 옵션으로 명령을 직접 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 `aws sso login --profile named-profile` 명령을 대신 사용합니다. 이름이 지정된 프로필에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 프로필 섹션을 참조하세요.

이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다(명명된 프로필에 대해 동일한 고려 사항 사용).

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

이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고해야 합니다.

> **Note**
>
> 이미 활성 AWS 액세스 포털 세션이 있고 `aws sso login`을 실행하는 경우 보안 인증 정보를 제공하지 않아도 됩니다.

로그인 과정에서 데이터 AWS CLI 액세스를 허용하라는 메시지가 표시될 수 있습니다. AWS CLI는 Python용 SDK를 기반으로 구축되므로 권한 메시지에는 `botocore` 이름의 변형이 포함될 수 있습니다.

예시

다음은 PowerShell용 도구와 함께 IAM Identity Center를 사용하는 방법의 예제입니다. 이 예제에서는 다음을 가정합니다.

- 이 주제의 앞부분에서 설명한 대로 IAM Identity Center를 사용하도록 설정하고 구성했습니다. SSO 속성은 `[default]` 프로필에 있습니다.
- `aws sso login`을 사용하여 AWS CLI를 통해 로그인하는 경우 해당 사용자는 최소한 Amazon S3에 대한 읽기 전용 권한이 있습니다.
- 해당 사용자는 일부 S3 버킷을 볼 수 있습니다.

다음 PowerShell 명령을 사용하여 S3 버킷 목록을 표시합니다.

```
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
```

위에서 언급한 것처럼 `[default]` 프로필을 사용하므로 `-ProfileName` 옵션을 사용하여 `Get-S3Bucket` cmdlet을 직접 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용

하는 경우 `Get-S3Bucket -ProfileName` *named-profile* 명령을 사용합니다. 이름이 지정된 프로필에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 프로필 섹션을 참조하세요.

추가 정보

- 프로필 및 환경 변수 사용과 같은 PowerShell용 도구 인증에 대한 자세한 옵션은 AWS SDK 및 도구 참조 가이드의 구성 장을 참조하세요.
- 일부 명령은 AWS 리전을 지정해야 합니다. `-Region` cmdlet 옵션, `[default]` 프로필, `AWS_REGION` 환경 변수 등 여러 가지 방법으로 이를 수행할 수 있습니다. 자세한 정보는 이 가이드의 AWS 리전 지정 섹션과 AWS SDK 및 도구 참조 가이드의 AWS 리전을 참조하세요.
- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 IAM의 보안 모범 사례를 참조하세요.
- 단기 AWS 보안 인증 정보를 만들려면 IAM 사용 설명서에서 임시 보안 인증 정보를 참조하세요.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 표준화된 보안 인증 공급자를 참조하세요.

AWS 리전 지정

AWS Tools for PowerShell 명령을 실행할 때 사용할 AWS 리전을 지정하는 방법에는 두 가지가 있습니다.

- 개별 명령에서 `-Region` 공통 매개 변수를 사용합니다.
- `Set-DefaultAWSRegion` 명령을 사용하여 모든 명령에 대한 기본 리전을 설정합니다.

Tools for Windows PowerShell에서 사용할 리전을 확인할 수 없는 경우 많은 AWS cmdlet이 실패합니다. 예외에는 Amazon Amazon S3 Amazon SES 및에 대한 cmdlet이 포함되며 AWS Identity and Access Management, 이는 자동으로 글로벌 엔드포인트로 기본 설정됩니다.

단일 AWS 명령에 대한 리전을 지정하려면

다음과 같이 `-Region` 매개 변수를 명령에 추가합니다.

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

현재 세션의 모든 AWS CLI 명령에 대해 기본 리전을 설정하려면

PowerShell 명령 프롬프트에서 다음 명령을 입력합니다.

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

> **Note**
>
> 이 설정은 현재 세션 동안만 지속됩니다. 이 설정을 모든 PowerShell 세션에 적용하려면 `Import-Module` 명령에 추가할 때처럼 PowerShell 프로파일에 이 명령을 추가합니다.

모든 AWS CLI 명령에 대한 현재 기본 리전을 보려면

PowerShell 명령 프롬프트에서 다음 명령을 입력합니다.

```
PS > Get-DefaultAWSRegion

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

모든 AWS CLI 명령에 대해 현재 기본 리전을 지우려면

PowerShell 명령 프롬프트에서 다음 명령을 입력합니다.

```
PS > Clear-DefaultAWSRegion
```

사용 가능한 모든 AWS 리전 목록을 보려면

PowerShell 명령 프롬프트에서 다음 명령을 입력합니다. 샘플 출력의 세 번째 열은 현재 세션의 기본 값인 리전을 나타냅니다.

```
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**
>
> 일부 리전은 지원되지만 `Get-AWSRegion` cmdlet의 출력에 포함되지 않을 수 있습니다. 예를 들어, 아직 글로벌 지역이 아닌 리전에서 이러한 문제가 발생할 수 있습니다. `-Region` 파라미터를 추가하여 리전을 지정할 수 없는 경우, 대신에 다음 단원에서와 같이 사용자 지정 엔드포인트에서 리전을 지정해 보세요.

사용자 지정 또는 비표준 엔드포인트 지정

다음과 같은 샘플 형식으로 `-EndpointUrl` 일반 파라미터를 Tools for Windows PowerShell 명령에 추가하여 사용자 지정 엔드포인트를 URL로 지정합니다.

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

다음은 `Get-EC2Instance` cmdlet 사용을 보여주는 예입니다. 이 예에서는 사용자 지정 엔드포인트가 `us-west-2` 또는 미국 서부(오레곤)에 있지만, `Get-AWSRegion`을 통해 열거되지 않는 리전을 포함하여 지원되는 다른 AWS 리전을 모두 사용할 수 있습니다.

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

추가 정보

AWS 리전에 대한 자세한 내용은 SDK 및 도구 참조 안내서의 AWS 리전을 참조하세요. AWS SDKs

를 사용하여 페더레이션 ID 구성 AWS Tools for PowerShell

조직의 사용자가 AWS 리소스에 액세스할 수 있도록 하려면 보안, 감사 가능성, 규정 준수 및 역할 및 계정 분리를 지원하는 기능을 위해 반복 가능한 표준 인증 방법을 구성해야 합니다. 페더레이션 AWS APIs 액세스 없이 API에 액세스할 수 있는 기능을 사용자에게 제공하는 것이 일반적이지만 페더레이션 사용의 목적을 무력화하는 AWS Identity and Access Management (IAM) 사용자도 생성해야 합니다. 이 주제에서는 페더레이션 액세스 솔루션을 용이하게 AWS Tools for PowerShell 하는의 SAML(Security Assertion Markup Language) 지원에 대해 설명합니다.

의 SAML 지원을 AWS Tools for PowerShell 통해 사용자에게 서비스에 대한 페더레이션 액세스를 제공할 수 AWS 있습니다. SAML은 서비스 간에, 특히 자격 증명 공급자(예: Active Directory Federation Services)와 서비스 공급자(예:) 간에 사용자 인증 및 권한 부여 데이터를 전송하기 위한 XML 기반 개방형 표준 형식입니다 AWS. SAML과 그 작동 방식에 대한 자세한 내용은 Wikipedia의 SAML 또는 Organization for the Advancement of Structured Information Standards(OASIS) 웹 사이트의 SAML Technical Specifications를 참조하십시오. 의 SAML 지원 AWS Tools for PowerShell 은 SAML 2.0과 호환됩니다.

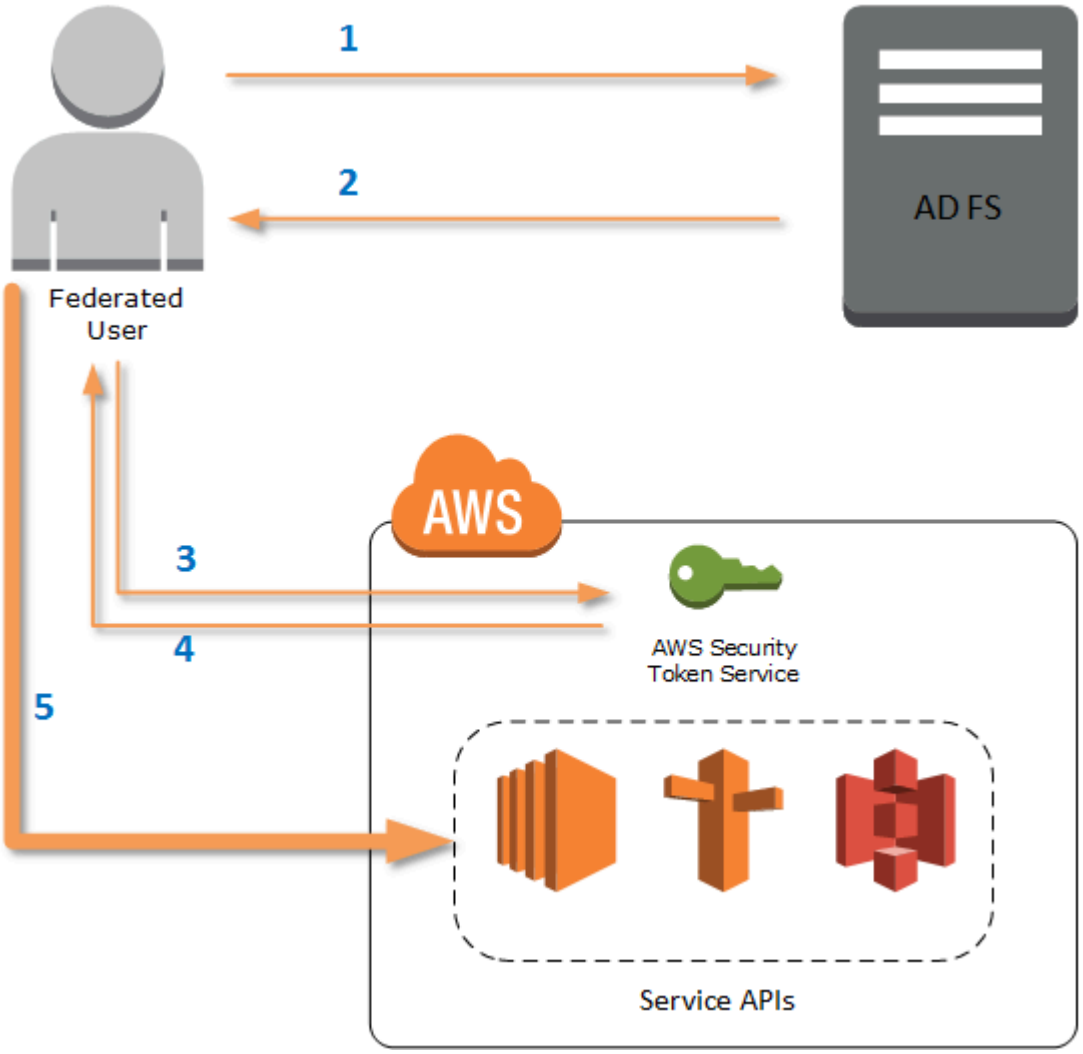
사전 조건

처음으로 SAML 지원을 사용하려면 먼저 다음 요소가 제 위치에 준비되어 있어야 합니다.

- 조직 자격 증명만 사용하여 콘솔 액세스를 위해 AWS 계정과 올바르게 통합된 연동 자격 증명 솔루션입니다. 특히 Active Directory 페더레이션 서비스에 대해이 작업을 수행하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 SAML 2.0 페더레이션 정보 및 블로그 게시물, Windows Active Directory, AD FS 및 SAML 2.0을 AWS 사용하여에 페더레이션 활성화를 참조하세요. 블로그 게시물에서는 AD FS 2.0을 설명하지만, AD FS 3.0을 실행 중인 경우라도 단계는 비슷합니다.
- 로컬 워크스테이션에 AWS Tools for PowerShell 설치된 버전 3.1.31.0 이상.

자격 증명 연동 사용자가 AWS 서비스 APIs에 대한 연동 액세스를 가져오는 방법

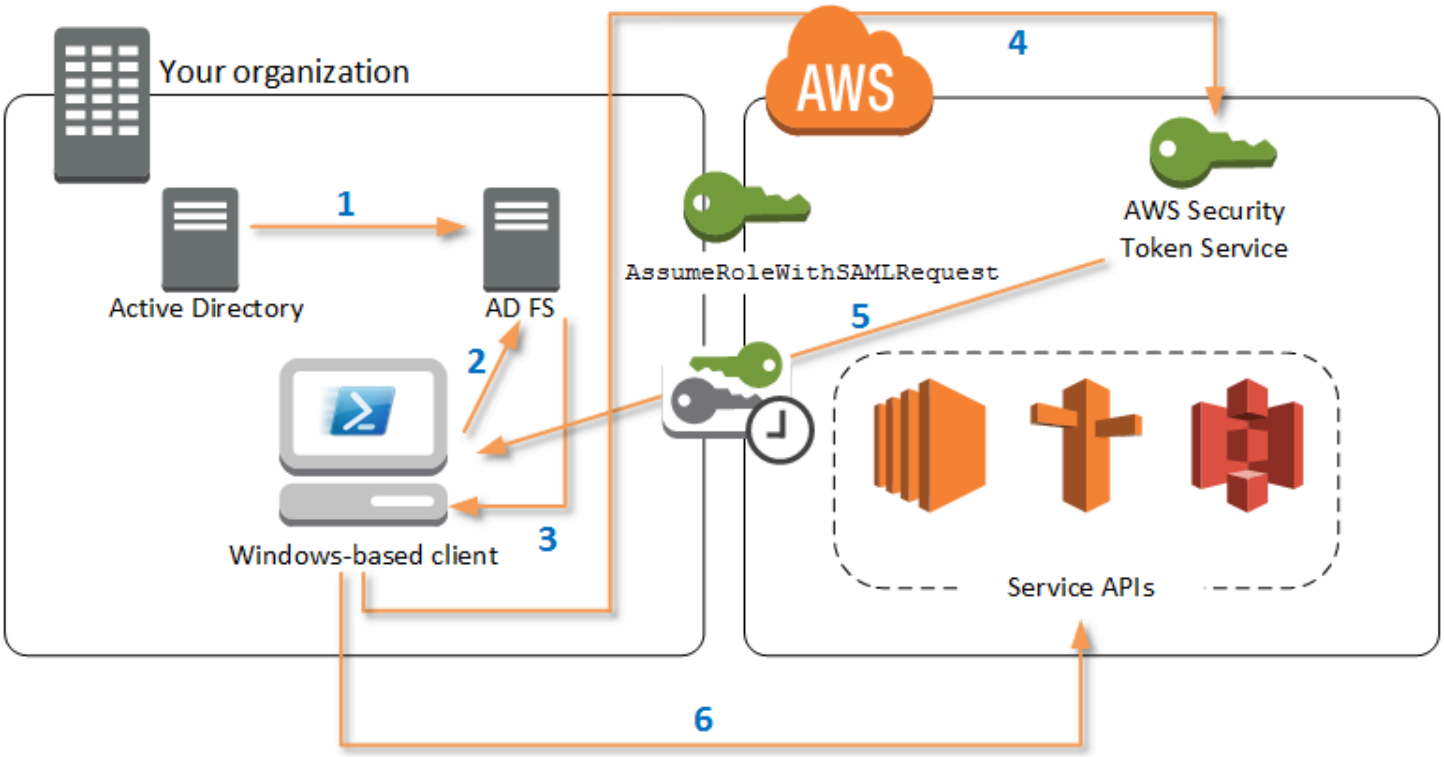
다음 프로세스는 AD FS에서 Active Directory(AD) 사용자를 페더레이션하여 AWS 리소스에 액세스하는 방법을 개략적으로 설명합니다.



1. 연동 사용자 컴퓨터의 클라이언트는 AD FS에 대해 인증합니다.
2. 인증에 성공하면 AD FS는 사용자에게 SAML 어설션을 보냅니다.
3. 사용자의 클라이언트는 SAML 페더레이션 요청의 일부로 AWS Security Token Service (STS)에 SAML 어설션을 전송합니다.
4. STS는 사용자가 수임할 수 있는 역할에 대한 AWS 임시 자격 증명이 포함된 SAML 응답을 반환합니다.
5. 사용자는 요청 시 임시 자격 증명을 포함하여 AWS 서비스 APIs에 액세스합니다 AWS Tools for PowerShell.

에서 SAML 지원 작동 방식 AWS Tools for PowerShell

이 섹션에서는 AWS Tools for PowerShell cmdlet이 사용자에 대한 SAML 기반 자격 증명 페더레이션 구성을 활성화하는 방법을 설명합니다.



1. AWS Tools for PowerShell 는 Windows 사용자의 현재 자격 증명을 사용하거나 사용자가 호출할 자격 증명이 필요한 cmdlet을 실행하려고 할 때 대화식으로 AD FS에 대해 인증합니다 AWS.
2. AD FS에서 사용자를 인증합니다.
3. AD FS는 어설션을 포함하는 SAML 2.0 인증 응답을 생성합니다. 어설션의 목적은 사용자에 대한 정보를 식별하고 제공하는 것입니다. AWS Tools for PowerShell 는 SAML 어설션에서 사용자의 승인된 역할 목록을 추출합니다.
4. AWS Tools for PowerShell 는 `AssumeRoleWithSAMLRequest` API 호출을 통해 요청된 역할의 Amazon 리소스 이름(ARN)을 포함한 SAML 요청을 STS로 전달합니다.
5. SAML 요청이 유효하면 STS는 AWS , `AccessKeyId`, `SecretAccessKey` 및 `SessionToken`을 포함하는 응답을 반환합니다. 이러한 자격 증명은 3,600초(1시간) 동안 유효합니다.
6. 이제 사용자는 사용자의 역할이 액세스할 권한이 있는 AWS 서비스 APIs로 작업할 수 있는 유효한 자격 증명을 갖게 됩니다.는 후속 AWS API 호출에 이러한 자격 증명을 AWS Tools for PowerShell 자동으로 적용하고 만료되면 자동으로 갱신합니다.

> **Note**
>
> 자격 증명이 만료되어 새 자격 증명이 필요하면 AWS Tools for PowerShell 은 AD FS에서 자동으로 재인증을 수행하여 이후 1시간 동안 유효한 새 자격 증명을 획득합니다. 도메인에 조인된 계정의 사용자의 경우, 이 프로세스가 자동으로 수행됩니다. 도메인에 가입되지 않은 계정의 경우는 사용자가 다시 인증하기 전에 자격 증명을 입력하도록 AWS Tools for PowerShell 프롬프트합니다.

PowerShell SAML 구성 Cmdlet 사용 방법

AWS Tools for PowerShell 에는 SAML 지원을 제공하는 두 개의 새로운 cmdlet이 포함되어 있습니다.

- `Set-AWSSamlEndpoint`는 AD FS 엔드포인트를 구성하고 표시 이름을 엔드포인트에 할당하며 원할 경우 엔드포인트의 인증 유형을 설명합니다.
- `Set-AWSSamlRoleProfile`은 `Set-AWSSamlEndpoint` cmdlet에 제공했던 표시 이름을 지정하여 AD FS 엔드포인트와 연결할 사용자 계정 프로파일을 만들거나 편집합니다. 각 역할 프로파일은 사용자에게 수행할 수 있는 권한이 부여된 단일 역할로 매핑됩니다.

  AWS 자격 증명 프로필과 마찬가지로 역할 프로필에 표시 이름을 할당합니다. `Set-AWSCredential` cmdlet과 동일한 표시 이름을 사용하거나 AWS 서비스 API를 호출하는 cmdlet의 `-ProfileName` 파라미터 값으로 사용할 수 있습니다. APIs

새 AWS Tools for PowerShell 세션을 엽니다. PowerShell 3.0 이상을 실행 중인 경우, cmdlet 중 하나를 실행할 때 AWS Tools for PowerShell 모듈을 자동으로 가져옵니다. PowerShell 2.0을 실행하는 경우 다음 예제와 같이 ```Import-Module`` cmdlet을 실행하여 모듈을 수동으로 가져와야 합니다.

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

Set-AWSSamlEndpoint 및 **Set-AWSSamlRoleProfile** cmdlet을 실행하는 방법

1. 먼저 AD FS 시스템에 대한 엔드포인트 설정을 구성합니다. 가장 간단한 방법은 이 단계처럼 변수에 엔드포인트를 저장하는 것입니다. 계정 ID 및 AD FS 호스트 이름 자리 표시자를 자신의 고유 계정 ID와 AD FS 호스트 이름으로 대체해야 합니다. `Endpoint` 파라미터에 AD FS 호스트 이름을 지정합니다.

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

2. 엔드포인트 설정을 만들려면 Set-AWSSamlEndpoint cmdlet을 실행하여 AuthenticationType 파라미터에 대해 올바른 값을 지정합니다. 유효한 값으로는 Basic, Digest, Kerberos, Negotiate 및 NTLM이 있습니다. 이 파라미터를 지정하지 않는 경우 기본값은 Kerberos입니다.

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

   이 cmdlet은 -StoreAs 파라미터를 사용하여 지정한 표시 이름을 반환하므로 다음 줄에서 Set-AWSSamlRoleProfile을 실행할 때 이 이름을 사용할 수 있습니다.

3. 이제 Set-AWSSamlRoleProfile cmdlet을 실행하여 AD FS 자격 증명 공급자를 대상으로 인증을 수행하고 사용자에게 수행할 수 있는 권한이 부여된 역할 세트(SAML 어설션에서)를 가져올 수 있습니다.

   Set-AWSSamlRoleProfile cmdlet은 반환되는 역할 세트를 사용하여 사용자에게 지정된 프로파일과 연결할 역할을 선택하라는 메시지를 표시하거나, 파라미터에 제공된 역할 데이터가 존재하는지 확인합니다(없는 경우 사용자에게 선택하라는 메시지가 표시됨). 사용자에게 한 역할에만 권한이 부여된 경우 cmdlet은 사용자에게 메시지를 표시하지 않고 해당 역할을 프로파일과 자동으로 연결합니다. 도메인에 조인된 사용을 위해 프로파일을 설정할 때는 자격 증명을 제공할 필요가 없습니다.

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

   또는 non-domain-joined 계정의 경우 다음 줄과 같이 Active Directory 자격 증명을 제공한 다음 사용자가 액세스할 수 있는 AWS 역할을 선택할 수 있습니다. 이는 조직 내에서 역할을 구분하기 위해 여러 Active Directory 사용자 계정이 있는 경우(예를 들면 관리 기능)에 유용합니다.

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

4. 어느 경우든, Set-AWSSamlRoleProfile cmdlet은 프로파일에 저장되어야 할 역할을 선택하라는 메시지를 표시합니다. 다음 예에서는 ADFS-Dev와 ADFS-Production이라는 두 가지 사용 가능한 역할을 보여 줍니다. IAM 역할은 AD FS 관리자에 의해 AD 로그인 자격 증명과 연결됩니다.

```
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"):
```

   또는 RoleARN, PrincipalARN 및 NetworkCredential 파라미터(선택 사항)를 입력하여 프롬프트 메시지 없이 역할을 지정할 수 있습니다. 지정된 역할이 인증 시 반환된 어설션에 나열되어 있지 않으면 사용자에게 사용 가능한 역할 중에서 선택하라는 메시지가 표시됩니다.

```
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. 다음 코드에서처럼 StoreAllRoles 파라미터를 추가하여 단일 명령을 통해 모든 역할에 대한 프로파일을 만들 수 있습니다. 역할 이름이 프로파일 이름으로 사용됩니다.

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

역할 프로파일을 사용하여 AWS 자격 증명이 필요한 Cmdlet을 실행하는 방법

AWS 자격 증명이 필요한 cmdlet을 실행하려면 AWS 공유 자격 증명 파일에 정의된 역할 프로파일을 사용할 수 있습니다. 프로파일에 설명된 역할에 대한 임시 AWS 자격 증명을 자동으로 가져오려면 역할 프로파일의 이름을에 입력합니다Set-AWSCredential(또는의 ProfileName 파라미터 값으로 제공 AWS Tools for PowerShell).

한 번에 역할 프로파일 하나만 사용할 수 있지만 셸 세션 내에서 프로파일 간에 전환할 수 있습니다. Set-AWSCredential cmdlet은 인증하지 않고 실행 시 자체적으로 자격 증명을 가져옵니다. 이 cmdlet은 사용자가 지정된 역할 프로파일을 사용하고자 한다고 기록합니다. AWS 자격 증명이 필요한 cmdlet을 실행할 때까지는 인증이나 자격 증명 요청이 발생하지 않습니다.

이제 SAMLDemoProfile 프로파일에서 얻은 임시 AWS 자격 증명을 사용하여 AWS 서비스 APIs. 다음 단원에서는 역할 프로파일 사용법 예를 보여 줍니다.

예제 1: **Set-AWSCredential**을 사용하여 기본 역할 지정

이 예제에서는를 사용하여 AWS Tools for PowerShell 세션의 기본 역할을 설정합니다`Set-AWSCredential`. 그러고 나면 자격 증명이 필요하고 지정된 역할에 의해 권한이 부여된 cmdlet을 실행할 수 있습니다. 이 예제는 미국 서부(오레곤) 리전에서 `Set-AWSCredential` cmdlet을 사용하여 지정한 프로파일과 연결되어 있는 모든 Amazon Elastic Compute Cloud 인스턴스를 나열합니다.

```
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}
```

예제 2: PowerShell 세션 중 역할 프로파일 변경

이 예시에서는 `SAMLDemoProfile` 프로파일과 연결된 역할의 AWS 계정에서 사용 가능한 모든 Amazon S3 버킷을 나열합니다. 이 예제는 AWS Tools for PowerShell 세션의 앞부분에서 다른 프로파일을 사용했을 수도 있지만, 이를 지원하는 cmdlet이 있는 `-ProfileName` 파라미터에 대해 다른 값을 지정하여 프로파일을 변경할 수 있음을 보여줍니다. 이 작업은 PowerShell 명령줄을 통해 Amazon S3을 관리하는 관리자에게 일반적인 작업입니다.

```
PS > Get-S3Bucket -ProfileName SAMLDemoProfile

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

`Get-S3Bucket` cmdlet은 `Set-AWSSamlRoleProfile` cmdlet을 실행하여 생성된 프로파일의 이름을 지정합니다. 이 명령은 (예를 들면 `Set-AWSCredential` cmdlet을 실행하여) 세션의 조기에 역할 프로파일을 설정했으며 `Get-S3Bucket` cmdlet에 대해 다른 역할 프로파일을 사용하려는 경우에 유용

할 수 있습니다. 프로파일 관리자는 임시 자격 증명을 `Get-S3Bucket` cmdlet에 사용할 수 있도록 설정합니다.

자격 증명은 1시간(STS에서 적용되는 제한)이 경과된 후 만료되지만, AWS Tools for PowerShell 는 도구가 현재 자격 증명이 만료되었음을 감지할 때 새 SAML 어설션을 요청하여 자격 증명을 자동으로 새로 고칩니다.

도메인에 조인된 사용자의 경우, 인증 시 현재 사용자의 Windows 자격 증명이 사용되므로 이 프로세스가 중단 없이 발생합니다. non-domain-joined 사용자 계정의 경우 사용자 암호를 요청하는 PowerShell 자격 증명 프롬프트를 AWS Tools for PowerShell 표시합니다. 사용자는 사용자를 재인증하고 새 어설션을 가져오는 데 사용되는 자격 증명을 제공합니다.

예제 3: 리전의 인스턴스 가져오기

다음 예제는 `ADFS-Production` 프로파일에서 사용된 계정과 연결된 아시아 태평양(시드니) 리전의 모든 Amazon EC2 인스턴스를 나열합니다. 이 명령은 리전의 모든 Amazon EC2 인스턴스를 반환하는 유용한 명령입니다.

```
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
```

추가 읽기 자료

연동 API 액세스를 구현하는 방법에 대한 일반적인 정보는 How to Implement a General Solution for Federated API/CLI Access Using SAML 2.0을 참조하십시오.

지원 질문이나 의견은 PowerShell 스크립팅용 AWS 개발자 포럼 또는 .NET Development를 참조하세요.

Cmdlet 검색 및 별칭

이 단원에서는 AWS Tools for PowerShell에서 지원하는 서비스 목록을 표시하는 방법, 이러한 서비스를 지원하기 위해 AWS Tools for PowerShell에서 제공하는 cmdlet 세트를 표시하는 방법, 이러한 서비스에 액세스하기 위해 대체 cmdlet 이름(별칭)을 찾는 방법을 설명합니다.

Cmdlet 검색

모든 AWS 서비스 작업(또는 API) 은 각 서비스에 대한 API 참조 안내서에 설명되어 있습니다. 예를 들면 IAM API 참조를 참조하세요. 대부분의 경우 AWS 서비스 API와 AWS PowerShell cmdlet 간에는 일대일 통신이 있습니다. AWS 서비스 API 이름에 해당하는 cmdlet 이름을 가져오려면 `-ApiOperation` 파라미터 및 AWS 서비스 API 이름을 지정하여 AWS `Get-AWSCmdletName` cmdlet을 실행합니다. 예를 들어 사용 가능한 `DescribeInstances` AWS 서비스 API를 기반으로 하는 모든 cmdlet 이름을 가져오려면 다음 명령을 실행합니다.

```
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` 매개 변수가 기본 매개 변수이므로 매개 변수 이름을 생략할 수 있습니다. 다음 예제는 이전 예제와 동일합니다.

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

API와 서비스의 이름을 둘 다 알고 있는 경우 `-Service` 파라미터와 함께 cmdlet 명사 접두사나 AWS 서비스 이름의 일부를 포함시킬 수 있습니다. 예를 들면 Amazon EC2의 cmdlet 명사 접두사는 `EC2`입니다. Amazon EC2 서비스에서 `DescribeInstances` API에 해당하는 cmdlet 이름을 가져오려면 다음 명령 중 하나를 실행합니다. 이들은 모두 동일한 출력의 결과입니다.

```
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
```

이러한 명령의 파라미터 값은 대/소문자를 구분합니다.

원하는 AWS 서비스 API나 AWS 서비스의 이름을 모르는 경우에는 `-ApiOperation` 파라미터와 함께 일치하는 패턴 및 `-MatchWithRegex` 파라미터를 사용할 수 있습니다. 예를 들어 `SecurityGroup`을 포함하는 사용 가능한 cmdlet 이름을 모두 가져오려면 다음 명령을 실행합니다.

```
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 서비스의 이름은 알지만 AWS 서비스 API의 이름을 모르는 경우에는 검색 범위를 단일 서비스로 지정할 수 있도록 `-MatchWithRegex` 파라미터와 `-Service` 파라미터를 모두 포함시킵니다. 예를 들어 Amazon EC2 서비스에서만 `SecurityGroup`을 포함하는 모든 cmdlet 이름을 모두 가져오려면 다음 명령을 실행합니다.

```
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) 명령의 이름을 알고 있는 경우에는 `-AwsCliCommand` 매개 변수와 원하는 AWS CLI 명령 이름을 사용하여 동일한 API를 기반으로 하는 cmdlet 이름을 가져올 수 있습니다. 예를 들어 Amazon EC2 서비스에서 `authorize-security-group-ingress` AWS CLI 명령 호출에 해당하는 cmdlet 이름을 가져오려면 다음 명령을 실행합니다.

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

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

`Get-AWSCmdletName` cmdlet은 AWS CLI 명령 이름만 있으면 서비스와 AWS API를 식별할 수 있습니다.

Tools for PowerShell Core의 모든 cmdlet의 목록을 가져오려면 다음 예제에 나와 있는 것처럼 PowerShell `Get-Command` cmdlet을 실행합니다.

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

`-Module AWSPowerShell`과 동일한 명령을 실행하여 AWS Tools for Windows PowerShell의 cmdlet을 확인할 수 있습니다.

`Get-Command` cmdlet은 cmdlet 목록을 알파벳순으로 생성합니다. 기본적으로 목록은 PowerShell 명사가 아닌 PowerShell 동사로 정렬됩니다.

그와 달리 서비스별로 결과를 정렬하려면 다음 명령을 실행합니다.

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

`Get-Command` cmdlet에 의해 반환되는 cmdlet을 필터링하려면 출력을 PowerShell `Select-String` cmdlet에 파이프합니다. 예를 들어 AWS 리전에서 사용하는 cmdlet 세트를 보려면 다음 명령을 실행합니다.

```
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
```

또한 cmdlet 명사의 서비스 접두사를 필터링하여 특정 서비스에 대한 cmdlet을 찾을 수도 있습니다. 사용 가능한 서비스 접두사 목록을 보려면 `Get-AWSPowerShellVersion -ListServiceVersionInfo`을 실행합니다. 다음 예제는 Amazon CloudWatch Events 서비스를 지원하는 cmdlet을 반환합니다.

```
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
```

Cmdlet 이름 지정 및 별칭

AWS Tools for PowerShell에서 각 서비스용 cmdlet은 해당 서비스용 AWS SDK에서 제공하는 메서드를 기반으로 합니다. 하지만 PowerShell의 이름 지정 규칙 때문에 cmdlet의 이름이 해당 API 호출이나 기반이 되는 메서드의 이름과 다를 수 있습니다. 예를 들어 `Get-EC2Instance` cmdlet은 Amazon EC2 `DescribeInstances` 메서드를 기반으로 합니다.

경우에 따라, cmdlet 이름이 메서드 이름과 비슷하지만, 실제로는 다른 기능을 수행할 수도 있습니다. 예를 들어 Amazon S3`GetObject` 메서드는 Amazon S3 객체를 검색합니다. 그러나, `Get-S3Object` cmdlet은 객체 자체보다 Amazon S3 객체에 대한 정보를 반환합니다.

```
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 PowerShell를 사용하여 S3 객체를 검색하려면 `Read-S3Object` cmdlet을 실행합니다.

```
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**
>
> cmdlet에 해당하는 AWS SDK API의 이름은 AWS cmdlet의 해당 cmdlet 도움말에 나와 있습니다.
> 표준 PowerShell 동사와 예상 의미에 대한 자세한 내용은 PowerShell 명령에 대해 승인된 동사를 참조하십시오.

`Remove` 동사를 사용하는 모든 AWS cmdlet과 `-Terminate` 파라미터를 추가할 때 `Stop-EC2Instance` cmdlet은 계속하기 전에 확인 메시지를 표시합니다. 확인 메시지를 건너뛰려면 명령에 `-Force` 파라미터를 추가합니다.

> **Important**
>
> AWS cmdlet은 `-WhatIf` 스위치를 지원하지 않습니다.

에일리어스

AWS Tools for PowerShell을 설정하면 다수의 AWS cmdlet에 대한 별칭을 포함하는 별칭 파일이 설치됩니다. 이러한 별칭은 cmdlet 이름보다 더 직관적입니다. 예를 들어 서비스 이름과 AWS SDK 메서드 이름은 일부 별칭의 PowerShell 동사와 명사를 대체합니다. 예를 들면 `EC2-DescribeInstances` 별칭이 있습니다.

그 밖의 별칭에는, 표준 PowerShell 표기 규칙을 따르지는 않지만 실제 작업을 더 잘 설명할 수 있는 동사가 사용됩니다. 예를 들면 별칭 파일은 `Get-S3Content` 별칭을 `Read-S3Object` cmdlet으로 매핑합니다.

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

별칭 파일은 AWS Tools for PowerShell 설치 디렉터리에 있습니다. 별칭을 환경에 로드하려면 파일을 도트 소싱(dot-sourcing)합니다. 다음은 Windows용 예입니다.

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

Linux 또는 macOS 셸의 경우 다음과 같이 보일 수 있습니다.

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

모든 AWS Tools for PowerShell 별칭을 표시하려면 다음 명령을 실행합니다. 이 명령은 PowerShell ? cmdlet의 `Where-Object` 별칭과 `Source` 속성을 사용하여 `AWSPowerShell.NetCore` 모듈에서 나온 별칭만 필터링합니다.

```
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

...
```

이 파일에 사용자 지정 별칭을 추가하려면 PowerShell의 `$MaximumAliasCount` 기본 설정 변수를 5500보다 큰 값으로 늘려야 할 수 있습니다. 기본값은 4096이며, 이 값을 최대 32768까지 높일 수 있습니다. 이렇게 하려면 다음을 실행합니다.

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

변경 사항이 성공적으로 적용되었는지 확인하려면 변수 이름을 입력하여 현재 값을 표시합니다.

```
PS > $MaximumAliasCount
32768
```

AWS Tools for PowerShell의 파이프라이닝, 출력 및 반복

파이프라이닝

PowerShell은 사용자가 한 cmdlet의 출력을 다음 cmdlet의 입력으로 보내는 파이프라인에 cmdlet을 연결하도록 권장합니다. 다음 예에서는 AWS Tools for PowerShell을 사용하여 이 작업을 수행하는 방법을 보여줍니다. 명령은 현재 기본 리전의 모든 Amazon EC2 인스턴스를 가져온 다음 중지합니다.

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

Cmdlet 출력

파이프라이닝을 더 잘 지원하기 위해 AWS SDK for .NET의 응답에서 일부 데이터가 기본적으로 삭제될 수 있습니다. AWS Tools for PowerShell cmdlet의 출력은 방출된 컬렉션 객체의 `Note` 속성으로서 서비스 응답 및 결과 인스턴스를 포함하도록 더 이상 변경되지 않습니다. 대신에, 단일 모음을 출력으로 내보내는 이러한 호출의 경우, 모음이 이제 PowerShell 파이프라인에 열거됩니다. 다시 말해서 연결할 수 있는 모음 객체를 포함하지 않기 때문에 SDK 응답과 결과 데이터가 파이프라인에 존재할 수 없다는 뜻입니다.

대부분의 사용자에게는 이 데이터가 필요하지 않겠지만, cmdlet에서 수행된 기본 AWS 서비스 호출로부터 보내고 받은 내용을 정확히 알 수 있으므로 진단 목적으로 유용할 수 있습니다. AWS Tools for

PowerShell V4부터 cmdlet은 `-Select *` 파라미터와 인수를 사용하여 전체 서비스 응답을 반환할 수 있습니다.

> **Note**
>
> V4 이전의 AWS Tools for PowerShell 버전에서는 AWS cmdlet 간접 호출 및 각 간접 호출에 대해 수신된 서비스 응답의 기록을 유지하는 `$AWSHistory`라는 세션 변수가 도입되었습니다. Tools for PowerShell의 V4에서는 이 세션 변수가 전체 서비스 응답을 반환하는 데 사용될 수 있는 `-Select *` 파라미터 및 인수를 위해 더 이상 사용되지 않습니다. 이 파라미터는 이 주제에 설명되어 있습니다.
>
> > **Note**
> >
> > 이 시험판 설명서는 미리 보기 릴리스의 기능에 관한 것입니다. 변경될 수 있습니다.
>
> `$AWSHistory` 변수는 AWS Tools for PowerShell의 V5에서 제거됩니다. 자세한 내용은 블로그 게시물 AWS Tools for PowerShell의 예정된 메이저 버전 5 공지를 참조하세요.

응답의 모든 데이터를 반환하는 방법을 설명하려면 다음 예제를 고려하세요.

첫 번째 예제에서는 Amazon S3 버킷 목록을 반환합니다. 이는 기본 설정 동작입니다.

```
PS > Get-S3Bucket

CreationDate           BucketName
------------           ----------
9/22/2023 10:54:35 PM  amzn-s3-demo-bucket1
9/22/2023 11:04:37 AM  amzn-s3-demo-bucket2
9/22/2023 12:54:34 PM  amzn-s3-demo-bucket3
```

두 번째 예제에서는 AWS SDK for .NET 응답 객체를 반환합니다. `-Select *`가 지정되었으므로 출력에는 `Buckets` 속성의 버킷 모음을 포함하는 전체 API 응답이 포함됩니다. 이 예제에서는 `Format-List` cmdlet이 반드시 필요한 것은 아니지만 모든 속성이 표시되도록 하기 위해 존재합니다.

```
PS > Get-S3Bucket -Select * | Format-List

LoggedAt          : 10/1/2023 9:45:52 AM
Buckets           : {amzn-s3-demo-bucket1, amzn-s3-demo-bucket2,
```

```
                    amzn-s3-demo-bucket3}
Owner             : Amazon.S3.Model.Owner
ContinuationToken :
ResponseMetadata  : Amazon.Runtime.ResponseMetadata
ContentLength     : 0
HttpStatusCode    : OK
```

페이징된 데이터를 통한 반복

다음 섹션에서는 가능한 다양한 유형의 반복에 대해 설명합니다.

자동 반복

지정된 호출에 대해 반환된 객체의 기본 최대 수를 부과하거나 페이지 지정 가능한 결과 세트를 지원하는 서비스 API의 경우 대부분의 cmdlet은 자동 반복을 구현하여 기본 동작인 "page-to-completion"을 활성화합니다. 이 시나리오 에서 cmdlet은 전체 데이터 세트를 파이프라인으로 반환하기 위해 사용자를 대신하여 필요한 만큼 호출합니다.

`Get-S3Object` cmdlet을 사용하는 다음 예제에서는 `$result` 변수가 `amzn-s3-demo-bucket1`이라는 버킷(잠재적으로 매우 큰 데이터 세트)의 모든 키에 대한 `S3Object` 인스턴스를 포함합니다.

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1
```

다음 예제에서는 자동 반복 중 각 페이지의 결과 수를 기본값인 1000에서 500으로 줄입니다. 이 예제에서는 각 호출에 대해 반환되는 결과가 절반에 불과하므로 자동 반복 직접 호출을 두 배로 수행합니다.

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500
```

> **Note**
>
> AWS Tools for PowerShell V4에서 호출된 작업에 대한 일부 cmdlet은 자동 반복을 구현하지 않습니다. cmdlet에 다음 섹션에서 설명하는 `-NoAutoIteration` 파라미터가 없는 경우 자동 반복을 구현하지 않습니다.

자동 반복 비활성화

Tools for PowerShell이 데이터의 첫 페이지만 반환하도록 하려면 `-NoAutoIteration` 파라미터를 추가하여 데이터의 추가 페이지가 반환되지 않도록 할 수 있습니다.

다음 예제에서는 -NoAutoIteration 및 -MaxKey 파라미터를 사용하여 반환되는 S3Object 인스턴스의 수를 버킷에서 처음 발견되는 500개 이하로 제한합니다.

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -
NoAutoIteration
```

사용 가능했지만 반환되지 않은 데이터가 더 있는지 확인하려면 -Select * 파라미터와 인수를 사용하고 다음 토큰 속성에 값이 있는지 확인합니다.

다음 예제에서는 버킷에 500를 초과하는 객체가 있는 경우 $true를 반환하고 그렇지 않은 경우 $false를 반환합니다.

```
PS > $result = Get-S3Object -BucketName amzn-s3-demo-bucket1 -MaxKey 500 -
NoAutoIteration -Select *
PS > $null -eq $result.NextMarker
```

> **Note**
>
> 다음 토큰 응답 속성과 cmdlet 파라미터의 이름은 cmdlet마다 다릅니다. 자세한 내용은 각 cmdlet에 대한 도움말 설명서를 참조하세요.

수동 반복

다음 예제에서는 각 반복 후 조건을 평가하는 do 루프를 사용하여 버킷의 모든 S3 객체를 반환합니다. do 루프는 Get-S3Object가 $result.NextMarker를 $null로 설정할 때까지 반복을 수행하여 더 이상 페이징된 데이터가 남아 있지 않음을 나타냅니다. 루프의 출력은 $s3Objects 변수에 할당됩니다.

```
$s3Objects = do
{
    $splatParams = @{
        BucketName = 'amzn-s3-demo-bucket1'
        MaxKey = 500
        Marker = $result.NextMarker
        NoAutoIteration = $true
        Select = '*'
    }
    $result = Get-S3Object @splatParams
```

```
    $result.S3Objects
}
while ($null -ne $result.NextMarker)
```

이 예제에서는 PowerShell 스플래팅을 사용하여 파라미터와 인수를 인라인으로 선언하여 발생하는 긴 코드 줄을 방지합니다.

보안 인증 정보 및 프로파일 확인

자격 증명 검색 순서

명령을 실행하면 AWS Tools for PowerShell는 다음 순서로 자격 증명을 검색합니다. 사용 가능한 자격 증명을 찾으면 중지됩니다.

1. 명령줄에 파라미터로 내장된 리터럴 자격 증명입니다.

   가급적이면 명령줄에 리터럴 자격 증명을 추가하기 보다는 프로파일을 사용하는 것이 좋습니다.
2. 지정된 프로파일 이름 또는 프로파일 위치.
   - 프로파일 이름만 지정할 경우 이 명령은 AWS SDK 저장소의 지정된 프로파일을 찾고 여기에 지정된 프로파일이 없으면 기본 위치의 AWS 공유 자격 증명 파일에서 지정된 프로파일을 찾습니다.
   - 프로파일 위치만 지정하는 경우, 이 명령은 해당 자격 증명 파일에서 `default` 프로파일을 찾습니다.
   - 이름과 위치를 모두 지정하는 경우, 이 명령은 해당 자격 증명 파일에서 지정된 프로파일을 찾습니다.

   지정된 프로파일이나 위치가 없으면 명령에서 예외가 발생합니다. 프로파일이나 위치를 지정하지 않은 경우에만 검색이 다음 단계로 이동합니다.
3. `-Credential` 파라미터에서 지정된 자격 증명.
4. 세션 프로파일(존재하는 경우).
5. 기본 프로파일(다음 순서대로).

   a. `default` SDK 저장소의 AWS 프로파일.

   b. `default` 공유 자격 증명 파일의 AWS 프로파일.

   c. `AWS PS Default` SDK 저장소의 AWS 프로파일.
6. IAM 역할을 사용하도록 구성된 Amazon EC2 인스턴스에서 명령이 실행 중인 경우, 인스턴스 프로파일로부터 액세스한 EC2 인스턴스의 임시 자격 증명입니다.

Amazon EC2 인스턴스에 IAM 역할 사용에 대한 자세한 내용은 AWS SDK for .NET 단원을 참조하세요.

이 검색을 통해 지정된 자격 증명을 찾지 못한 경우 명령에서 예외가 발생합니다.

사용자 및 역할에 대한 추가 정보

AWS에서 Tools for PowerShell 명령을 실행하려면 작업에 적합한 사용자, 권한 집합 및 서비스 역할의 조합이 필요합니다.

만드는 특정 사용자, 권한 집합 및 서비스 역할과 이를 사용하는 방식은 요구 사항에 따라 달라집니다. 다음은 사용 이유 및 생성 방법에 대한 몇 가지 추가 정보입니다.

사용자 및 권한 집합

장기 보안 인증 정보가 있는 IAM 사용자 계정을 사용하여 AWS 서비스에 액세스할 수는 있지만 이는 더 이상 모범 사례가 아니므로 피해야 합니다. 개발 중에도 AWS IAM Identity Center에서 사용자 및 권한 집합을 만들고 ID 소스에서 제공하는 임시 보안 인증 정보를 사용하는 것이 가장 좋은 방법입니다.

개발 시에는 직접 생성했거나 도구 인증 구성에서 부여받은 사용자를 사용할 수 있습니다. 적절한 AWS Management Console 권한이 있는 경우 해당 사용자에 대해 최소 권한으로 다양한 권한 집합을 생성하거나 개발 프로젝트용으로 특별히 새 사용자를 생성하여 최소 권한으로 권한 집합을 제공할 수도 있습니다. 어떤 방법을 선택할지는 상황에 따라 달라집니다.

이러한 사용자 및 권한 세트와 생성 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 인증 및 액세스 및 AWS IAM Identity Center 사용 설명서의 시작을 참조하세요.

서비스 역할

사용자를 대신하여 AWS 서비스에 액세스하도록 AWS 서비스 역할을 설정할 수 있습니다. 이러한 유형의 액세스는 여러 사람이 원격으로 애플리케이션을 실행하는 경우(예: 이러한 목적으로 생성한 Amazon EC2 인스턴스)에 적합합니다.

서비스 역할을 만드는 프로세스는 상황에 따라 다르지만 기본적으로 다음과 같습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 역할을 선택한 다음 역할 생성을 선택합니다.

3. AWS 서비스를 선택하고, EC2(예제)를 찾아 선택한 다음 EC2 사용 사례(예제)를 선택합니다.
4. 다음을 선택하고 애플리케이션에서 사용할 AWS 서비스에 대한 적절한 정책을 선택합니다.

> **Warning**
>
> 이 정책은 계정의 거의 모든 항목에 대한 읽기 및 쓰기 권한을 허용하므로 AdministratorAccess 정책을 선택하지 않습니다.

5. 다음을 선택합니다. 역할 이름, 설명 및 원하는 태그를 입력합니다.

   태그에 대한 정보는 IAM 사용 설명서의 AWS 리소스 태그를 사용하여 액세스 제어에서 찾을 수 있습니다.
6. 역할 생성을 선택합니다.

IAM 사용 설명서의 IAM ID(사용자, 그룹 및 역할)에서 IAM 역할에 대한 개략적인 정보를 찾을 수 있습니다. 역할에 대한 자세한 내용은 IAM 역할 주제를 참조하세요.

레거시 보안 인증 사용

이 섹션의 주제에서는 AWS IAM Identity Center를 사용하지 않고 장기 또는 단기 보안 인증 정보를 사용하는 방법에 대한 정보를 제공합니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

> **Note**
>
> 이러한 주제의 정보는 단기 또는 장기 보안 인증 정보를 수동으로 획득하고 관리해야 하는 상황을 위한 것입니다. 단기 및 장기 보안 인증 정보에 대한 자세한 내용은 AWS 및 도구 참조 가이드의 다른 인증 방법을 참조하세요.
> 보안 모범 사례를 보려면에 설명된 AWS IAM Identity Center대로를 사용합니다도구 인증 구성.

보안 인증에 대한 중요 경고 및 지침

보안 인증에 대한 경고

- 계정의 루트 자격 증명을 사용하여 AWS 리소스에 액세스하지 마십시오. 이 보안 인증은 계정 액세스에 제한이 없고 취소하기 어렵습니다.
- 금지 사항. 명령이나 스크립트에 리터럴 액세스 키나 보안 인증 정보를 넣지 않습니다. 그렇게 하면 보안 인증 정보가 실수로 노출될 위험이 있습니다.
- 공유 AWS `credentials` 파일에 저장된 모든 자격 증명은 일반 텍스트로 저장됩니다.

보안 인증 정보를 안전하게 관리하기 위한 추가 지침

자격 AWS 증명을 안전하게 관리하는 방법에 대한 일반적인 설명은의 AWS 보안 자격 증명AWS 일반 참조과 IAM 사용 설명서의 보안 모범 사례 및 사용 사례를 참조하세요. 해당 설명과 더불어 다음 사항을 고려하세요.

- AWS 루트 사용자 보안 인증 정보를 사용하는 대신 IAM Identity Center에 사용자 등 추가 사용자를 만들고 해당 보안 인증 정보를 사용합니다. 다른 사용자의 보안 인증 정보는 필요한 경우 또는 일시적인 경우 해지할 수 있습니다. 또한 각 사용자에게 특정 리소스 및 작업에만 액세스할 수 있도록 정책을 적용하여 최소 권한 권한을 유지할 수 있습니다.

- Amazon EC2 Container Service(Amazon ECS) 작업에 작업용 IAM 역할을 사용하세요.

- Amazon EC2 인스턴스에서 실행 중인 애플리케이션에 IAM 역할을 사용하십시오.

주제

- AWS 자격 증명 사용
- 의 공유 자격 증명 AWS Tools for PowerShell

AWS 자격 증명 사용

각 AWS Tools for PowerShell 명령에는 해당 웹 서비스 요청에 암호화 방식으로 서명하는 데 사용되는 AWS 자격 증명 세트가 포함되어야 합니다. 명령별로, 세션별로, 또는 모든 세션에 대해 자격 증명을 지정할 수 있습니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

> **Note**
>
> 이 주제의 정보는 단기 또는 장기 보안 인증 정보를 수동으로 획득하고 관리해야 하는 상황을 위한 것입니다. 단기 및 장기 보안 인증 정보에 대한 자세한 내용은 AWS 및 도구 참조 가이드의 다른 인증 방법을 참조하세요.
> 보안 모범 사례를 보려면에 설명된 AWS IAM Identity Center대로를 사용합니다도구 인증 구성.

자격 증명이 노출되지 않도록 하기 위해 명령에 리터럴 자격 증명을 추가하지 않는 것이 좋습니다. 대신에, 사용할 각 자격 증명 세트에 대한 프로파일을 만들고 두 자격 증명 저장소 중 하나에 프로파일을 저장합니다. 명령에서 이름을 기준으로 올바른 프로파일을 지정하면 AWS Tools for PowerShell 에서 관련 자격 증명을 검색합니다. 자격 AWS 증명을 안전하게 관리하는 방법에 대한 일반적인 설명은의 AWS 액세스 키 관리 모범 사례를 참조하세요Amazon Web Services 일반 참조.

> **Note**
>
> 자격 증명을 가져오고를 사용하려면 AWS 계정이 필요합니다 AWS Tools for PowerShell. AWS 계정을 생성하려면 AWS Account Management 참조 안내서의 시작하기: 처음 AWS 사용하시나요?를 참조하세요.

주제

- 자격 증명 저장소 위치
- 프로파일 관리
- 자격 증명 지정
- 자격 증명 검색 순서
- 의 자격 증명 처리 AWS Tools for PowerShell Core

자격 증명 저장소 위치

는 다음 두 가지 자격 증명 저장소 중 하나를 사용할 AWS Tools for PowerShell 수 있습니다.

- 자격 증명을 암호화하여 홈 폴더에 저장하는 AWS SDK 스토어입니다. Windows에서 이 저장소는 `C:\Users\username\AppData\Local\AWSToolkit\RegisteredAccounts.json` 위치에 있습니다.

  AWS SDK for .NET 및 Toolkit for Visual Studio에서도 AWS SDK 저장소를 사용할 수 있습니다.
- 홈 폴더에 있지만 자격 증명 파일을 일반 텍스트로 저장하는 공유 자격 증명 파일.

  기본적으로 자격 증명 파일은 다음에 저장됩니다.

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

  AWS SDKs 및 도 자격 증명 파일을 사용할 AWS Command Line Interface 수 있습니다. AWS 사용자 컨텍스트 외부에서 스크립트를 실행하는 경우 자격 증명이 포함된 파일을 모든 사용자 계정(로컬 시스템 및 사용자)이 자격 증명에 액세스할 수 있는 위치로 복사해야 합니다.

프로파일 관리

프로파일을 사용하면 다양한 자격 증명 세트를 참조할 수 있습니다 AWS Tools for PowerShell. AWS Tools for PowerShell cmdlet을 사용하여 AWS SDK 스토어에서 프로필을 관리할 수 있습니다. 또한 Toolkit for Visual Studio를 사용하거나, AWS SDK for .NET을(를) 사용하여 프로그래밍 방식으로 AWS SDK 저장소를 관리할 수도 있습니다. 자격 증명 파일에서 프로필을 관리하는 방법에 대한 지침은 AWS 액세스 키 관리 모범 사례를 참조하세요.

새 프로파일 추가

AWS SDK 스토어에 새 프로필을 추가하려면 명령을 실행합니다`Set-AWSCredential`. 액세스 키와 보안 키를 지정한 프로파일 이름 아래의 기본 자격 증명 파일에 저장합니다.

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

- `-AccessKey` - 액세스 키 ID입니다.
- `-SecretKey` - 보안 키입니다.

- `-StoreAs` - 프로파일 이름으로, 고유해야 합니다. 기본 프로파일을 지정하려면 이름 `default`를 사용합니다.

프로파일 업데이트

AWS SDK 스토어는 수동으로 유지 관리해야 합니다. 나중에 서비스의 자격 증명을 변경하는 경우(예: IAM 콘솔 사용) 로컬에 저장된 자격 증명으로 명령을 실행하면 실패하고 다음 오류 메시지가 표시됩니다.

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

프로파일에 대해 `Set-AWSCredential` 명령을 반복하고 새 액세스 및 보안 키를 전달하여 프로파일을 업데이트할 수 있습니다.

프로파일 나열

다음 명령을 사용하여 현재 이름 목록을 확인할 수 있습니다. 이 예에서 Shirley라는 사용자는 세 개의 프로파일에 액세스할 수 있으며, 이들은 모두 공유 자격 증명 파일(`~/.aws/credentials`)에 저장되어 있습니다.

```
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
```

프로파일 제거

더 이상 필요하지 않은 프로파일을 제거하려면 다음 명령을 사용합니다.

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

`-ProfileName` 파라미터는 삭제할 프로파일을 지정합니다.

더 이상 사용되지 않는 명령 Clear-AWSCredential은 이전 버전과의 호환성을 위해 계속 제공되지만, `Remove-AWSCredentialProfile`가 선호됩니다.

자격 증명 지정

자격 증명을 지정하는 방법에는 몇 가지가 있습니다. 선호되는 방법은 명령줄에 리터럴 자격 증명을 통합하는 대신 프로파일을 식별하는 것입니다.는 자격 증명 검색 순서에 설명된 검색 순서를 사용하여 프로파일을 AWS Tools for PowerShell 찾습니다.

Windows에서 AWS SDK 스토어에 저장된 AWS 자격 증명은 로그인한 Windows 사용자 자격 증명으로 암호화됩니다. 이러한 암호는 다른 계정을 사용하여 해독할 수 없으며, 원래 생성된 계정과 다른 디바이스에서 사용할 수 없습니다. 예약된 작업이 실행될 사용자 계정과 같은 다른 사용자의 작업 증명이 필요한 작업을 수행하려면 해당 사용자로서 컴퓨터에 로그인할 때 사용할 수 있는 자격 증명 프로파일(이전 단원에서 설명)을 설정합니다. 작업 수행 사용자로 로그인하여 자격 증명 설정 단계를 완료하고 해당 사용자에게 적합한 프로파일을 생성합니다. 그런 다음 로그아웃을 하고 사용자 고유의 자격 증명으로 다시 로그인하여 예약된 작업을 설정합니다.

> **Note**
>
> 프로파일을 지정하려면 `-ProfileName` 명령 파라미터를 사용합니다. 이 파라미터는 이전 AWS Tools for PowerShell 릴리스의 `-StoredCredentials` 파라미터와 동일합니다. 이전 버전과의 호환성을 위해 `-StoredCredentials`도 여전히 지원됩니다.

기본 프로파일(권장)

자격 증명이 라는 프로파일에 저장된 경우 모든 AWS SDKs 및 관리 도구가 로컬 컴퓨터에서 자격 증명을 자동으로 찾을 수 있습니다`default`. 예를 들어 로컬 컴퓨터에 `default`라는 이름의 프로파일이 있으면 `Initialize-AWSDefaultConfiguration` cmdlet이나 `Set-AWSCredential` cmdlet을 실행할 필요가 없습니다. 이들 도구는 해당 프로파일에 저장된 액세스 및 보안 키 데이터를 자동으로 사용합니다. 기본 리전(`Get-DefaultAWSRegion`의 결과) 이외의 AWS 리전을 사용하려면 `Set-DefaultAWSRegion`을 실행하고 리전을 지정합니다.

프로파일의 이름이 `default`가 아니지만 현재 세션의 기본 프로파일로 사용하려면 `Set-AWSCredential`을 실행하여 기본 프로파일로 설정합니다.

`Initialize-AWSDefaultConfiguration`을 실행하면 모든 PowerShell 세션의 기본 프로파일을 지정할 수 있습니다. 그러면 cmdlet은 사용자 지정 이름 프로파일에서 자격 증명을 로드하지만, `default` 프로파일을 명명된 프로파일로 덮어씁니다.

인스턴스 프로파일 없이 출시된 Amazon EC2 인스턴스의 PowerShell 세션을 실행하는 중 이거나 자격 증명 프로파일을 수동으로 설정하고자 하는 경우가 아니면 `Initialize-`

AWSDefaultConfiguration을 실행하지 않는 것이 좋습니다. 이 경우 자격 증명 프로파일에는 자격 증명이 포함되어 있지 않다는 점에 유의하십시오. EC2 인스턴스에서 Initialize-AWSDefaultConfiguration을 실행한 결과로 생성되는 자격 증명 프로파일은 자격 증명을 직접 저장하지 않고 인스턴스 메타데이터(자동으로 교체되는 임시 자격 증명을 제공)를 가리킵니다. 하지만 인스턴스의 리전은 저장합니다. 인스턴스가 실행 중인 리전 이외의 리전에 대하여 호출을 실행하고 싶으면 Initialize-AWSDefaultConfiguration를 실행해야 하는 또 다른 상황이 발생할 수 있습니다. 이 명령을 실행하면 인스턴스 메타데이터에 저장된 리전이 영구적으로 재정의됩니다.

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

> **Note**
>
> 기본 자격 증명은 AWS SDK 스토어의 default 프로파일 이름 아래에 포함됩니다. 이 명령은 기존 프로파일을 해당 이름으로 덮어씁니다.

EC2 인스턴스가 인스턴스 프로파일을 사용해 시작된 경우, PowerShell은 인스턴스 프로파일에서 AWS 자격 증명 및 리전 정보를 자동으로 가져옵니다. 따라서 Initialize-AWSDefaultConfiguration를 실행할 필요가 없습니다. 인스턴스 프로파일을 사용해 시작된 EC2 인스턴스에서 Initialize-AWSDefaultConfiguration cmdlet을 실행할 필요가 없습니다. PowerShell이 기본적으로 사용하고 있는 것과 동일한 인스턴스 프로파일 데이터를 사용하기 때문입니다.

세션 프로파일

Set-AWSCredential을 사용하여 특정 세션에 대한 기본 프로파일을 지정합니다. 이 프로파일이 세션 기간 동안 기본 프로파일을 재정의합니다. 현재 default 프로파일을 대신하여 세션에서 사용자가 이름을 지정한 프로파일을 사용하려는 경우에 권장합니다.

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

> **Note**
>
> 1.1 이전 버전의 Tools for Windows PowerShell에서는 Set-AWSCredential cmdlet이 올바르게 작동하지 않아서 "MyProfileName"에 의해 지정된 프로파일을 덮어씁니다. 최신 버전 Tools for Windows PowerShell을 사용하는 것이 좋습니다.

명령 프로파일

개별 명령에서 `-ProfileName` 파라미터를 추가하여 해당되는 단일 명령에만 적용되는 프로파일을 지정할 수 있습니다. 이 프로파일은 다음 예제에서와 같이 기본 또는 세션 프로파일을 재정의합니다.

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

> **Note**
>
> 기본 또는 세션 프로파일을 지정할 때는 `-Region` 파라미터를 추가하여 기본 또는 세션 리전을 지정할 수도 있습니다. 자세한 내용은 AWS 리전 지정 단원을 참조하십시오. 다음 예제에서는 기본 프로파일 및 리전을 지정합니다.
>
> ```
> PS > Initialize-AWSDefaultConfiguration -ProfileName MyProfileName -Region us-west-2
> ```

기본적으로 AWS 공유 자격 증명 파일은 사용자의 홈 폴더(`C:\Users\username\.aws`Windows 또는 Linux)`~/.aws`에 있는 것으로 간주됩니다. 다른 위치에 자격 증명 파일을 지정하려면 `-ProfileLocation` 파라미터를 포함시키고 자격 증명 파일 경로를 지정합니다. 다음 예제에서는 특정 명령에 대한 기본 이외의 자격 증명 파일을 지정합니다.

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

> **Note**
>
> 일반적으로 AWS에 로그인하지 않은 상태에서 PowerShell 스크립트를 실행 중인 경우(예를 들면 일반 업무 시간 이외에 예약된 작업으로서 PowerShell 스크립트를 실행 중인 경우), 사용할 프로파일을 지정할 때 `-ProfileLocation` 파라미터를 추가하고 값을 자격 증명을 저장할 파일의 경로로 설정합니다. AWS Tools for PowerShell 스크립트가 올바른 계정 자격 증명으로 실행되도록 하려면 스크립트가 AWS 계정을 사용하지 않는 컨텍스트 또는 프로세스에서 실행될 때마다 `-ProfileLocation` 파라미터를 추가해야 합니다. 또한 로컬 시스템에 액세스할 수 있는 위치 또는 스크립트가 작업을 수행하는 데 사용하는 다른 계정으로 자격 증명 파일을 복사할 수도 있습니다.

자격 증명 검색 순서

명령을 실행하면가 다음 순서로 자격 증명을 AWS Tools for PowerShell 검색합니다. 사용 가능한 자격 증명을 찾으면 중지됩니다.

1. 명령줄에 파라미터로 내장된 리터럴 자격 증명입니다.

   가급적이면 명령줄에 리터럴 자격 증명을 추가하기 보다는 프로파일을 사용하는 것이 좋습니다.
2. 지정된 프로파일 이름 또는 프로파일 위치.
   - 프로필 이름만 지정하는 경우 명령은 SDK 스토어에서 AWS 지정된 프로필을 찾고, 없는 경우 기본 위치의 AWS 공유 자격 증명 파일에서 지정된 프로필을 찾습니다.
   - 프로파일 위치만 지정하는 경우, 이 명령은 해당 자격 증명 파일에서 `default` 프로파일을 찾습니다.
   - 이름과 위치를 모두 지정하는 경우, 이 명령은 해당 자격 증명 파일에서 지정된 프로파일을 찾습니다.

   지정된 프로파일이나 위치가 없으면 명령에서 예외가 발생합니다. 프로파일이나 위치를 지정하지 않은 경우에만 검색이 다음 단계로 이동합니다.
3. `-Credential` 파라미터에서 지정된 자격 증명.
4. 세션 프로파일(존재하는 경우).
5. 기본 프로파일(다음 순서대로).
   a. AWS SDK 스토어의 `default` 프로필입니다.
   b. AWS 공유 자격 증명 파일의 `default` 프로필입니다.
   c. AWS SDK 스토어의 `AWS PS Default` 프로필입니다.
6. IAM 역할을 사용하도록 구성된 Amazon EC2 인스턴스에서 명령이 실행 중인 경우, 인스턴스 프로파일로부터 액세스한 EC2 인스턴스의 임시 자격 증명입니다.

   Amazon EC2 인스턴스에 IAM 역할 사용에 대한 자세한 내용은 AWS SDK for .NET 단원을 참조하세요.

이 검색을 통해 지정된 자격 증명을 찾지 못한 경우 명령에서 예외가 발생합니다.

의 자격 증명 처리 AWS Tools for PowerShell Core

의 Cmdlet은와 마찬가지로 실행 시 AWS 액세스 및 보안 키 또는 자격 증명 프로필 이름을 AWS Tools for PowerShell Core 수락합니다 AWS Tools for Windows PowerShell. Windows에서 실행될 때 두

모듈 모두 AWS SDK for .NET 자격 증명 저장소 파일(사용자별 AppData\Local\AWSToolkit\RegisteredAccounts.json 파일에 저장)에 액세스할 수 있습니다.

이 파일은 키를 암호화된 형식으로 저장하므로 다른 컴퓨터에서는 사용할 수 없습니다. 가 자격 증명 프로필을 AWS Tools for PowerShell 검색하는 첫 번째 파일이며가 자격 증명 프로필을 AWS Tools for PowerShell 저장하는 파일이기도 합니다. 자격 AWS SDK for .NET 증명 스토어 파일에 대한 자세한 내용은 AWS 자격 증명 구성을 참조하세요. 현재 Tools for Windows PowerShell 모듈에서는 다른 파일이나 위치에 자격 증명 쓰기를 지원하지 않습니다.

두 모듈 모두 다른 AWS SDKs 및에서 사용하는 AWS 공유 자격 증명 파일에서 프로필을 읽을 수 있습니다 AWS CLI. Windows에서 이 파일의 기본 위치는 C:\Users\<userid>\.aws\credentials입니다. Windows 이외의 플랫폼에서는 이 파일이 ~/.aws/credentials에 저장됩니다. -ProfileLocation 파라미터를 사용하여 기본값이 아닌 파일 이름이나 파일 위치를 가리킬 수 있습니다.

SDK 자격 증명 저장소는 Windows 암호화 API를 사용하여 자격 증명을 암호화된 형태로 보관합니다. 이러한 APIs는 다른 플랫폼에서 사용할 수 없으므로 AWS Tools for PowerShell Core 모듈은 AWS 공유 자격 증명 파일만 사용하고 공유 자격 증명 파일에 새 자격 증명 프로파일 쓰기를 지원합니다.

Set-AWSCredential cmdlet을 사용하는 다음 예제에서는 AWSPowerShell 또는 AWSPowerShell.NetCore 모듈을 사용하여 Windows에서 자격 증명 프로파일을 처리하는 옵션을 보여줍니다.

```
# 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
```

다음 예제에서는 Linux 또는 macOS 운영 체제에서 AWSPowerShell.NetCore 모듈의 작동을 보여 줍니다.

```
# 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
```

의 공유 자격 증명 AWS Tools for PowerShell

Tools for Windows PowerShell은 AWS CLI 및 기타 AWS SDKs와 마찬가지로 AWS 공유 자격 증명 파일 사용을 지원합니다. Tools for Windows PowerShell은 이제 .NET `assume role` 자격 증명 파일과 AWS 공유 자격 증명 파일 모두에 `basic`대한 `session`, 및 자격 증명 프로파일 읽기 및 쓰기를 지원합니다. 이 기능은 새 `Amazon.Runtime.CredentialManagement` 네임스페이스를 통해 활성화됩니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

> **Note**
>
> 이 주제의 정보는 단기 또는 장기 보안 인증 정보를 수동으로 획득하고 관리해야 하는 상황을 위한 것입니다. 단기 및 장기 보안 인증 정보에 대한 자세한 내용은 AWS 및 도구 참조 가이드의 다른 인증 방법을 참조하세요.
> 보안 모범 사례를 보려면에 설명된 AWS IAM Identity Center대로를 사용합니다도구 인증 구성.

새 프로필 유형 및 AWS 공유 자격 증명 파일에 대한 액세스는 자격 증명 관련 cmdlet, Initialize-AWSDefaultConfiguration, New-AWSCredential 및 Set-AWSCredential에 추가된 다음 파라미터에서 지원됩니다. 서비스 cmdlet에서는 공통 파라미터인 `-ProfileName`을 추가하여 새 프로파일을 참조할 수 있습니다.

에서 IAM 역할 사용 AWS Tools for PowerShell

AWS 공유 자격 증명 파일을 사용하면 추가 유형의 액세스가 가능합니다. 예를 들어 IAM 사용자의 장기 자격 증명 대신 IAM 역할을 사용하여 AWS 리소스에 액세스할 수 있습니다. 이렇게 하려면 역할을 수임할 권한이 있는 표준 프로파일을 가져야 합니다. 역할을 지정한 프로파일을 사용하도록 AWS Tools for PowerShell 에 지시하면는 `SourceProfile` 파라미터로 식별되는 프로파일을 AWS Tools for PowerShell 조회합니다. 이러한 자격 증명은 `RoleArn` 파라미터에 지정된 역할에 대한 임시 자격 증명을 요청하는 데 사용됩니다. 제3자가 역할을 수임할 때 선택적으로 멀티 팩터 인증(MFA) 디바이스 또는 `ExternalId` 코드를 사용하도록 요구할 수 있습니다.

파라미터 이름	설명
ExternalId	역할에 필요할 경우 역할 수임 시 사용될 사용자 정의 외부 ID입니다. 이는 일반적으로 계정에 대한 액세스 권한을 제3자에게 위임할 때만 필요합니다. 제3자는 할당된 역할을 위임할 때 외부 ID를 파라미터로 포함해야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리소스에 대한 액세스 권한을 타사에 부여할 때 외부 ID를 사용하는 방법을 참조하세요.
MfaSerial	역할에 필요할 경우 역할 수임 시 사용될 MFA 일련 번호입니다. 자세한 내용은 IAM 사용 설명

파라미터 이름	설명
	서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.
RoleArn	역할 자격 증명 수임을 위해 수임할 역할의 ARN입니다. IAM 역할 생성 및 사용에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.
SourceProfile	역할 자격 증명 수임에 사용될 소스 프로파일의 이름입니다. 이 프로파일에서 확인된 자격 증명은 RoleArn 파라미터가 지정한 역할을 수임하는 데 사용됩니다.

역할 수임을 위한 프로파일 설정

다음은 IAM 역할을 직접 수임할 수 있도록 원본 프로파일을 설정하는 방법을 보여주는 예입니다.

첫 번째 명령은 역할 프로파일에서 참조하는 원본 프로파일을 생성합니다.. 두 번째 명령은 어떤 역할을 수임할 것인지에 대한 역할 프로파일을 생성합니다. 세 번째 명령은 역할 프로파일에 대한 자격 증명을 표시합니다.

```
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
```

Tools for Windows PowerShell 서비스 cmdlet과 함께 이 역할 프로파일을 사용하려면 -ProfileName 공통 파라미터를 명령에 추가하여 역할 프로파일을 참조합니다. 다음 예제에서는 이전 예제에서 정의한 역할 프로파일을 사용하여 Get-S3Bucket cmdlet에 액세스 AWS Tools for PowerShell 합니다.에서

자격 증명을 조회하고my_source_profile, 이러한 자격 증명을 사용하여 사용자를 AssumeRole 대신하여를 호출한 다음, 이러한 임시 역할 자격 증명을 사용하여를 호출합니다Get-S3Bucket.

```
PS > Get-S3Bucket -ProfileName my_role_profile

CreationDate           BucketName
------------           ----------
2/27/2017 8:57:53 AM   4ba3578c-f88f-4d8b-b95f-92a8858dac58-bucket1
2/27/2017 10:44:37 AM  2091a504-66a9-4d69-8981-aaef812a02c3-bucket2
```

자격 증명 프로파일 유형 사용

자격 증명 프로파일 유형을 설정하려면 프로파일 유형에 필요한 정보를 제공하는 파라미터를 알아야 합니다.

자격 증명 유형	사용해야 하는 파라미터
기본 이들은 IAM 사용자에 대한 장기 자격 증명입니다.	-AccessKey -SecretKey
세션: 이들은 Use-STSRole cmdlet의 직접 호출 등을 통해 수동으로 검색하는 IAM 역할에 대한 단기 자격 증명입니다..	-AccessKey -SecretKey -SessionToken
역할: 이들은 사용자를 위해 AWS Tools for PowerShell 이 검색하는 IAM 역할에 대한 단기 자격 증명입니다.	-SourceProfile -RoleArn 선택 사항: -ExternalId 선택 사항: -MfaSerial

ProfilesLocation 공통 파라미터

-ProfileLocation을 사용하여 공유 자격 증명 파일에 쓰고 cmdlet에 자격 파일에서 읽도록 지정할 수 있습니다. -ProfileLocation 파라미터를 추가하여 Tools for Windows PowerShell에서 공유 자

격 증명 파일을 사용할지 또는 .NET 자격 증명 파일을 사용할지 여부를 제어할 수 있습니다. 다음 표에서는 Tools for Windows PowerShell에서 이 파라미터가 작동하는 방식을 설명합니다.

프로파일 위치 값	프로파일 해결 동작
null 값(설정되지 않음) 또는 비어 있음	먼저 지정된 이름이 있는 프로파일에 대한 .NET 자격 증명 파일을 검색함. 프로필을 찾을 수 없는 경우에서 AWS 공유 자격 증명 파일을 검색합니다*(user's home directory)* \.aws \credentials .
AWS 공유 자격 증명 파일 형식의 파일 경로	특정 이름이 있는 프로파일에 대해 지정된 파일만 검색함.

자격 증명을 자격 증명 파일에 저장

자격 증명을 써서 두 자격 증명 파일 중 하나에 저장하려면 `Set-AWSCredential` cmdlet을 실행합니다. 다음 예에서는 이 작업을 수행하는 방법을 보여줍니다. 첫 번째 명령은 `Set-AWSCredential`에서 `-ProfileLocation`을(를) 사용하여 `-ProfileName` 파라미터에 의해 지정된 프로파일에 액세스 키와 보안 키를 추가합니다. 두 번째 행에서는 Get-Content cmdlet을 실행하여 자격 증명 파일의 내용을 표시합니다.

```
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
```

자격 증명 프로파일 표시

Get-AWSCredential cmdlet을 실행하고 `-ListProfileDetail` 파라미터를 추가하여 자격 증명 파일 유형 및 위치와 프로파일 이름 목록을 반환합니다.

```
PS > Get-AWSCredential -ListProfileDetail

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

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

자격 증명 프로파일 제거

자격 증명 프로파일을 제거하려면 새 Remove-AWSCredentialProfile cmdlet을 실행합니다. Clear-AWSCredential은 더 이상 사용되지 않지만 이전 버전과의 호환성을 위해 여전히 제공되고 있습니다.

중요 정보

Initialize-AWSDefaultConfiguration, New-AWSCredential 및 Set-AWSCredential만 역할 프로파일에 대한 파라미터를 지원합니다. `Get-S3Bucket -SourceProfile source_profile_name -RoleArn arn:aws:iam::999999999999:role/role_name`와 같은 명령에서 역할 파라미터를 직접 지정할 수 없습니다. 서비스 cmdlet은 `SourceProfile` 또는 `RoleArn` 파라미터를 직접 지원하지 않으므로 이 기능은 작동하지 않습니다. 대신 이러한 파라미터를 프로파일에 저장한 다음 `-ProfileName` 파라미터를 사용하여 명령을 호출해야 합니다.

AWS Tools for Windows PowerShell의 기능

이 섹션의 일부 주제에서는 프로젝트 및 스크립트를 생성할 때 고려해야 할 Tools for Windows PowerShell의 기능에 대한 정보를 제공합니다. 이 섹션의 다른 주제에서는 도구, 환경 및 프로젝트를 구성할 수 있는 고급 방법에 대한 정보를 제공합니다. 먼저 도구를 설치하고 설정해야 합니다.

코드 예제와 함께 특정 AWS 서비스용 소프트웨어 개발에 대한 자세한 내용은 AWS 서비스 작업 단원을 참조하세요. 추가 코드 예제는 Tools for PowerShell V4 코드 예시 단원을 참조하세요.

주제

관찰성

관찰성은 시스템의 현재 상태를 내보내는 데이터에서 추론할 수 있는 범위입니다. 방출되는 데이터를 일반적으로 원격 측정이라고 합니다. AWS 서비스 사용 시 원격 측정에 대한 자세한 내용은 AWS SDK for .NET 개발자 안내서의 관찰성을 참조하세요.

다음 코드는 AWS Tools for PowerShell에서 관찰성을 활성화하는 방법의 예를 보여줍니다.

```
<#
    This is an example of generating telemetry for AWS Tools for PowerShell.
    Each cmdlet that interacts with an Amazon Web Service creates a trace containing
 spans
    for underlying processes and AWS SDK for .NET operations.
    This example is written using PowerShell 7 and .NET 8.
    It requires the installation of the .NET CLI tool.
    Note that implementation varies by the exporter/endpoint, which is not specified in
 this example.
    For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
#>

# Set this value to a common folder path on your computer for local development of code
 repositories.
$devProjectsPath = [System.IO.Path]::Join('C:', 'Dev', 'Repos')

# If these values are changed, update the hardcoded method invocation toward the end of
 this script.
# Values must follow constraints for namespaces and classes.
$telemetryProjectName = 'ExampleAWSPowerShellTelemetryImplementation'
```

```
$serviceName = 'ExamplePowerShellService'

# This example supposes that the OTLP exporter requires these two properties,
# but some exporters require different properties or no properties.
$telemetryEndPoint = 'https://example-endpoint-provider.io'
$telemetryHeaders = 'x-example-header=abc123'

$dllsPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName, 'bin',
 'Release', 'net8.0', 'publish')

$telemetryProjectPath = [System.IO.Path]::Join($devProjectsPath, $telemetryProjectName)

# This script is designed to recreate the example telemetry project each time it's
 executed.
Remove-Item -Path $telemetryProjectPath -Recurse -Force -ErrorAction 'SilentlyContinue'
$null = New-Item -Path $devProjectsPath -Name $telemetryProjectName -ItemType
 'Directory'

<#
    Create and build a C#-based .NET 8 project that implements
    OpenTelemetry Instrumentation for the AWS Tools for PowerShell.
#>

Set-Location -Path $telemetryProjectPath

dotnet new classlib

# Other exporters are available.
# For more information, see https://opentelemetry.io/docs/languages/net/exporters/.
dotnet add package OpenTelemetry.Exporter.OpenTelemetryProtocol
dotnet add package OpenTelemetry.Instrumentation.AWS

$classContent = @"
using OpenTelemetry;
using OpenTelemetry.Resources;
using OpenTelemetry.Trace;

namespace Example.Telemetry;

public class AWSToolsForPowerShellTelemetry
{
    public static void InitializeAWSInstrumentation()
    {
        Sdk.CreateTracerProviderBuilder()
```

```
        .ConfigureResource(e => e.AddService("$ServiceName"))
        .AddAWSInstrumentation()
        // Exporters vary so options might need to be changed or omitted.
        .AddOtlpExporter(options =>
        {
            options.Endpoint = new Uri("$telemetryEndPoint");
            options.Headers = "$telemetryHeaders";
        })
        .Build();
    }
}
"@

$csFilePath = [System.IO.Path]::Join($telemetryProjectPath, ($serviceName + '.cs'))
Set-Content -Path $csFilePath -Value $classContent

dotnet build
dotnet publish -c Release

<#
    Add additional modules here for any other cmdlets that you require.
    Beyond this point, additional AWS Tools for PowerShell modules will fail to import
    due to conflicts with the AWS SDK for .NET assemblies that are added next.
#>

Import-Module -Name 'AWS.Tools.Common'
Import-Module -Name 'AWS.Tools.DynamoDBv2'


# Load assemblies for the telemetry project, excluding the AWS SDK for .NET assemblies
# that were already loaded by importing AWS Tools for PowerShell modules.
$dlls = (Get-ChildItem $dllsPath -Filter *.dll -Recurse ).FullName
$AWSSDKAssembliesAlreadyLoaded =
 [Threading.Thread]::GetDomain().GetAssemblies().Location | Where-Object {$_ -like
 '*AWSSDK*' } | Split-Path -Leaf
$dlls.Where{$AWSSDKAssembliesAlreadyLoaded -notcontains ($_ | Split-Path -
Leaf)}.ForEach{Add-Type -Path $_}

# Invoke the method defined earlier in this script.
[Example.Telemetry.AWSToolsForPowerShellTelemetry]::InitializeAWSInstrumentation()

<#
    Now telemetry will be exported for AWS Tools for PowerShell cmdlets
    that are invoked directly or indirectly.
```

```
    Execute this cmdlet or execute your own PowerShell script.
#>
Get-DDBTable -TableName 'DotNetTests-HashTable' -Region 'us-east-1'
```

에서 AWS 서비스 작업 AWS Tools for PowerShell

이 섹션에서는를 사용하여 AWS 서비스에 AWS Tools for PowerShell 액세스하는 예를 제공합니다. 이 예제는 cmdlet을 사용하여 실제 AWS 작업을 수행하는 방법을 보여 줍니다. 이러한 예제에서는 Tools for PowerShell에서 제공하는 cmdlet을 사용합니다. 사용할 수 있는 cmdlet을 확인하려면 AWS Tools for PowerShell Cmdlet Reference를 참조하세요.

PowerShell 파일 연결 인코딩

의 일부 cmdlet은 기존 파일 또는 레코드를 AWS Tools for PowerShell 편집합니다 AWS. Amazon Route 53를 위한 ChangeResourceRecordSets API를 호출하는 `Edit-R53ResourceRecordSet`이 바로 그 예입니다.

PowerShell 5.1 이전 릴리스에서 파일을 편집하거나 연결할 때 PowerShell이 UTF-8이 아니라 UTF-16으로 출력을 인코딩합니다. 그러면 원치 않는 문자가 추가되고 잘못된 결과가 생길 수 있습니다. 16진수 편집기에서 원치 않는 문자를 볼 수 있습니다.

파일 출력을 UTF-16으로 변환하지 않으려면 다음 예제와 같이 PowerShell의 `Out-File` cmdlet에 명령을 파이프하고 UTF-8 인코딩을 지정할 수 있습니다.

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

PowerShell 콘솔 내에서 AWS CLI 명령을 실행하는 경우 동일한 동작이 적용됩니다. PowerShell 콘솔에서 AWS CLI 명령의 출력을 로 파이프할 수 `Out-File` 있습니다. `Export-Csv`나 `Export-Clixml` 같은 다른 cmdlet에도 `Encoding` 파라미터가 있습니다. `Encoding` 매개 변수가 있고 연결된 파일의 출력을 올바르게 인코딩하도록 해주는 cmdlet의 전체 목록을 보려면 다음 명령을 실행합니다.

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

> **Note**
>
> PowerShell Core를 포함한 PowerShell 6.0 이상에서는 연결된 파일 출력을 위해 UTF-8 인코딩을 자동으로 유지합니다.

PowerShell 도구에 대해 반환된 객체

네이티브 PowerShell 환경에서 AWS Tools for PowerShell 더 유용하게 사용하기 위해 AWS Tools for PowerShell cmdlet에서 반환되는 객체는 일반적으로 AWS SDK의 해당 API에서 반환되는 JSON 텍스트 객체가 아닌 .NET 객체입니다. 예를 들어, `Get-S3Bucket`는 Amazon S3 JSON 응답 객체가 아닌 `Buckets` 컬렉션을 방출합니다. `Buckets` 컬렉션은 PowerShell 파이프라인에 배치되어 적절한 방식으로 상호 작용할 수 있습니다. 마찬가지로 `Get-EC2Instance`은 `DescribeEC2Instances` JSON 결과 객체가 아니라 `Reservation` .NET 객체 컬렉션을 방출합니다. 이러한 동작은 설계상이며 AWS Tools for PowerShell 환경을 관용적인 PowerShell과 더 일관되게 유지할 수 있습니다.

필요한 경우 실제 서비스 응답을 사용할 수 있습니다. 이러한 응답은 반환된 객체에서 `note` 속성으로 저장됩니다. `NextToken` 필드를 사용하여 페이징을 지원하는 API 작업의 경우, `note` 속성으로도 연결됩니다.

Amazon EC2

이 단원에서는 다음 방법을 비롯하여 Amazon EC2 인스턴스를 시작하는 데 필요한 단계를 안내합니다.

- Amazon Machine Images(AMI) 목록을 검색합니다.
- SSH 인증을 위한 키 페어를 생성합니다.
- Amazon EC2 보안 그룹을 생성 및 구성합니다.
- 인스턴스를 시작하고 인스턴스에 대한 정보를 검색합니다.

Amazon S3

이 단원은 Amazon S3에 호스팅된 정적 웹 사이트를 생성하는 데 필요한 단계를 안내합니다. 다음 방법을 설명합니다.

- Amazon S3 버킷을 생성하고 삭제합니다.
- 파일을 Amazon S3 버킷에 객체로 업로드합니다.
- Amazon S3 버킷에서 객체를 삭제합니다.
- Amazon S3 버킷을 웹 사이트로 지정합니다.

AWS Lambda 및 AWS Tools for PowerShell

이 섹션에서는 PowerShell용 AWS Lambda 도구 모듈에 대한 간략한 개요를 제공하고 모듈 설정에 필요한 단계를 설명합니다.

Amazon SNS 및 Amazon SQS

이 단원에서는 Amazon SNS 주제에 대한 Amazon SQS 대기열을 구독하는 데 필요한 단계를 안내합니다. 다음 방법을 설명합니다.

- Amazon SNS 주제를 생성합니다.
- Amazon SQS 대기열을 생성합니다.
- 주제에 대한 대기열을 구독합니다.
- 메시지를 주제로 전송합니다.
- 대기열에서 메시지를 검색합니다.

CloudWatch

이 단원에서는 CloudWatch에 사용자 지정 데이터를 게시하는 방법의 예를 제공합니다.

- CloudWatch 대시보드에 사용자 지정 지표를 게시합니다.

참고

- AWS Tools for Windows PowerShell 시작

주제

- Amazon S3 및 Tools for Windows PowerShell
- Amazon EC2 및 Tools for Windows PowerShell
- AWS Lambda 및 AWS Tools for PowerShell
- Amazon SQS, Amazon SNS 및 Tools for Windows PowerShell
- 의 CloudWatch AWS Tools for Windows PowerShell
- cmdlet에서 ClientConfig 파라미터 사용

Amazon S3 및 Tools for Windows PowerShell

이 단원에서는 Amazon S3 및 CloudFront를 사용하여 AWS Tools for Windows PowerShell을(를) 사용하는 정적 웹 사이트를 만듭니다. 이 프로세스에서는 이러한 서비스를 이용한 일반적인 작업을 설명합니다. 이 시연은 AWS관리 콘솔을 사용한 유사한 프로세스를 설명하는 정적 웹 사이트 호스팅에 대한 시작 안내서를 모델링하고 있습니다.

여기에 표시된 명령은 PowerShell 세션의 기본 자격 증명 및 기본 리전을 설정했다고 가정합니다. 그러므로 자격 증명 리전이 cmdlet 호출에 포함되지 않습니다.

> **Note**
>
> 현재는 버킷이나 객체 이름을 변경하기 위한 Amazon S3 API가 없으므로 이 작업을 수행하기 위한 Tools for Windows PowerShell cmdlet도 없습니다. S3의 객체 이름을 변경하려면 Copy-S3Object cmdlet을 실행하여 새 이름을 가진 객체로 객체를 복사한 후, Remove-S3Object cmdlet을 실행하여 원래 객체를 삭제하는 것이 좋습니다.

다음 사항도 참조하세요.

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- Amazon S3에서 정적 웹 사이트 호스팅
- Amazon S3 콘솔

주제

- Amazon S3 버킷 생성, 리전 확인 및 제거(선택 사항)
- Amazon S3 버킷을 웹 사이트로 구성하고 로깅 활성화
- Amazon S3 버킷에 객체 업로드
- Amazon S3 객체 및 버킷 삭제
- Amazon S3에 인라인 텍스트 콘텐츠 업로드

Amazon S3 버킷 생성, 리전 확인 및 제거(선택 사항)

`New-S3Bucket` cmdlet을 사용하여 새 Amazon S3 버킷을 생성합니다. 다음 예제에서는 `website-example`이라는 버킷을 생성합니다. 버킷의 이름은 모든 리전에서 고유해야 합니다. 이 예제에서는 `us-west-1` 리전에 버킷을 생성합니다.

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

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

Get-S3BucketLocation cmdlet을 사용하여 버킷이 위치한 리전을 확인할 수 있습니다.

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

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

이 자습서를 마치면 다음 라인을 사용하여 이 버킷을 제거할 수 있습니다. 후속 예제에서 사용해야 하므로 이 버킷을 제 위치에 그대로 둘 것을 제안합니다.

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

버킷 제거 프로세스는 완료하는 데 다소 시간이 걸립니다. 동일한 이름의 버킷을 즉시 다시 생성하려고 하면 이전 버킷이 완전히 사라질 때까지 New-S3Bucket cmdlet이 실패할 수 있습니다.

참고

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- Put 버킷(Amazon S3 서비스 참조)
- AWS Amazon S3용 PowerShell 리전

Amazon S3 버킷을 웹 사이트로 구성하고 로깅 활성화

Write-S3BucketWebsite cmdlet을 사용하여 Amazon S3 버킷을 정적 웹 사이트로 구성합니다. 다음 예제에서는 기본 콘텐츠 웹 페이지에 index.html 이름과 기본 오류 웹 페이지에 error.html 이름을 지정합니다. 이 cmdlet은 이러한 페이지를 만들지 않습니다. 이러한 페이지를 Amazon S3 객체로서 업로드해야 합니다.

```
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    :
```

참고

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- Put 버킷 웹 사이트(Amazon S3 API 참조)
- Put 버킷 ACL(Amazon S3 API 참조)

Amazon S3 버킷에 객체 업로드

로컬 파일 시스템의 파일을 Amazon S3 버킷에 객체로 업로드하려면 `Write-S3Object` cmdlet을 사용합니다. 아래 예제에서는 간단한 HTML 파일 두 개를 작성하여 Amazon S3 버킷에 업로드하고 업로드된 객체의 유무를 확인합니다. `-File`에 대한 `Write-S3Object` 파라미터는 로컬 파일 시스템의 파일 이름을 지정합니다. `-Key` 파라미터는 Amazon S3에서 해당 객체에 사용될 이름을 지정합니다.

Amazon은 파일 확장명에서 객체의 콘텐츠 유형(이 경우 ".html")을 유추합니다.

```
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
```

미리 준비된 ACL 옵션

Tools for Windows PowerShell을 사용하여 미리 준비된 ACL을 지정하는 값은 AWS SDK for .NET에서 사용되는 것과 동일합니다. 하지만 이러한 값은 Amazon S3 `Put Object` 작업에 사용되는 값과는 다릅니다. Tools for Windows PowerShell은 다음과 같은 미리 준비된 ACL을 지원합니다.

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

미리 준비된 ACL 설정에 대한 자세한 내용은 액세스 제어 목록 개요를 참조하십시오.

멀티파트 업로드에 관한 정보

Amazon S3 API를 사용하여 5GB보다 큰 파일을 업로드하는 경우 멀티파트 업로드를 사용해야 합니다. 하지만 Tools for Windows PowerShell에서 제공하는 `Write-S3Object` cmdlet은 5GB보다 큰 파일 업로드를 투명하게 처리할 수 있습니다.

웹 사이트 테스트

이때 브라우저에서 탐색하여 웹 사이트를 테스트할 수 있습니다. Amazon S3에 호스팅된 정적 웹 사이트의 URL은 표준 형식을 따릅니다.

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

예:

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

참고

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- Put 객체(Amazon S3 API 참조)
- 미리 준비된 ACL(Amazon S3 API 참조)

Amazon S3 객체 및 버킷 삭제

이 단원에서는 이전 단원에 생성된 웹 사이트를 삭제하는 방법을 설명합니다. HTML 파일에 대한 객체를 삭제하고 나서 이 사이트에 대한 Amazon S3 버킷을 삭제하면 됩니다.

먼저, `Remove-S3Object` cmdlet을 실행하여 Amazon S3 버킷에서 HTML 파일에 대한 객체를 삭제합니다.

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

`False` 응답은 Amazon S3에서 요청을 처리하는 방법의 예상된 아티팩트입니다. 이 경우 이는 문제를 의미하지 않습니다.

이제 `Remove-S3Bucket` cmdlet을 실행해 사이트에서 현재 비어 있는 Amazon S3 버킷을 삭제할 수 있습니다.

```
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    :
```

AWS Tools for PowerShell의 1.1 이상 버전에서는 `-DeleteBucketContent` 매개 변수를 `Remove-S3Bucket`에 추가할 수 있습니다. 그러면 지정된 버킷에서 모든 객체 및 객체 버전이 삭제되고 나서 버킷 자체가 제거됩니다. 버킷의 객체 또는 객체 버전 수에 따라 이 작업에는 상당한 시간이 걸릴 수도 있습니다. 1.1 이전 버전의 Tools for Windows PowerShell에서는 버킷을 비워야만 `Remove-S3Bucket`에서 버킷을 삭제할 수 있었습니다.

> **Note**
>
> `-Force` 매개 변수를 추가하지 않는 한, cmdlet을 실행하기 전에 AWS Tools for PowerShell에서 확인 메시지가 표시됩니다.

참고

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- 객체 삭제(Amazon S3 API 참조)
- DeleteBucket(Amazon S3 API 참조)

Amazon S3에 인라인 텍스트 콘텐츠 업로드

`Write-S3Object` cmdlet에서는 Amazon S3에 인라인 텍스트 콘텐츠를 업로드하는 기능을 지원합니다. `-Content` (별칭 `-Text`)를 사용하면 파일에 먼저 붙여 넣지 않고도 Amazon S3에 업로드할 텍스트 기반 내용을 지정할 수 있습니다. 이 파라미터에는 간단한 한 줄 문자열은 물론 여기에 나오는 여러 줄을 포함하는 문자열도 사용할 수 있습니다.

```
PS > # Specifying content in-line, single line text:
PS > write-s3object amzn-s3-demo-bucket -key myobject.txt -content "file content"

PS > # Specifying content in-line, multi-line text: (note final newline needed to end
 in-line here-string)
```

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

Amazon EC2 및 Tools for Windows PowerShell

AWS Tools for PowerShell을 사용하여 Amazon EC2와 관련된 일반적인 작업을 수행할 수 있습니다.

여기에 표시된 예제에서는 PowerShell 세션의 기본 자격 증명 및 기본 리전을 설정했다고 가정합니다. 그러므로 cmdlet을 호출할 때 자격 증명이나 리전을 포함하지 않습니다. 자세한 내용은 AWS Tools for Windows PowerShell 시작 섹션을 참조하세요.

주제

- 키 페어 만들기
- Windows PowerShell을 사용하여 보안 그룹 생성
- Windows PowerShell을 사용하여 Amazon Machine Images 찾기
- Windows PowerShell을 사용하여 Amazon EC2 인스턴스 시작

키 페어 만들기

다음 `New-EC2KeyPair` 예제에서는 키 페어를 만들고 PowerShell 변수 `$myPSKeyPair`에 이를 저장합니다.

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

키 페어 객체를 `Get-Member` cmdlet에 파이프하여 객체의 구조를 확인합니다.

```
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;}
```

키 페어 객체를 `Format-List` cmdlet에 파이프하여 `KeyName`, `KeyFingerprint` 및 `KeyMaterial` 멤버의 값을 봅니다. (읽기 쉽도록 출력을 잘랐습니다.)

```
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 멤버는 키 페어의 프라이빗 키를 저장합니다. 퍼블릭 키는 AWS에 저장됩니다. AWS에서 퍼블릭 키를 검색할 수는 없지만, 프라이빗 키의 KeyFingerprint를 퍼블릭 키용으로 AWS에서 반환된 것과 비교하여 퍼블릭 키를 확인할 수 있습니다.

키 페어 지문 보기

Get-EC2KeyPair cmdlet을 사용하여 키 페어의 지문을 볼 수 있습니다.

```
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
```

프라이빗 키 저장

프라이빗 키를 파일에 저장하려면 KeyFingerMaterial 멤버를 Out-File cmdlet에 파이프합니다.

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

프라이빗 키를 파일에 쓸 때 -Encoding ascii를 지정해야 합니다. 그렇지 않으면, openssl과 같은 도구에서 파일을 올바르게 읽지 못할 수도 있습니다. 다음과 같은 명령을 사용하여 결과 파일의 형식이 올바른지 확인할 수 있습니다.

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

(openssl 도구는 AWS Tools for PowerShell 또는 AWS SDK for .NET에 포함되지 않습니다.)

키 페어 제거

인스턴스를 시작하고 연결하려면 키 페어가 필요합니다. 키 페어를 사용한 후 제거할 수 있습니다. AWS에서 퍼블릭 키를 제거하려면 Remove-EC2KeyPair cmdlet을 사용합니다. 메시지가 표시되면 Enter를 눌러서 키 페어를 제거합니다.

```
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"):
```

`$myPSKeyPair` 변수는 현재 PowerShell 세션에 여전히 존재하며 여전히 키 페어 정보를 포함합니다. `myPSKeyPair.pem` 파일도 존재합니다. 하지만 키 페어의 퍼블릭 키가 AWS에 더 이상 저장되어 있지 않으므로 프라이빗 키는 더 이상 유효하지 않습니다.

Windows PowerShell을 사용하여 보안 그룹 생성

AWS Tools for PowerShell를 사용하여 보안 그룹을 생성 및 구성할 수 있습니다. 응답으로 보안 그룹의 ID가 반환됩니다.

인스턴스에 연결해야 하는 경우 SSH 트래픽(Linux) 또는 RDP 트래픽(Windows)를 허용하도록 보안 그룹을 구성해야 합니다.

주제

- 사전 조건
- EC2-VPC에 대한 보안 그룹 생성

사전 조건

컴퓨터의 퍼블릭 IP 주소를 CIDR 표기법으로 지정해야 합니다. 서비스를 사용하여 로컬 컴퓨터의 퍼블릭 IP 주소를 확인할 수 있습니다. 예를 들면, Amazon에서는 http://checkip.amazonaws.com/ 또는 https://checkip.amazonaws.com/ 서비스를 제공합니다. IP 주소를 제공하는 다른 서비스를 찾으려면 "what is my IP address"로 검색하십시오. 고정 IP 주소 없이 ISP를 통해, 또는 방화벽 뒤에서 연결하는 경우에는 클라이언트 컴퓨터가 사용할 수 있는 IP 주소의 범위를 찾아야 합니다.

> **Warning**
>
> `0.0.0.0/0`을 지정하면 전 세계의 모든 IP 주소에서 트래픽을 사용하도록 설정됩니다. SSH 및 RDP 프로토콜의 경우, 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션 환경에서는 적절한 개별 IP 주소 또는 주소 범위에서만 액세스 권한을 부여해야 합니다.

EC2-VPC에 대한 보안 그룹 생성

> **Warning**
>
> EC2-Classic은 2022년 8월 15일에 사용 중지되었습니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. 자세한 내용은 블로그 게시물 EC2-Classic 네트워킹은 사용 중지 중입니다. 준비 방법은 다음과 같습니다를 참조하세요.

다음 `New-EC2SecurityGroup` 예제에서는 `-VpcId` 매개 변수를 추가하여 지정된 VPC에 대한 보안 그룹을 생성합니다.

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

보안 그룹의 초기 구성을 보려면 `Get-EC2SecurityGroup` cmdlet을 사용합니다. 기본적으로 VPC의 보안 그룹은 모든 아웃바운드 트래픽을 허용하는 아웃바운드 규칙을 포함합니다. 이름으로 EC2-VPC에 대한 보안 그룹을 참조할 수 없습니다.

```
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 포트 22(SSH) 및 TCP 포트 3389에서 인바운드 트래픽에 대한 권한을 정의하려면 `New-Object` cmdlet을 사용합니다. 다음 예제 스크립트는 단일 IP 주소 `203.0.113.25/32`에서 TCP 포트 22 및 3389에 대한 사용 권한을 정의합니다.

```
$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 )
```

보안 그룹이 업데이트되었는지 확인하려면 `Get-EC2SecurityGroup` cmdlet을 다시 사용합니다.

```
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                : {}
```

인바운드 규칙을 보기 위해 이전 명령에서 반환한 컬렉션 개체에서 `IpPermissions` 속성을 검색할 수 있습니다.

```
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 PowerShell을 사용하여 Amazon Machine Images 찾기

Amazon EC2 인스턴스를 시작할 때 인스턴스의 템플릿으로 사용할 Amazon Machine Images(AMI)를 지정합니다. 그러나는 최신 업데이트 및 보안 개선 사항을 새 AMIs에 AWS 제공하기 때문에 AWS

Windows AMIs의 IDs는 자주 변경됩니다. Get-EC2Image 및 Get-EC2ImageByName cmdlet을 사용하여 현재 Windows AMI를 찾아서 해당 ID를 가져올 수 있습니다.

주제

- Get-EC2Image
- Get-EC2ImageByName

Get-EC2Image

`Get-EC2Image` cmdlet은 사용할 수 있는 AMI 목록을 검색합니다.

`-Owner`에서 Amazon이나 사용자에게 속하는 AMI만 검색되도록 `amazon, self` 파라미터와 어레이 값 `Get-EC2Image`를 사용합니다. 이 컨텍스트에서 사용자란 cmdlet을 호출하는 데 사용한 자격 증명을 가진 사용자를 지칭합니다.

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

`-Filter` 파라미터를 사용하여 결과의 범위를 지정할 수 있습니다. 필터를 지정하려면 `Amazon.EC2.Model.Filter` 유형의 객체를 생성합니다. 예를 들어, 다음 필터를 사용하면 Windows AMI만 표시됩니다.

```
$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
```

다음 예에서는 이 cmdlet에서 반환되는 AMI 중 하나를 보여 주며, 이전 명령의 실제 출력은 여러 AMI에 대한 정보를 제공합니다.

```
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 cmdlet에서는 관심 있는 서버 구성 유형에 따라 AWS Windows AMI 목록을 필터링할 수 있습니다.

매개 변수 없이 실행할 경우 다음과 같이 cmdlet에서 현재 필터 이름의 전체 세트가 방출됩니다.

```
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
```

반환되는 이미지 세트의 범위를 좁히려면 Names 파라미터를 사용하여 하나 이상의 필터 이름을 지정합니다.

```
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 PowerShell을 사용하여 Amazon EC2 인스턴스 시작

Amazon EC2 인스턴스를 시작하려면 이전 단원에서 생성한 키 페어 및 보안 그룹이 필요합니다. Amazon Machine Images(AMI)의 ID도 필요합니다. 자세한 내용은 다음 설명서를 참조하세요.

- 키 페어 만들기
- Windows PowerShell을 사용하여 보안 그룹 생성
- Windows PowerShell을 사용하여 Amazon Machine Images 찾기

> **Important**
>
> 시작하는 인스턴스가 프리 티어에 해당되지 않는 경우 인스턴스를 시작한 후에 요금이 청구되고 유휴 상태를 포함해 인스턴스가 실행된 시간에 대해 과금됩니다.

주제

- VPC에서 인스턴스 시작
- VPC에서 스팟 인스턴스 시작

VPC에서 인스턴스 시작

> **Warning**
>
> EC2-Classic은 2022년 8월 15일에 사용 중지되었습니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. 자세한 내용은 블로그 게시물 EC2-Classic 네트워킹은 사용 중지 중입니다. 준비 방법은 다음과 같습니다를 참조하세요.

다음 명령은 지정된 프라이빗 서브넷에서 단일 m1.small 인스턴스를 생성합니다. 보안 그룹은 지정된 서브넷에 대해 유효해야 합니다.

```
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       : {}
```

처음에는 인스턴스가 pending 상태이지만 몇 분 후에 running 상태가 됩니다. 인스턴스에 대한 정보를 보려면 Get-EC2Instance cmdlet을 사용합니다. 인스턴스가 두 개 이상인 경우 Filter 파라미터를 사용하여 예약 ID에 대해 결과를 필터링할 수 있습니다. 먼저 Amazon.EC2.Model.Filter 유형의 객체를 생성합니다. 그런 다음 필터를 사용하고 Get-EC2Instance 속성을 표시하는 Instances을 호출합니다.

```
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
```

VPC에서 스팟 인스턴스 시작

다음 예제 스크립트는 지정된 서브넷에서 스팟 인스턴스를 요청합니다. 보안 그룹은 지정된 서브넷을 포함하는 VPC에 대해 생성된 보안 그룹이어야 합니다.

```
$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 및 AWS Tools for PowerShell

AWSLambdaPSCore 모듈을 사용하면 .NET Core 2.1 런타임을 사용하여 PowerShell Core 6.0에서 AWS Lambda 함수를 개발할 수 있습니다. PowerShell 개발자는 Lambda를 사용하여 PowerShell 환경에서 AWS 리소스를 관리하고 자동화 스크립트를 작성할 수 있습니다. Lambda의 PowerShell 지원을 사용하면 Amazon S3 이벤트 또는 Amazon CloudWatch 예약 이벤트 등과 같은 Lambda 이벤트에 대응하여 PowerShell 스크립트 또는 함수를 실행할 수 있습니다. AWSLambdaPSCore 모듈은 PowerShell용 별도의 AWS 모듈이며 AWS Tools for PowerShell,의 일부가 아니며 AWSLambdaPSCore 모듈을 설치해도가 설치되지 않습니다 AWS Tools for PowerShell.

AWSLambdaPSCore 모듈을 설치한 후 사용 가능한 PowerShell cmdlet을 사용하거나 직접 개발하여 서버리스 함수를 작성할 수 있습니다. PowerShell용 AWS Lambda 도구 모듈에는 PowerShell 기반 서버리스 애플리케이션을 위한 프로젝트 템플릿과 프로젝트를 게시할 도구가 포함되어 있습니다 AWS.

AWSLambdaPSCore 모듈 지원은 Lambda를 지원하는 모든 리전에서 사용할 수 있습니다. 지원되는 리전에 대한 자세한 내용은 AWS 리전 표를 참조하세요.

사전 조건

AWSLambdaPSCore 모듈을 설치하고 사용하려면 먼저 다음 절차를 수행해야 합니다. 이러한 단계에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 PowerShell 개발 환경 설정을 참조하세요.

- 올바른 버전의 PowerShell 설치 – Lambda의 PowerShell 지원은 크로스 플랫폼 PowerShell Core 6.0 릴리스를 기반으로 합니다. Windows, Linux 또는 Mac에서 PowerShell Lambda 함수를 개발할 수 있습니다. 이 버전 이상의 PowerShell이 설치되어 있지 않으면 Microsoft PowerShell 설명서 웹 사이트의 지침을 참조하세요.
- .NET Core 2.1 SDK 설치 – PowerShell은 .NET Core를 기반으로 하기 때문에 Lambda의 PowerShell 지원은 .NET Core와 PowerShell Lambda 함수 모두에 동일한 .NET Core 2.1 Lambda 런타임을 사용합니다. Lambda PowerShell 게시 cmdlet은 .NET Core 2.1 SDK를 사용하여 Lambda 배포 패키지를 생성합니다. .NET Core 2.1 SDK는 Microsoft 다운로드 센터에서 구할 수 있습니다. Runtime이 아닌 SDK를 설치해야 합니다.

AWSLambdaPSCore 모듈 설치

사전 요구 사항을 모두 갖췄으면 이제 AWSLambdaPSCore 모듈을 설치할 수 있습니다. PowerShell Core 세션에서 다음 명령을 실행합니다.

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

이제 PowerShell에서 Lambda 함수 개발을 시작할 수 있습니다. 시작하는 방법에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 PowerShell에서 Lambda 함수를 작성하기 위한 프로그래밍 모델을 참조하세요.

참고

- AWS 개발자 블로그에서 PowerShell Core에 대한 Lambda 지원 발표
- PowerShell 갤러리의 AWSLambdaPSCore 모듈
- PowerShell 개발 환경 설정
- AWS GitHub의 Powershell용 Lambda 도구
- AWS Lambda 콘솔

Amazon SQS, Amazon SNS 및 Tools for Windows PowerShell

이 단원에서는 다음 작업을 수행하는 방법을 보여 주는 예제를 제공합니다.

- Amazon SQS 대기열을 생성하고 대기열 Amazon 리소스 이름(ARN)을 가져옵니다.
- Amazon SNS 주제를 생성합니다.
- 대기열에 메시지를 보낼 수 있도록 SNS 주제에 권한을 부여합니다.
- SNS 주제에 대한 대기열을 구독합니다.
- IAM 사용자 또는 AWS 계정에 SNS 주제에 게시하고 SQS 대기열에서 메시지를 읽을 수 있는 권한을 부여합니다.
- 주제에 대한 메시지를 게시하고 대기열에서 메시지를 읽어 결과를 확인합니다.

Amazon SQS 대기열 생성 및 대기열 ARN 가져오기

다음 명령은 기본 리전에 SQS 대기열을 생성합니다. 새 대기열의 URL이 출력에 표시됩니다.

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

다음 명령은 대기열의 ARN을 검색합니다.

```
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 주제 생성

다음 명령은 기본 리전에서 SNS 주제를 생성하고 새 주제의 ARN을 반환합니다.

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

SNS 주제에 권한 부여

다음 예제 스크립트는 SQS 대기열과 SNS 주제를 모두 생성하고, SQS 대기열로 메시지를 보낼 수 있도록 SNS 주제에 대한 권한을 부여합니다.

```
# 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 }
```

SNS 주제에 대한 대기열을 구독합니다.

다음 명령은 SNS 주제 myTopic에 대한 대기열 myQueue를 구독하고 구독 ID를 반환합니다.

```
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
```

권한 부여

다음 명령은 myTopic 주제에 대한 sns:Publish 작업을 수행할 수 있는 권한을 부여합니다.

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

다음 명령은 대기열 myQueue에 대한 sqs:ReceiveMessage 및 sqs:DeleteMessage 작업을 수행할 수 있는 권한을 부여합니다.

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

결과 확인

다음 명령은 SNS 주제 myTopic에 메시지를 게시하여 새 대기열 및 주제를 테스트하고 MessageId를 반환합니다.

```
PS > Publish-SNSMessage `
    -TopicArn arn:aws:sns:us-west-2:123456789012:myTopic `
```

```
    -Message "Have A Nice Day!"
728180b6-f62b-49d5-b4d3-3824bb2e77f4
```

다음 명령은 SQS 대기열 myQueue에서 메시지를 검색하여 표시합니다.

```
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==
```

의 CloudWatch AWS Tools for Windows PowerShell

이 단원에서는 Tools for Windows PowerShell을 사용하여 사용자 지정 지표 데이터를 CloudWatch에 게시하는 방법의 예를 보여 줍니다.

이 예제에서는 PowerShell 세션의 기본 자격 증명 및 기본 리전을 설정했다고 가정합니다.

CloudWatch 대시보드에 사용자 지정 지표 게시

다음 PowerShell 코드는 CloudWatch `MetricDatum` 객체를 초기화하여 서비스에 게시합니다. CloudWatch 콘솔로 이동하여 이 작업의 결과를 볼 수 있습니다.

```
$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
```

다음 사항에 유의하세요.

- `$dat.Timestamp`를 초기화하는 데 사용하는 날짜/시간 정보는 세계시(UTC)로 설정해야 합니다.
- `$dat.Value`을 초기화하는 데 사용하는 값은 따옴표 안에 문자열 값으로 지정하거나 숫자 값(따옴표 없음)으로 지정할 수 있습니다. 이 예에서는 문자열 값을 보여 줍니다.

참고

- 에서 AWS 서비스 작업 AWS Tools for PowerShell
- AmazonCloudWatchClient.PutMetricData (.NET SDK 참조)
- MetricDatum (서비스 API 참조)
- Amazon CloudWatch 콘솔

cmdlet에서 ClientConfig 파라미터 사용

`ClientConfig` 파라미터는 서비스에 연결할 때 특정 구성 설정을 지정하는 데 사용할 수 있습니다. 이 파라미터의 가능한 대부분의 속성은 `Amazon.Runtime.ClientConfig` 클래스에 정의되어 있으며, 이는 AWS 서비스용 API로 상속됩니다. 단순 상속의 예는

Amazon.Keyspaces.AmazonKeyspacesConfig 클래스를 참조하세요. 또한 일부 서비스는 해당 서비스에만 적합한 추가 속성을 정의합니다. 정의된 추가 속성의 예는 Amazon.S3.AmazonS3Config 클래스, 특히 ForcePathStyle 속성을 참조하세요.

ClientConfig 파라미터 사용

ClientConfig 파라미터를 사용하려면 명령줄에서 파라미터를 ClientConfig 객체로 지정하거나 PowerShell 스플래팅을 사용하여 파라미터 값 컬렉션을 명령에 단위로 전달할 수 있습니다. 이러한 메서드는 아래 예와 같습니다. 예에서는 AWS.Tools.S3 모듈을 설치하여 가져왔고 적절한 권한이 있는 [default] 보안 인증 정보 프로필이 있다고 가정합니다.

ClientConfig 객체 정의

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

PowerShell 스플래팅을 사용하여 **ClientConfig** 속성 추가

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

Get-S3Object @params
```

정의되지 않은 속성 사용

PowerShell 스플래팅을 사용할 때 존재하지 않는 ClientConfig 속성을 지정하면 AWS Tools for PowerShell은 런타임 때까지 오류를 감지하지 못하고 런타임 시 예외를 반환합니다. 위의 예 수정:

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

```
    }
    BucketName="<BUCKET_NAME>"
}

Get-S3Object @params
```

다음과 비슷한 예외가 생성됩니다.

```
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 리전 지정

`ClientConfig` 파라미터를 사용하여 명령에 AWS 리전을 설정할 수 있습니다. 리전은 `RegionEndpoint` 속성을 통해 설정됩니다. AWS Tools for PowerShell은 다음 우선 순위에 따라 사용할 리전을 계산합니다.

1. `-Region` 파라미터
2. `ClientConfig` 파라미터에 전달된 리전
3. PowerShell 세션 상태
4. 공유된 AWS`config` 파일
5. 환경 변수
6. Amazon EC2 인스턴스 메타데이터 서비스(활성화된 경우)

Tools for PowerShell V4 코드 예시

이 주제의 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- Tools for PowerShell V4를 사용한 ACM 예제
- Tools for PowerShell V4를 사용한 Application Auto Scaling 예제
- Tools for PowerShell V4를 사용한 WorkSpaces 애플리케이션 예제
- Tools for PowerShell V4를 사용한 Aurora 예제
- Tools for PowerShell V4를 사용한 Auto Scaling 예제
- AWS Budgets Tools for PowerShell V4를 사용한 예제
- AWS Cloud9 Tools for PowerShell V4를 사용한 예제
- CloudFormation Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 CloudFront 예제
- Tools for PowerShell V4를 사용한 CloudTrail 예제
- Tools for PowerShell V4를 사용한 CloudWatch 예제
- Tools for PowerShell V4를 사용한 CodeCommit 예제
- Tools for PowerShell V4를 사용한 CodeDeploy 예제
- Tools for PowerShell V4를 사용한 CodePipeline 예제
- Tools for PowerShell V4를 사용한 Amazon Cognito ID 예제
- AWS Config Tools for PowerShell V4를 사용한 예제

- Tools for PowerShell V4를 사용한 Device Farm 예제
- Directory Service Tools for PowerShell V4를 사용한 예제
- AWS DMS Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 DynamoDB 예제
- Tools for PowerShell V4를 사용한 Amazon EC2 예제
- Tools for PowerShell V4를 사용한 Amazon ECR 예제
- Tools for PowerShell V4를 사용한 Amazon ECS 예제
- Tools for PowerShell V4를 사용한 Amazon EFS 예제
- Tools for PowerShell V4를 사용한 Amazon EKS 예제
- Tools for PowerShell V4를 사용한 Elastic Load Balancing - 버전 1 예제
- Tools for PowerShell V4를 사용한 Elastic Load Balancing - 버전 2 예제
- Tools for PowerShell V4를 사용한 Amazon FSx 예제
- Tools for PowerShell V4를 사용한 Amazon Glacier 예제
- AWS Glue Tools for PowerShell V4를 사용한 예제
- AWS Health Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 IAM 예제
- Tools for PowerShell V4를 사용한 Kinesis 예제
- Tools for PowerShell V4를 사용한 Lambda 예제
- Tools for PowerShell V4를 사용한 Amazon ML 예제
- Tools for PowerShell V4를 사용한 Macie 예제
- AWS 가격표 Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 Resource Groups 예제
- Tools for PowerShell V4를 사용한 Resource Groups Tagging API 예제
- Tools for PowerShell V4를 사용한 Route 53 예제
- Tools for PowerShell V4를 사용한 Amazon S3 예제
- Tools for PowerShell V4를 사용한 Security Hub CSPM 예제
- Tools for PowerShell V4를 사용한 Amazon SES 예제
- Tools for PowerShell V4를 사용한 Amazon SES API v2 예제
- Tools for PowerShell V4를 사용한 Amazon SNS 예제

- Tools for PowerShell V4를 사용한 Amazon SQS 예제
- AWS STS Tools for PowerShell V4를 사용한 예제
- 지원 Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 Systems Manager 예제
- Tools for PowerShell V4를 사용한 Amazon Translate 예제
- AWS WAFV2 Tools for PowerShell V4를 사용한 예제
- Tools for PowerShell V4를 사용한 WorkSpaces 예제

Tools for PowerShell V4를 사용한 ACM 예제

다음 코드 예제에서는 ACM과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-ACMCertificate`

다음 코드 예시는 `Get-ACMCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인증서의 ARN을 사용하여 인증서와 해당 체인을 반환하는 방법을 보여줍니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCertificate를 참조하세요.

`Get-ACMCertificateDetail`

다음 코드 예시는 `Get-ACMCertificateDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 인증서의 세부 정보를 반환합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCertificate를 참조하세요.

`Get-ACMCertificateList`

다음 코드 예시는 `Get-ACMCertificateList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 인증서 ARN 목록과 각 도메인 이름을 검색합니다. cmdlet은 자동으로 페이지를 지정하여 모든 ARN을 검색합니다. 페이지 지정을 수동으로 제어하려면 -MaxItem 파라미터를 사용하여 각 서비스 직접 호출에 대해 반환되는 인증서 ARN 수를 제어하고 -NextToken 파라미터를 사용하여 각 직접 호출의 시작점을 나타냅니다.

```
Get-ACMCertificateList
```

출력:

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

예제 2: 제공된 상태와 일치하는 인증서 상태의 모든 인증서 ARN 목록을 검색합니다.

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

예제 3: 이 예제에서는 키 유형이 RSA_2048이고, 확장된 키 사용 또는 용도가 CODE_SIGNING인 us-east-1 리전의 모든 인증서 목록을 반환합니다. ListCertificates Filters API 참조 주제: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html에서 이러한 필터링 파라미터의 값을 찾아볼 수 있습니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListCertificates를 참조하세요.

`New-ACMCertificate`

다음 코드 예시는 `New-ACMCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 새 인증서를 만듭니다. 서비스는 새 인증서의 ARN을 반환합니다.

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

출력:

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

예제 2: 새 인증서를 만듭니다. 서비스는 새 인증서의 ARN을 반환합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RequestCertificate를 참조하세요.

`Remove-ACMCertificate`

다음 코드 예시는 `Remove-ACMCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 제공된 ARN 및 연결된 프라이빗 키로 식별되는 인증서를 삭제합니다. cmdlet에서는 작업을 진행하기 전에 확인 프롬프트를 표시합니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가하세요.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCertificate를 참조하세요.

`Send-ACMValidationEmail`

다음 코드 예시는 `Send-ACMValidationEmail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 'www.example.com'에 대한 도메인 소유권을 검증하기 위한 이메일을 보내도록 요청합니다. 쉘의 $ConfirmPreference가 '중간' 이하로 설정된 경우 진행하기 전에 cmdlet에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResendValidationEmail을 참조하세요.

Tools for PowerShell V4를 사용한 Application Auto Scaling 예제

다음 코드 예제에서는 Application Auto Scaling과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-AASScalableTarget`

다음 코드 예시는 `Add-AASScalableTarget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 규모 조정 가능 대상을 등록하거나 업데이트합니다. 규모 조정 가능 대상은 Application Auto Scaling에서 스케일 아웃 및 스케일 인할 수 있는 리소스입니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterScalableTarget을 참조하세요.

`Get-AASScalableTarget`

다음 코드 예시는 `Get-AASScalableTarget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네임스페이스의 Application Auto Scaling 규모 조정 가능 대상에 대한 정보를 제공합니다.

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

출력:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScalableTargets을 참조하세요.

`Get-AASScalingActivity`

다음 코드 예시는 `Get-AASScalingActivity`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이전 6주 동안 지정된 서비스 네임스페이스의 조정 활동에 대한 설명이 포함된 정보를 제공합니다.

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

출력:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in state
 ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScalingActivities를 참조하세요.

`Get-AASScalingPolicy`

다음 코드 예시는 `Get-AASScalingPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 지정된 서비스 네임스페이스에 대한 Application Auto Scaling 조정 정책을 설명합니다.

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

출력:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-out-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                  : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-in
PolicyName                              : default-scale-in
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
```

```
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScalingPolicies를 참조하세요.

`Get-AASScheduledAction`

다음 코드 예시는 `Get-AASScheduledAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 실행되지 않았거나 종료 시간에 도달하지 않은 Auto Scaling 그룹에 예약된 작업을 나열합니다.

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

출력:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScheduledActions을 참조하세요.

Remove-AASScalableTarget

다음 코드 예시는 Remove-AASScalableTarget의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 Application Auto Scaling 규모 조정 가능 대상의 등록을 취소합니다. 규모 조정 가능 대상의 등록을 취소하면 연결된 조정 정책이 삭제됩니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterScalableTarget을 참조하세요.

Remove-AASScalingPolicy

다음 코드 예시는 Remove-AASScalingPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Application Auto Scaling 규모 조정 가능 대상에 대해 지정된 조정 정책을 삭제합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteScalingPolicy를 참조하세요.

Remove-AASScheduledAction

다음 코드 예시는 Remove-AASScheduledAction의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Application Auto Scaling 규모 조정 가능 대상에 대해 지정된 예약된 작업을 삭제합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteScheduledAction을 참조하세요.

Set-AASScalingPolicy

다음 코드 예시는 Set-AASScalingPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 Application Auto Scaling 규모 조정 가능 대상에 대한 정책을 생성 또는 업데이트합니다. 각 규모 조정 가능 대상은 서비스 네임스페이스, 리소스 ID 및 규모 조정 가능 차원으로 식별됩니다.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
```

```
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutScalingPolicy를 참조하세요.

`Set-AASScheduledAction`

다음 코드 예시는 `Set-AASScheduledAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet에서는 Application Auto Scaling 규모 조정 가능 대상에 대한 예약된 작업을 생성 또는 업데이트합니다. 각 규모 조정 가능 대상은 서비스 네임스페이스, 리소스 ID 및 규모 조정 가능 차원으로 식별됩니다.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutScheduledAction을 참조하세요.

Tools for PowerShell V4를 사용한 WorkSpaces 애플리케이션 예제

다음 코드 예제에서는 WorkSpaces 애플리케이션에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

Add-APSResourceTag

다음 코드 예시는 Add-APSResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 리소스에 리소스 태그를 추가합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResource를 참조하세요.

Copy-APSImage

다음 코드 예시는 Copy-APSImage의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 이미지를 다른 리전에 복사합니다.

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

출력:

```
TestImageCopy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CopyImage를 참조하세요.

`Disable-APSUSer`

다음 코드 예시는 `Disable-APSUSer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 USERPOOL에서 사용자를 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableUser를 참조하세요.

`Enable-APSUser`

다음 코드 예시는 `Enable-APSUser`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 USERPOOL에서 비활성화된 사용자를 활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableUser를 참조하세요.

`Get-APSAssociatedFleetList`

다음 코드 예시는 `Get-APSAssociatedFleetList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택과 연결된 플릿을 표시합니다.

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

출력:

```
PowershellFleet
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAssociatedFleets을 참조하세요.

Get-APSAssociatedStackList

다음 코드 예시는 Get-APSAssociatedStackList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿과 연결된 스택을 표시합니다.

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

출력:

```
PowershellStack
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAssociatedStacks을 참조하세요.

Get-APSDirectoryConfigList

다음 코드 예시는 Get-APSDirectoryConfigList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream에서 생성된 디렉터리 구성을 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDirectoryConfigs를 참조하세요.

Get-APSFleetList

다음 코드 예시는 Get-APSFleetList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿의 세부 정보를 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeFleets을 참조하세요.

Get-APSImageBuilderList

다음 코드 예시는 Get-APSImageBuilderList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 ImageBuilder의 세부 정보를 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImageBuilders를 참조하세요.

`Get-APSImageList`

다음 코드 예시는 `Get-APSImageList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 프라이빗 AppStream 이미지를 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImages를 참조하세요.

`Get-APSImagePermission`

다음 코드 예시는 `Get-APSImagePermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 공유 AppStream 이미지에 대한 이미지 권한을 표시합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImagePermissions을 참조하세요.

`Get-APSSessionList`

다음 코드 예시는 `Get-APSSessionList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿에 대한 세션 목록을 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSessions을 참조하세요.

`Get-APSStackList`

다음 코드 예시는 `Get-APSStackList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에는 AppStream Stack 목록이 표시됩니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStacks를 참조하세요.

`Get-APSTagsForResourceList`

다음 코드 예시는 `Get-APSTagsForResourceList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 리소스의 태그를 표시합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

`Get-APSUsageReportSubscription`

다음 코드 예시는 `Get-APSUsageReportSubscription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStreamUsageReport 구성 세부 정보를 표시합니다.

```
Get-APSUsageReportSubscription
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeUsageReportSubscriptions을 참조하세요.

Get-APSUser

다음 코드 예시는 Get-APSUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 활성화된 상태의 사용자 목록을 표시합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeUsers를 참조하세요.

Get-APSUserStackAssociation

다음 코드 예시는 Get-APSUserStackAssociation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택에 할당된 사용자 목록을 표시합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeUserStackAssociations을 참조하세요.

New-APSDirectoryConfig

다음 코드 예시는 New-APSDirectoryConfig의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream에 디렉터리 구성을 생성합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDirectoryConfig를 참조하세요.

New-APSFleet

다음 코드 예시는 New-APSFleet의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 새로운 AppStream 플릿을 생성합니다.

```
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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFleet을 참조하세요.

New-APSImageBuilder

다음 코드 예시는 New-APSImageBuilder의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream에 Image Builder를 생성합니다.

```
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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateImageBuilder를 참조하세요.

New-APSImageBuilderStreamingURL

다음 코드 예시는 New-APSImageBuilderStreamingURL의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 유효 기간이 2시간인 ImageBuilder 스트리밍 URL을 생성합니다.

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

출력:

```
Expires              StreamingURL
-------              ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateImageBuilderStreamingURL을 참조하세요.

New-APSStack

다음 코드 예시는 New-APSStack의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 새로운 AppStream 스택을 생성합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateStack을 참조하세요.

New-APSStreamingURL

다음 코드 예시는 New-APSStreamingURL의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 Stack의 스트리밍 URL을 생성합니다.

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

출력:

```
Expires                StreamingURL
```

```
-------                  ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateStreamingURL을 참조하세요.

New-APSUsageReportSubscription

다음 코드 예시는 New-APSUsageReportSubscription의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 사용 보고서를 활성화합니다.

```
New-APSUsageReportSubscription
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateUsageReportSubscription을 참조하세요.

New-APSUser

다음 코드 예시는 New-APSUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 USERPOOL에 사용자를 생성합니다.

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

출력:

```
Test@lab.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateUser를 참조하세요.

`Register-APSFleet`

다음 코드 예시는 `Register-APSFleet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택에 플릿을 등록합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateFleet을 참조하세요.

`Register-APSUserStackBatch`

다음 코드 예시는 `Register-APSUserStackBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 USERPOOL의 사용자에게 스택을 할당합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchAssociateUserStack을 참조하세요.

`Remove-APSDirectoryConfig`

다음 코드 예시는 `Remove-APSDirectoryConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream Directory 구성을 제거합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDirectoryConfig를 참조하세요.

`Remove-APSFleet`

다음 코드 예시는 `Remove-APSFleet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 플릿을 삭제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFleet을 참조하세요.

`Remove-APSImage`

다음 코드 예시는 `Remove-APSImage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 이미지를 삭제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteImage를 참조하세요.

`Remove-APSImageBuilder`

다음 코드 예시는 `Remove-APSImageBuilder`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 ImageBuilder를 삭제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteImageBuilder를 참조하세요.

`Remove-APSImagePermission`

다음 코드 예시는 `Remove-APSImagePermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 이미지의 권한을 제거합니다.

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteImagePermissions을 참조하세요.

`Remove-APSResourceTag`

다음 코드 예시는 `Remove-APSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 리소스에서 리소스 태그를 제거합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagResource를 참조하세요.

`Remove-APSStack`

다음 코드 예시는 `Remove-APSStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택을 삭제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteStack을 참조하세요.

`Remove-APSUsageReportSubscription`

다음 코드 예시는 `Remove-APSUsageReportSubscription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 사용 보고서 구독을 비활성화합니다.

```
Remove-APSUsageReportSubscription
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteUsageReportSubscription을 참조하세요.

`Remove-APSUser`

다음 코드 예시는 `Remove-APSUser`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 USERPOOL에서 사용자를 삭제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteUser를 참조하세요.

`Revoke-APSSession`

다음 코드 예시는 `Revoke-APSSession`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 플릿에 대한 세션을 취소합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ExpireSession을 참조하세요.

`Start-APSFleet`

다음 코드 예시는 `Start-APSFleet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿을 시작합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartFleet을 참조하세요.

`Start-APSImageBuilder`

다음 코드 예시는 `Start-APSImageBuilder`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 ImageBuilder를 시작합니다.

```
Start-APSImageBuilder -Name TestImage
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartImageBuilder를 참조하세요.

Stop-APSFleet

다음 코드 예시는 Stop-APSFleet의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿을 중지합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopFleet을 참조하세요.

`Stop-APSImageBuilder`

다음 코드 예시는 `Stop-APSImageBuilder`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 ImageBuilder를 중지합니다.

```
Stop-APSImageBuilder -Name TestImage
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopImageBuilder를 참조하세요.

`Unregister-APSFleet`

다음 코드 예시는 `Unregister-APSFleet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택에서 플릿 등록을 취소합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisassociateFleet을 참조하세요.

`Unregister-APSUserStackBatch`

다음 코드 예시는 `Unregister-APSUserStackBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 할당된 스택에서 사용자를 제거합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchDisassociateUserStack을 참조하세요.

`Update-APSDirectoryConfig`

다음 코드 예시는 `Update-APSDirectoryConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream에서 생성된 디렉터리 구성을 업데이트합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateDirectoryConfig를 참조하세요.

`Update-APSFleet`

다음 코드 예시는 `Update-APSFleet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 플릿의 속성을 업데이트합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateFleet을 참조하세요.

`Update-APSImagePermission`

다음 코드 예시는 `Update-APSImagePermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 AppStream 이미지를 다른 계정에 공유합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateImagePermissions을 참조하세요.

`Update-APSStack`

다음 코드 예시는 `Update-APSStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 스택에서 애플리케이션 설정 지속성과 홈 폴더를 업데이트(활성화)합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateStack을 참조하세요.

Tools for PowerShell V4를 사용한 Aurora 예제

다음 코드 예제에서는 Aurora와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-RDSOrderableDBInstanceOption`

다음 코드 예시는 `Get-RDSOrderableDBInstanceOption`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예는 AWS 리전에서 특정 DB 인스턴스 클래스를 지원하는 DB 엔진 버전을 나열합니다.

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

예 2: 이 예는 AWS 리전에서 특정 DB 엔진 버전을 지원하는 DB 인스턴스 클래스를 나열합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeOrderableDBInstanceOptions를 참조하세요.

Tools for PowerShell V4를 사용한 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-ASLoadBalancer`

다음 코드 예시는 `Add-ASLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 로드 밸런서를 연결합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachLoadBalancers를 참조하세요.

`Complete-ASLifecycleAction`

다음 코드 예시는 `Complete-ASLifecycleAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 수명 주기 작업을 완료합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CompleteLifecycleAction을 참조하세요.

`Disable-ASMetricsCollection`

다음 코드 예시는 `Disable-ASMetricsCollection`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 지정된 지표의 모니터링을 비활성화합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 지표의 모니터링을 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableMetricsCollection을 참조하세요.

`Dismount-ASInstance`

다음 코드 예시는 `Dismount-ASInstance`의 사용 방법을 보여줍니다.

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리하고 Auto Scaling이 대체 인스턴스를 시작하지 않도록 원하는 용량을 줄입니다.

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

출력:

```
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        :
```

예제 2: 이 예제에서는 원하는 용량을 줄이지 않고 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리합니다. Auto Scaling이 대체 인스턴스를 시작합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachInstances를 참조하세요.

`Dismount-ASLoadBalancer`

다음 코드 예시는 `Dismount-ASLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 로드 밸런서를 분리합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachLoadBalancers를 참조하세요.

`Enable-ASMetricsCollection`

다음 코드 예시는 `Enable-ASMetricsCollection`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 지정된 지표의 모니터링을 활성화합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 지표의 모니터링을 활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableMetricsCollection을 참조하세요.

`Enter-ASStandby`

다음 코드 예시는 `Enter-ASStandby`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 대기 모드로 전환하고, Auto Scaling이 대체 인스턴스를 시작하지 않도록 원하는 용량을 줄입니다.

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

출력:

```
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        :
```

예제 2: 이 예제에서는 원하는 용량을 줄이지 않고 지정된 인스턴스를 대기 모드로 전환합니다. Auto Scaling이 대체 인스턴스를 시작합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnterStandby를 참조하세요.

`Exit-ASStandby`

다음 코드 예시는 `Exit-ASStandby`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 대기 모드에서 해제합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ExitStandby를 참조하세요.

`Get-ASAccountLimit`

다음 코드 예시는 `Get-ASAccountLimit`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS 계정의 Auto Scaling 리소스 제한을 설명합니다.

```
Get-ASAccountLimit
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAccountLimits을 참조하세요.

Get-ASAdjustmentType

다음 코드 예시는 Get-ASAdjustmentType의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 조정 유형을 설명합니다.

```
Get-ASAdjustmentType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAdjustmentTypes을 참조하세요.

Get-ASAutoScalingGroup

다음 코드 예시는 Get-ASAutoScalingGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling 그룹의 이름을 나열합니다.

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

출력:

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹을 설명합니다.

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

출력:

```
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
```

예제 3: 이 예제에서는 지정된 두 개의 Auto Scaling 그룹을 설명합니다.

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

예제 4: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 Auto Scaling 인스턴스를 설명합니다.

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

예제 5: 이 예제에서는 모든 Auto Scaling 그룹을 설명합니다.

```
Get-ASAutoScalingGroup
```

예제 6: 이 예제에서는 지정된 Auto Scaling 그룹의 LaunchTemplate를 설명합니다. 이 예제에서는 "인스턴스 구매 옵션"이 "시작 템플릿 준수"로 설정되어 있다고 가정합니다. 이 옵션이 "구매 옵션 및 인스턴스 유형 결합"으로 설정된 경우 "MixedInstancesPolicy.LaunchTemplate" 속성을 사용하여 LaunchTemplate에 액세스할 수 있습니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAutoScalingGroups을 참조하세요.

`Get-ASAutoScalingInstance`

다음 코드 예시는 `Get-ASAutoScalingInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling 인스턴스의 ID를 나열합니다.

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

출력:

```
InstanceId
```

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

예제 2: 이 예제에서는 지정된 Auto Scaling 인스턴스를 설명합니다.

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

출력:

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

예제 3: 이 예제에서는 지정된 두 Auto Scaling 인스턴스를 설명합니다.

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

예제 4: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 Auto Scaling 인스턴스를 설명합니다.

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

예제 5: 이 예제에서는 모든 Auto Scaling 인스턴스를 설명합니다.

```
Get-ASAutoScalingInstance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAutoScalingInstances를 참조하세요.

`Get-ASAutoScalingNotificationType`

다음 코드 예시는 `Get-ASAutoScalingNotificationType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 알림 유형을 나열합니다.

```
Get-ASAutoScalingNotificationType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAutoScalingNotificationTypes을 참조하세요.

`Get-ASLaunchConfiguration`

다음 코드 예시는 `Get-ASLaunchConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 시작 구성의 이름을 나열합니다.

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

출력:

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

예제 2: 이 예제에서는 지정된 시작 구성을 설명합니다.

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

출력:

```
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                     :
```

예제 3: 이 예제에서는 지정된 두 가지 시작 구성을 설명합니다.

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

예제 4: 이 예제에서는 모든 시작 구성을 설명합니다.

```
Get-ASLaunchConfiguration
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLaunchConfigurations을 참조하세요.

`Get-ASLifecycleHook`

다음 코드 예시는 `Get-ASLifecycleHook`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 수명 주기 후크를 설명합니다.

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

출력:

```
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
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 수명 주기 후크를 설명합니다.

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

예제 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 수명 주기 후크를 설명합니다.

```
Get-ASLifecycleHook
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLifecycleHooks를 참조하세요.

`Get-ASLifecycleHookType`

다음 코드 예시는 `Get-ASLifecycleHookType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 수명 주기 후크의 유형을 나열합니다.

```
Get-ASLifecycleHookType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLifecycleHookTypes을 참조하세요.

`Get-ASLoadBalancer`

다음 코드 예시는 `Get-ASLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 로드 밸런서를 설명합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancers를 참조하세요.

`Get-ASMetricCollectionType`

다음 코드 예시는 `Get-ASMetricCollectionType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 지표 컬렉션 유형을 나열합니다.

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

출력:

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

예제 2: 이 예제에서는 해당 세부 수준을 나열합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMetricCollectionTypes을 참조하세요.

`Get-ASNotificationConfiguration`

다음 코드 예시는 `Get-ASNotificationConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 Auto Scaling 그룹과 연결된 알림 작업을 설명합니다.

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

출력:

```
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
```

예제 2: 이 예제에서는 모든 Auto Scaling 그룹과 연결된 알림 작업을 설명합니다.

```
Get-ASNotificationConfiguration
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeNotificationConfigurations을 참조하세요.

Get-ASPolicy

다음 코드 예시는 Get-ASPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹의 모든 정책을 설명합니다.

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

출력:

```
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         : {}
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 설명합니다.

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

예제 3: 이 예제에서는 모든 Auto Scaling 그룹에 대한 모든 정책을 설명합니다.

```
Get-ASPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePolicies를 참조하세요.

Get-ASScalingActivity

다음 코드 예시는 Get-ASScalingActivity의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹의 지난 6주 동안의 조정 활동을 설명합니다.

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

출력:

```
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        :
```

예제 2: 이 예제에서는 지정된 조정 활동을 설명합니다.

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

예제 3: 이 예제에서는 모든 Auto Scaling 그룹의 지난 6주 동안의 조정 활동을 설명합니다.

```
Get-ASScalingActivity
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScalingActivities를 참조하세요.

`Get-ASScalingProcessType`

다음 코드 예시는 `Get-ASScalingProcessType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 프로세스 유형을 나열합니다.

```
Get-ASScalingProcessType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScalingProcessTypes을 참조하세요.

`Get-ASScheduledAction`

다음 코드 예시는 `Get-ASScheduledAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 예약된 조정 작업을 설명합니다.

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

출력:

```
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
```

예제 2: 이 예제에서는 지정된 예약 조정 작업을 설명합니다.

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

예제 3: 이 예제에서는 지정된 시간에 시작하는 예약된 조정 작업을 설명합니다.

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

예제 4: 이 예제에서는 지정된 시간에 종료되는 예약된 조정 작업을 설명합니다.

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

예제 5: 이 예제에서는 모든 Auto Scaling 그룹에 예약된 조정 작업을 설명합니다.

```
Get-ASScheduledAction
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScheduledActions을 참조하세요.

Get-ASTag

다음 코드 예시는 Get-ASTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 키 값이 'myTag' 또는 'myTag2'인 태그를 설명합니다. 필터 이름의 가능한 값은 'auto-scaling-group', 'key', 'value', 'propagate-at-launch'입니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

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

출력:

```
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
```

예제 2: PowerShell 버전 2에서 Filter 파라미터에 대해 필터를 생성하려면 New-Object를 사용해야 합니다.

```
$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 )
```

예제 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 태그를 설명합니다.

```
Get-ASTag
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

`Get-ASTerminationPolicyType`

다음 코드 예시는 `Get-ASTerminationPolicyType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 종료 정책을 나열합니다.

```
Get-ASTerminationPolicyType
```

출력:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTerminationPolicyTypes을 참조하세요.

`Mount-ASInstance`

다음 코드 예시는 `Mount-ASInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 인스턴스를 연결합니다. Auto Scaling은 Auto Scaling 그룹의 원하는 용량을 자동으로 늘립니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachInstances를 참조하세요.

`New-ASAutoScalingGroup`

다음 코드 예시는 `New-ASAutoScalingGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름과 속성으로 Auto Scaling 그룹을 생성합니다. 원하는 기본 용량이 최소 크기입니다. 따라서 이 Auto Scaling 그룹은 지정된 두 가용 영역에 하나씩 있는 두 개의 인스턴스를 시작합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAutoScalingGroup을 참조하세요.

`New-ASLaunchConfiguration`

다음 코드 예시는 `New-ASLaunchConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'my-lc'라는 시작 구성을 생성합니다. 이 시작 구성을 사용하는 Auto Scaling 그룹에서 시작한 EC2 인스턴스는 지정된 인스턴스 유형, AMI, 보안 그룹, IAM 역할을 사용합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLaunchConfiguration을 참조하세요.

`Remove-ASAutoScalingGroup`

다음 코드 예시는 `Remove-ASAutoScalingGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 실행 중인 인스턴스가 없는 경우 지정된 Auto Scaling 그룹을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

예제 3: 이 예제에서는 지정된 Auto Scaling 그룹을 삭제하고 포함된 실행 중인 인스턴스를 모두 종료합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAutoScalingGroup을 참조하세요.

`Remove-ASLaunchConfiguration`

다음 코드 예시는 `Remove-ASLaunchConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Auto Scaling 그룹에 연결되지 않은 경우 지정된 시작 구성을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLaunchConfiguration을 참조하세요.

Remove-ASLifecycleHook

다음 코드 예시는 Remove-ASLifecycleHook의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 수명 주기 후크를 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLifecycleHook를 참조하세요.

Remove-ASNotificationConfiguration

다음 코드 예시는 Remove-ASNotificationConfiguration의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 알림 작업을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteNotificationConfiguration을 참조하세요.

`Remove-ASPolicy`

다음 코드 예시는 `Remove-ASPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePolicy를 참조하세요.

`Remove-ASScheduledAction`

다음 코드 예시는 `Remove-ASScheduledAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 예약된 작업을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteScheduledAction을 참조하세요.

Remove-ASTag

다음 코드 예시는 Remove-ASTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 태그를 제거합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

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

출력:

```
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"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

예제 3: PowerShell 버전 2에서 Tag 파라미터에 대해 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTags를 참조하세요.

Resume-ASProcess

다음 코드 예시는 Resume-ASProcess의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 재개합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 일시 중지된 모든 Auto Scaling 프로세스를 재개합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResumeProcesses를 참조하세요.

`Set-ASDesiredCapacity`

다음 코드 예시는 `Set-ASDesiredCapacity`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹의 크기를 설정합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹의 크기를 설정하고 새 크기로 크기 조정하기 전에 휴지 기간이 완료될 때까지 기다립니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetDesiredCapacity를 참조하세요.

`Set-ASInstanceHealth`

다음 코드 예시는 `Set-ASInstanceHealth`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스의 상태를 'Unhealthy'로 설정하여 서비스를 중단합니다. Auto Scaling은 인스턴스를 종료하고 교체합니다.

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

예제 2: 이 예제에서는 지정된 인스턴스의 상태를 'Healthy'로 설정하여 서비스를 유지합니다. Auto Scaling 그룹의 상태 확인 유예 기간이 적용되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetInstanceHealth를 참조하세요.

`Set-ASInstanceProtection`

다음 코드 예시는 `Set-ASInstanceProtection`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에 대한 인스턴스 보호를 활성화합니다.

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

예제 2: 이 예제에서는 지정된 인스턴스에 대한 인스턴스 보호를 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetInstanceProtection을 참조하세요.

`Set-ASTag`

다음 코드 예시는 `Set-ASTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 단일 태그를 추가합니다. 태그 키는 'myTag '이고 태그 값은 'myTagValue'입니다. Auto Scaling은 이 태그를 Auto Scaling 그룹에서 시작한 후속 EC2 인스턴스로 전파합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

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

예제 2: PowerShell 버전 2에서 Tag 파라미터에 대해 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateOrUpdateTags를 참조하세요.

`Start-ASPolicy`

다음 코드 예시는 `Start-ASPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 실행합니다.

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

예제 2: 이 예제에서는 휴지 기간이 완료될 때까지 기다린 후 지정된 Auto Scaling 그룹에 대해 지정된 정책을 실행합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ExecutePolicy를 참조하세요.

`Stop-ASInstanceInAutoScalingGroup`

다음 코드 예시는 `Stop-ASInstanceInAutoScalingGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 종료하고 Auto Scaling이 대체 인스턴스를 시작하지 않도록 Auto Scaling 그룹의 원하는 용량을 줄입니다.

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

출력:

```
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        :
```

예제 2: 이 예제에서는 Auto Scaling 그룹의 원하는 용량을 줄이지 않고 지정된 인스턴스를 종료합니다. Auto Scaling이 대체 인스턴스를 시작합니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TerminateInstanceInAutoScalingGroup을 참조하세요.

`Suspend-ASProcess`

다음 코드 예시는 `Suspend-ASProcess`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 일시 중지합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 Auto Scaling 프로세스를 일시 중지합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SuspendProcesses를 참조하세요.

`Update-ASAutoScalingGroup`

다음 코드 예시는 `Update-ASAutoScalingGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹을 최소 및 최대 크기로 업데이트합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹의 기본 휴지 기간을 업데이트합니다.

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

예제 3: 이 예제에서는 지정된 Auto Scaling 그룹의 가용 영역을 업데이트합니다.

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

예제 4: 이 예제에서는 Elastic Load Balancing 상태 확인을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAutoScalingGroup을 참조하세요.

`Write-ASLifecycleActionHeartbeat`

다음 코드 예시는 `Write-ASLifecycleActionHeartbeat`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 수명 주기 작업에 대한 하트비트를 기록합니다. 이렇게 하면 사용자 지정 작업을 완료할 때까지 인스턴스가 보류 상태로 유지됩니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RecordLifecycleActionHeartbeat를 참조하세요.

`Write-ASLifecycleHook`

다음 코드 예시는 `Write-ASLifecycleHook`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 수명 주기 후크를 추가합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutLifecycleHook를 참조하세요.

`Write-ASNotificationConfiguration`

다음 코드 예시는 `Write-ASNotificationConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹이 EC2 인스턴스를 시작할 때 지정된 SNS 주제에 알림을 보내도록 구성합니다.

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

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹이 EC2 인스턴스를 시작하거나 종료할 때 지정된 SNS 주제에 알림을 보내도록 구성합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutNotificationConfiguration을 참조하세요.

`Write-ASScalingPolicy`

다음 코드 예시는 `Write-ASScalingPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 정책을 추가합니다. 지정된 조정 유형은 ScalingAdjustment 파라미터를 해석하는 방법을 결정합니다. 'ChangeInCapacity'를 사용하면 양수 값이 지정된 인스턴스 수만큼 용량을 늘리고 음수 값은 지정된 인스턴스 수만큼 용량을 줄입니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutScalingPolicy를 참조하세요.

`Write-ASScheduledUpdateGroupAction`

다음 코드 예시는 `Write-ASScheduledUpdateGroupAction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 일회성 예약 작업을 생성하거나 업데이트하여 지정된 시작 시간에 원하는 용량을 변경합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutScheduledUpdateGroupAction을 참조하세요.

AWS Budgets Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Budgets.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`New-BGTBudget`

다음 코드 예시는 `New-BGTBudget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 예산 및 시간 제약 조건을 지정하고 이메일 알림이 포함된 새 예산을 생성합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateBudget을 참조하세요.

AWS Cloud9 Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Cloud9.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-C9EnvironmentData`

다음 코드 예시는 `Get-C9EnvironmentData`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Cloud9 개발 환경에 대한 정보를 가져옵니다.

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

출력:

```
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
```

예제 2:이 예제에서는 지정된 AWS Cloud9 개발 환경의 수명 주기 상태에 대한 정보를 가져옵니다.

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

출력:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEnvironments를 참조하세요.

`Get-C9EnvironmentList`

다음 코드 예시는 `Get-C9EnvironmentList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 사용 가능한 AWS Cloud9 개발 환경 식별자 목록을 가져옵니다.

```
Get-C9EnvironmentList
```

출력:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListEnvironments를 참조하세요.

`Get-C9EnvironmentMembershipList`

다음 코드 예시는 `Get-C9EnvironmentMembershipList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Cloud9 개발 환경의 환경 멤버에 대한 정보를 가져옵니다.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

출력:

```
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
```

예제 2:이 예제에서는 지정된 AWS Cloud9 개발 환경의 소유자에 대한 정보를 가져옵니다.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

출력:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

예제 3:이 예제에서는 multiple AWS Cloud9 개발 환경의 지정된 환경 멤버에 대한 정보를 가져옵니다.

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

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEnvironmentMemberships을 참조하세요.

`Get-C9EnvironmentStatus`

다음 코드 예시는 `Get-C9EnvironmentStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 지정된 AWS Cloud9 개발 환경에 대한 상태 정보를 가져옵니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEnvironmentStatus를 참조하세요.

`New-C9EnvironmentEC2`

다음 코드 예시는 `New-C9EnvironmentEC2`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 설정으로 AWS Cloud9 개발 환경을 생성하고 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 시작한 다음 인스턴스에서 환경으로 연결합니다.

```
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
```

출력:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateEnvironmentEc2를 참조하세요.

`New-C9EnvironmentMembership`

다음 코드 예시는 `New-C9EnvironmentMembership`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 환경 멤버를 지정된 AWS Cloud9 개발 환경에 추가합니다.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

출력:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateEnvironmentMembership을 참조하세요.

`Remove-C9Environment`

다음 코드 예시는 `Remove-C9Environment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Cloud9 개발 환경을 삭제합니다. 환경에 Amazon EC2 인스턴스가 연결된 경우 해당 인스턴스도 종료됩니다.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteEnvironment를 참조하세요.

`Remove-C9EnvironmentMembership`

다음 코드 예시는 `Remove-C9EnvironmentMembership`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Cloud9 개발 환경에서 지정된 환경 멤버를 삭제합니다.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteEnvironmentMembership을 참조하세요.

`Update-C9Environment`

다음 코드 예시는 `Update-C9Environment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 기존 AWS Cloud9 개발 환경의 지정된 설정을 변경합니다.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateEnvironment를 참조하세요.

`Update-C9EnvironmentMembership`

다음 코드 예시는 `Update-C9EnvironmentMembership`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Cloud9 개발 환경에 대해 지정된 기존 환경 멤버의 설정을 변경합니다.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

출력:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateEnvironmentMembership을 참조하세요.

CloudFormation Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CloudFormation.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-CFNStack`

다음 코드 예시는 `Get-CFNStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 모든 사용자 스택을 설명하는 스택 인스턴스 컬렉션을 반환합니다.

```
Get-CFNStack
```

예 2: 지정된 스택을 설명하는 스택 인스턴스를 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStacks를 참조하세요.

`Get-CFNStackEvent`

다음 코드 예시는 `Get-CFNStackEvent`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스택에 대한 모든 스택 관련 이벤트를 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStackEvents를 참조하세요.

`Get-CFNStackResource`

다음 코드 예시는 `Get-CFNStackResource`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 논리적 ID "MyDBInstance"로 지정된 스택과 연결된 템플릿에서 식별된 리소스의 설명을 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStackResource를 참조하세요.

`Get-CFNStackResourceList`

다음 코드 예시는 `Get-CFNStackResourceList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 스택과 연결된 최대 100개의 AWS 리소스에 대한 리소스 설명을 반환합니다. 스택과 연결된 모든 리소스의 세부 정보를 얻으려면 결과의 수동 페이징도 지원하는 Get-CFNStackResourceSummary를 사용하세요.

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

예 2: 논리적 ID "Ec2Instance"로 지정된 스택과 연결된 템플릿에서 식별된 Amazon EC2 인스턴스의 설명을 반환합니다.

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

예 3: 인스턴스 ID "i-123456"으로 식별되는 Amazon EC2 인스턴스를 포함하는 스택과 연결된 최대 100개의 리소스에 대한 설명을 반환합니다. 스택과 연결된 모든 리소스의 세부 정보를 얻으려면 결과의 수동 페이징도 지원하는 Get-CFNStackResourceSummary를 사용하세요.

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

예 4: 논리적 ID "Ec2Instance"로 지정된 스택의 템플릿에서 식별된 Amazon EC2 인스턴스의 설명을 반환합니다. 스택은 포함된 리소스의 물리적 리소스 ID를 사용하여 식별됩니다. 이 경우에는 인스턴스 ID가 "i-123456"인 Amazon EC2 인스턴스입니다. 템플릿 콘텐츠에 따라 다른 물리적 리소스를 사용하여 스택을 식별할 수도 있습니다(예: Amazon S3 버킷).

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStackResources를 참조하세요.

Get-CFNStackResourceSummary

다음 코드 예시는 Get-CFNStackResourceSummary의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스택과 연결된 모든 리소스에 대한 설명을 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListStackResources를 참조하세요.

Get-CFNStackSummary

다음 코드 예시는 Get-CFNStackSummary의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 모든 스택에 대한 요약 정보를 반환합니다.

```
Get-CFNStackSummary
```

예 2: 현재 생성 중인 모든 스택에 대한 요약 정보를 반환합니다.

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

예 3: 현재 생성 중이거나 업데이트 중인 모든 스택에 대한 요약 정보를 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListStacks를 참조하세요.

Get-CFNTemplate

다음 코드 예시는 Get-CFNTemplate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스택과 연결된 템플릿을 반환합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetTemplate을 참조하세요.

`Measure-CFNTemplateCost`

다음 코드 예시는 `Measure-CFNTemplateCost`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 템플릿을 실행하는 데 필요한 리소스를 설명하는 쿼리 문자열과 함께 AWS 월별 단순 계산기 URL을 반환합니다. 템플릿은 지정된 Amazon S3 URL에서 가져오고 단일 사용자 지정 파라미터가 적용됩니다. 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

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

예제 2: 템플릿을 실행하는 데 필요한 리소스를 설명하는 쿼리 문자열이 포함된 AWS 월별 단순 계산기 URL을 반환합니다. 템플릿은 제공된 콘텐츠에서 구문 분석되고 사용자 지정 매개 변수가 적용됩니다. 이 예에서는 템플릿 콘텐츠가 두 개의 파라미터 'KeyName'과 'InstanceType'을 선언했다고 가정합니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

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

예제 3: New-Object를 사용하여 템플릿 파라미터 세트를 빌드하고 템플릿을 실행하는 데 필요한 리소스를 설명하는 쿼리 문자열과 함께 AWS Simple Monthly Calculator URL을 반환합니다. 템플릿은 제공된 콘텐츠에서 구문 분석되고 사용자 지정 매개 변수를 포함합니다. 이 예에서는 템플릿 콘텐츠가 두 개의 파라미터 'KeyName'과 'InstanceType'을 선언했다고 가정합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EstimateTemplateCost를 참조하세요.

New-CFNStack

다음 코드 예시는 New-CFNStack의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 이름으로 새 스택을 생성합니다. 템플릿은 사용자 지정 파라미터를 사용하여 제공된 콘텐츠에서 구문 분석됩니다. 여기서, 'PK1' 및 'PK2'는 템플릿 콘텐츠에 선언된 파라미터의 이름을 나타내고, 'PV1' 및 'PV2'는 해당 파라미터의 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다. 스택 생성이 실패하면 롤백되지 않습니다.

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

예 2: 지정된 이름으로 새 스택을 생성합니다. 템플릿은 사용자 지정 파라미터를 사용하여 제공된 콘텐츠에서 구문 분석됩니다. 여기서, 'PK1' 및 'PK2'는 템플릿 콘텐츠에 선언된 파라미터의 이름을 나타내고, 'PV1' 및 'PV2'는 해당 파라미터의 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다. 스택 생성이 실패하면 롤백됩니다.

```
$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"
```

예 3: 지정된 이름으로 새 스택을 생성합니다. 템플릿은 사용자 지정 파라미터를 사용하여 Amazon S3 URL에서 가져옵니다. 여기서, 'PK1'은 템플릿 콘텐츠에 선언된 파라미터의 이름을 나타내고, 'PV1'은 해당 파라미터의 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다. 스택 생성이 실패하면 롤백됩니다(-DisableRollback $false를 지정하는 것과 동일).

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

예 4: 지정된 이름으로 새 스택을 생성합니다. 템플릿은 사용자 지정 파라미터를 사용하여 Amazon S3 URL에서 가져옵니다. 여기서, 'PK1'은 템플릿 콘텐츠에 선언된 파라미터의 이름을 나타내고, 'PV1'은 해당 파라미터의 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다. 스택 생성이 실패하면 롤백됩니다(-DisableRollback $false를 지정하는 것과 동일). 지정된 알림 AEN은 게시된 스택 관련 이벤트를 수신합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateStack을 참조하세요.

`Remove-CFNStack`

다음 코드 예시는 `Remove-CFNStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스택을 삭제합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteStack을 참조하세요.

`Resume-CFNUpdateRollback`

다음 코드 예시는 `Resume-CFNUpdateRollback`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이름이 지정된 스택의 롤백을 계속합니다. 스택은 'UPDATE_ROLLBACK_FAILED' 상태여야 합니다. 계속된 롤백이 성공하면 스택은 'UPDATE_ROLLBACK_COMPLETE' 상태가 됩니다.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ContinueUpdateRollback을 참조하세요.

`Stop-CFNUpdateStack`

다음 코드 예시는 `Stop-CFNUpdateStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스택에 대한 업데이트를 취소합니다.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelUpdateStack을 참조하세요.

`Test-CFNStack`

다음 코드 예시는 `Test-CFNStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 스택이 UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE 또는 UPDATE_COMPLETE 상태 중 하나에 도달했는지 테스트합니다.

```
Test-CFNStack -StackName MyStack
```

출력:

```
False
```

예제 2: 스택이 UPDATE_COMPLETE 또는 UPDATE_ROLLBACK_COMPLETE 상태에 도달했는지 테스트합니다.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Test-CFNStack을 참조하세요.

`Test-CFNTemplate`

다음 코드 예시는 `Test-CFNTemplate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 템플릿 콘텐츠의 유효성을 확인합니다. 출력에는 템플릿의 기능, 설명 및 파라미터가 자세히 설명되어 있습니다.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

예 2: Amazon S3 URL을 통해 액세스한 지정된 템플릿의 유효성을 확인합니다. 출력에는 템플릿의 기능, 설명 및 파라미터가 자세히 설명되어 있습니다.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ValidateTemplate을 참조하세요.

Update-CFNStack

다음 코드 예시는 Update-CFNStack의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 'myStack' 스택을 업데이트합니다. 'PK1'은 템플릿에 선언된 파라미터의 이름을 나타내고 'PV1'은 해당 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

예 2: 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 'myStack' 스택을 업데이트합니다. 'PK1' 및 'PK2'는 템플릿에 선언된 파라미터의 이름을 나타내고, 'PV1' 및 'PV2'는 요청된 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

예 3: 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 'myStack' 스택을 업데이트합니다. 'PK1'은 템플릿에 선언된 파라미터의 이름을 나타내고 'PV2'는 해당 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

예 4: Amazon S3에서 가져온 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 'myStack' 스택을 업데이트합니다. 'PK1' 및 'PK2'는 템플릿에 선언된 파라미터의 이름을 나타내고, 'PV1' 및 'PV2'는 요청된 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다.

```
Update-CFNStack -StackName "myStack" `
```

```
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

예 5: Amazon S3에서 가져온 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 이 예제에서 IAM 리소스를 포함하는 것으로 가정되는 'myStack' 스택을 업데이트합니다. 'PK1' 및 'PK2'는 템플릿에 선언된 파라미터의 이름을 나타내고, 'PV1' 및 'PV2'는 요청된 값을 나타냅니다. 사용자 지정 파라미터는 'ParameterKey' 및 'ParameterValue' 대신 'Key' 및 'Value'를 사용하여 지정할 수도 있습니다. IAM 리소스가 포함된 스택에서는 -Capabilities "CAPABILITY_IAM" 파라미터를 지정해야 합니다. 그렇지 않으면 업데이트가 실패하고 'InsufficientCapabilities' 오류가 발생합니다.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateStack을 참조하세요.

`Wait-CFNStack`

다음 코드 예시는 `Wait-CFNStack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 스택이 UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE 또는 UPDATE_COMPLETE 상태 중 하나에 도달했는지 테스트합니다. 스택이 해당 상태 중 하나가 아닌 경우 상태를 다시 테스트하기 전에 명령이 2초 동안 대기합니다. 스택이 요청된 상태 중 하나에 도달하거나 기본 제한 시간인 60초가 경과할 때까지 이 작업이 반복됩니다. 제한 시간이 초과되면 예외가 발생합니다. 스택이 제한 시간 내에 요청된 상태 중 하나에 도달하면 파이프라인으로 반환됩니다.

```
$stack = Wait-CFNStack -StackName MyStack
```

예제 2: 이 예제에서는 스택이 지정된 상태 중 하나에 도달할 때까지 총 5분(300초) 동안 대기합니다. 여기에서는 제한 시간 전에 해당 상태에 도달하므로 스택 객체가 파이프라인으로 반환됩니다.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

출력:

```
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
```

예제 3: 이 예제에서는 스택이 제한 시간(이 경우 기본값인 60초) 내에 요청된 상태 중 하나에 도달하지 않을 때의 오류 출력을 보여줍니다.

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Wait-CFNStack을 참조하세요.

Tools for PowerShell V4를 사용한 CloudFront 예제

다음 코드 예제에서는 CloudFront에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-CFCloudFrontOriginAccessIdentity`

다음 코드 예시는 `Get-CFCloudFrontOriginAccessIdentity`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예제는 -Id 파라미터로 지정된 특정 Amazon CloudFront 원본 액세스 ID를 반환합니다. -Id 파라미터가 필수는 아니지만 이 파라미터를 지정하지 않으면 결과가 반환되지 않습니다.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

출력:

```
     CloudFrontOriginAccessIdentityConfig    Id
  S3CanonicalUserId
```

```
    ------------------------------------    --
----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCloudFrontOriginAccessIdentity를 참조하세요.

`Get-CFCloudFrontOriginAccessIdentityConfig`

다음 코드 예시는 `Get-CFCloudFrontOriginAccessIdentityConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예제는 -Id 파라미터로 지정된 단일 Amazon CloudFront 원본 액세스 ID에 대한 구성 정보를 반환합니다. -Id 파라미터가 지정되지 않은 경우 오류가 발생합니다.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

출력:

```
    CallerReference                                          Comment
    ---------------                                          -------
    mycallerreference: 2/1/2011 1:16:32 PM                   Caller reference:
2/1/2011 1:16:32 PM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCloudFrontOriginAccessIdentityConfig를 참조하세요.

`Get-CFCloudFrontOriginAccessIdentityList`

다음 코드 예시는 `Get-CFCloudFrontOriginAccessIdentityList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예제는 Amazon CloudFront 원본 액세스 ID 목록을 반환합니다. -MaxItem 파라미터는 값 2를 지정하므로 결과에는 두 개의 ID가 포함됩니다.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

출력:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListCloudFrontOriginAccessIdentities를 참조하세요.

`Get-CFDistribution`

다음 코드 예시는 `Get-CFDistribution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 특정 배포에 대한 정보를 검색합니다.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDistribution을 참조하세요.

`Get-CFDistributionConfig`

다음 코드 예시는 `Get-CFDistributionConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 특정 배포에 대한 구성을 검색합니다.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDistributionConfig를 참조하세요.

`Get-CFDistributionList`

다음 코드 예시는 `Get-CFDistributionList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 배포판을 반환합니다.

```
Get-CFDistributionList
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDistributions를 참조하세요.

`New-CFDistribution`

다음 코드 예시는 `New-CFDistribution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 로깅 및 캐싱으로 구성된 기본 CloudFront 배포를 생성합니다.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
```

```
    -Aliases_Quantity 0
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDistribution을 참조하세요.

New-CFInvalidation

다음 코드 예시는 New-CFInvalidation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 ID가 EXAMPLENSTXAXE인 배포에 대해 새 무효화를 생성합니다. CallerReference는 사용자가 선택한 고유한 ID입니다. 이 경우 2019년 5월 15일 오전 9시를 나타내는 타임스탬프가 사용됩니다. $Paths 변수는 사용자가 원하지 않는 이미지 및 미디어 파일에 대한 세 가지 경로를 배포 캐시의 일부로 저장합니다. -Paths_Quantity 매개 변수 값은 -Paths_Item 매개 변수에 지정된 총 경로 수입니다.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

출력:

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateInvalidation을 참조하세요.

New-CFSignedCookie

다음 코드 예시는 New-CFSignedCookie의 사용 방법을 보여줍니다.

예제 1: 준비된 정책을 사용하여 지정된 리소스에 대해 서명된 쿠키를 생성합니다. 쿠키는 1년 동안 유효합니다.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

출력:

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

예제 2: 사용자 지정 정책을 사용하여 지정된 리소스에 대해 서명된 쿠키를 생성합니다. 쿠키는 24시간 뒤부터 유효하며 1주일 후에 만료됩니다.

```
$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
```

출력:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

예제 3: 사용자 지정 정책을 사용하여 지정된 리소스에 대해 서명된 쿠키를 생성합니다. 쿠키는 24시간 뒤부터 유효하며 1주일 후에 만료됩니다. 리소스에 대한 액세스는 지정된 IP 범위로 제한됩니다.

```
$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
```

출력:

```
Policy
                                                                  ------

                                    [CloudFront-Policy, eyJTd...wIjo...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 New-CFSignedCookie를 참조하세요.

`New-CFSignedUrl`

다음 코드 예시는 `New-CFSignedUrl`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 준비된 정책을 사용하여 지정된 리소스에 대해 서명된 URL을 생성합니다. URL은 1시간 동안 유효합니다. 서명된 URL이 포함된 System.Uri 객체가 파이프라인으로 내보내집니다.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
```

```
}
New-CFSignedUrl @params
```

예제 2: 사용자 지정 정책을 사용하여 지정된 리소스에 대해 서명된 URL을 생성합니다. URL은 24시간 뒤부터 유효하며 1주일 후에 만료됩니다.

```
$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
```

예제 3: 사용자 지정 정책을 사용하여 지정된 리소스에 대해 서명된 URL을 생성합니다. URL은 24시간 뒤부터 유효하며 1주일 후에 만료됩니다. 리소스에 대한 액세스는 지정된 IP 범위로 제한됩니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 New-CFSignedUrl을 참조하세요.

Tools for PowerShell V4를 사용한 CloudTrail 예제

다음 코드 예제에서는 CloudTrail과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Find-CTEvent`

다음 코드 예시는 `Find-CTEvent`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지난 7일 동안 발생한 모든 이벤트를 반환합니다. 기본적으로 cmdlet은 모든 이벤트를 전달하기 위해 자동으로 여러 번의 직접 호출을 사용하며, 서비스에서 추가 데이터를 사용할 수 없다고 표시되면 종료됩니다.

```
Find-CTEvent
```

예제 2: 현재 쉘 기본값이 아닌 리전을 지정하여 지난 7일 동안 발생한 모든 이벤트를 반환합니다.

```
Find-CTEvent -Region eu-central-1
```

예제 3: RunInstances API 직접 호출과 연결된 모든 이벤트를 반환합니다.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

예제 4: 처음 5개의 사용 가능한 이벤트를 반환합니다.

```
Find-CTEvent -MaxResult 5
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 LookupEvents를 참조하세요.

Get-CTTrail

다음 코드 예시는 Get-CTTrail의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 계정의 현재 리전과 연결된 모든 추적의 설정을 반환합니다.

```
Get-CTTrail
```

예제 2: 지정된 추적에 대한 설정을 반환합니다.

```
Get-CTTrail -TrailNameList trail1,trail2
```

예제 3: 현재 쉘 기본값이 아닌 리전(이 경우 프랑크푸르트(eu-central-1) 리전)에서 생성된 지정된 추적에 대한 설정을 반환합니다.

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrails을 참조하세요.

Get-CTTrailStatus

다음 코드 예시는 Get-CTTrailStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 'myExampleTrail'인 추적의 상태 정보를 반환합니다. 반환된 데이터에는 전송 오류, Amazon SNS 및 Amazon S3 오류, 추적의 로깅 시작 및 중지 시간에 대한 정보가 포함됩니다. 이 예제에서는 추적이 현재 쉘 기본값과 동일한 리전에서 생성되었다고 가정합니다.

```
Get-CTTrailStatus -Name myExampleTrail
```

예제 2: 현재 쉘 기본값이 아닌 리전(이 경우 프랑크푸르트(eu-central-1) 리전)에서 생성된 추적의 상태 정보를 반환합니다.

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetTrailStatus를 참조하세요.

`New-CTTrail`

다음 코드 예시는 `New-CTTrail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 로그 파일 스토리지에 'amzn-s3-demo-bucket' 버킷을 사용할 추적을 생성합니다.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

예제 2: 로그 파일 스토리지에 'amzn-s3-demo-bucket' 버킷을 사용할 추적을 생성합니다. 로그를 나타내는 S3 객체의 공통 키 접두사는 'mylogs'입니다. 새 로그가 버킷에 전달되면 알림이 SNS 주제 'mlog-deliverytopic'으로 전송됩니다. 이 예제에서는 스플래팅을 사용하여 cmdlet에 파라미터 값을 제공합니다.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTrail을 참조하세요.

`Remove-CTTrail`

다음 코드 예시는 `Remove-CTTrail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 추적을 삭제합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치 파라미터를 추가합니다.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTrail을 참조하세요.

`Start-CTLogging`

다음 코드 예시는 `Start-CTLogging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 'myExampleTrail'이라는 추적에 대한 AWS API 호출 및 로그 파일 전송 기록을 시작합니다. 이 예제에서는 추적이 현재 쉘 기본값과 동일한 리전에서 생성되었다고 가정합니다.

```
Start-CTLogging -Name myExampleTrail
```

예제 2: 현재 쉘 기본값이 아닌 리전(이 경우 프랑크푸르트(eu-central-1) 리전)에서 생성된 추적에 대한 AWS API 호출 및 로그 파일 전송의 기록을 시작합니다.

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartLogging을 참조하세요.

`Stop-CTLogging`

다음 코드 예시는 `Stop-CTLogging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 'myExampleTrail'이라는 추적에 대한 AWS API 호출 녹음 및 로그 파일 전송을 일시 중지합니다. 이 예제에서는 추적이 현재 쉘 기본값과 동일한 리전에서 생성되었다고 가정합니다.

```
Stop-CTLogging -Name myExampleTrail
```

예제 2: 현재 쉘 기본값(이 경우 프랑크푸르트(eu-central-1) 리전)이 아닌 다른 리전에서 생성된 추적에 대한 AWS API 호출 및 로그 파일 전송 기록을 일시 중지합니다.

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopLogging을 참조하세요.

`Update-CTTrail`

다음 코드 예시는 `Update-CTTrail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 글로벌 서비스 이벤트(예: IAM의 이벤트)가 기록되도록 지정된 추적을 업데이트하고 앞으로 생성되는 로그 파일의 공통 키 접두사를 'globallogs'로 변경합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

예제 2: 새 로그 전송에 대한 알림이 지정된 SNS 주제로 전송되도록 지정된 추적을 업데이트합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

예제 3: 로그가 다른 버킷으로 전송되도록 지정된 추적을 업데이트합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateTrail을 참조하세요.

Tools for PowerShell V4를 사용한 CloudWatch 예제

다음 코드 예제에서는 CloudWatch와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-CWAlarm`

다음 코드 예시는 `Get-CWAlarm`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: CloudWatch에서 복합 및 지표 경보를 포함한 모든 경보를 반환합니다.

```
Get-CWAlarm -MaxRecords 1
```

출력:

```
CompositeAlarms MetricAlarms         NextToken
--------------- ------------         ---------
                {MetricAlarms-01}    NextToken-01
                {MetricAlarms-02}    NextToken-02
                {MetricAlarms-03}    NextToken-03
```

예제 2: -AlarmType 파라미터를 CompositeAlarms로 설정한 후 CloudWatch에서 복합 경보 데이터만 반환합니다.

```
Get-CWAlarm -AlarmType 'CompositeAlarms'
```

출력:

```
CompositeAlarms        MetricAlarms NextToken
---------------        ------------ ---------
{CompositeAlarms-01}
{CompositeAlarms-02}
{CompositeAlarms-03}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAlarms를 참조하세요.

`Get-CWDashboard`

다음 코드 예시는 `Get-CWDashboard`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 지정된 대시보드의 arn 본문을 반환합니다.

```
Get-CWDashboard -DashboardName Dashboard1
```

출력:

```
DashboardArn                                                DashboardBody
------------                                                -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDashboard를 참조하세요.

`Get-CWDashboardList`

다음 코드 예시는 `Get-CWDashboardList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 계정의 대시보드 컬렉션을 반환합니다.

```
Get-CWDashboardList
```

출력:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

예시 2: 이름이 접두사 'dev'로 시작하는 계정의 대시보드 컬렉션을 반환합니다.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDashboards를 참조하세요.

`Remove-CWDashboard`

다음 코드 예시는 `Remove-CWDashboard`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 지정된 대시보드를 삭제하고 계속하기 전에 확인을 위해 승격합니다. 확인을 우회하려면 명령에 -Force 스위치를 추가합니다.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDashboards를 참조하세요.

`Write-CWDashboard`

다음 코드 예시는 `Write-CWDashboard`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 지표 위젯 2개를 나란히 포함하도록 'Dashboard1'이라는 대시보드를 생성하거나 업데이트합니다.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
```

```
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
                      "BucketName",
                      "amzn-s3-demo-bucket"
                   ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

예시 2: 대시보드를 생성하거나 업데이트하여 대시보드를 설명하는 콘텐츠를 cmdlet에 전달합니다.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutDashboard를 참조하세요.

`Write-CWMetricData`

다음 코드 예시는 `Write-CWMetricData`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 새 MetricDatum 객체를 생성하고, Amazon Web Services CloudWatch 지표에 씁니다.

```
### 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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutMetricData를 참조하세요.

Tools for PowerShell V4를 사용한 CodeCommit 예제

다음 코드 예제에서는 CodeCommit과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-CCBranch`

다음 코드 예시는 `Get-CCBranch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리의 지정된 브랜치에 대한 정보를 가져옵니다.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

출력:

```
BranchName                              CommitId
```

```
----------                              --------
MyNewBranch                             7763222d...561fc9c9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBranch를 참조하세요.

Get-CCBranchList

다음 코드 예시는 Get-CCBranchList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리의 브랜치 이름 목록을 가져옵니다.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

출력:

```
master
MyNewBranch
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListBranches를 참조하세요.

Get-CCRepository

다음 코드 예시는 Get-CCRepository의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리에 대한 정보를 가져옵니다.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetRepository를 참조하세요.

Get-CCRepositoryBatch

다음 코드 예시는 Get-CCRepositoryBatch의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리 중 어떤 리포지토리를 찾을 수 있고 찾을 수 없는지 확인합니다.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

출력:

```
Repositories                          RepositoriesNotFound
------------                          --------------------
{MyDemoRepo, MyNewRepo}               {AMissingRepo}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetRepositories를 참조하세요.

Get-CCRepositoryList

다음 코드 예시는 Get-CCRepositoryList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 리포지토리를 이름을 기준으로 오름차순으로 나열합니다.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

출력:

```
RepositoryId                             RepositoryName
------------                             --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE     MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE     MyNewRepo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListRepositories를 참조하세요.

New-CCBranch

다음 코드 예시는 New-CCBranch의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리의 지정된 이름과 지정된 커밋 ID로 새 브랜치를 생성합니다.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateBranch를 참조하세요.

New-CCRepository

다음 코드 예시는 New-CCRepository의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름과 지정된 설명으로 새 리포지토리를 생성합니다.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRepository를 참조하세요.

`Remove-CCRepository`

다음 코드 예시는 `Remove-CCRepository`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리를 강제로 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 리포지토리를 삭제합니다.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

출력:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRepository를 참조하세요.

`Update-CCDefaultBranch`

다음 코드 예시는 `Update-CCDefaultBranch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리의 기본 브랜치를 지정된 브랜치로 변경합니다.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateDefaultBranch를 참조하세요.

`Update-CCRepositoryDescription`

다음 코드 예시는 `Update-CCRepositoryDescription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리에 대한 설명을 변경합니다.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateRepositoryDescription을 참조하세요.

`Update-CCRepositoryName`

다음 코드 예시는 `Update-CCRepositoryName`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리포지토리의 이름을 변경합니다.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateRepositoryName을 참조하세요.

Tools for PowerShell V4를 사용한 CodeDeploy 예제

다음 코드 예제에서는 CodeDeploy와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

Add-CDOnPremiseInstanceTag

다음 코드 예시는 Add-CDOnPremiseInstanceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 온프레미스 인스턴스에 대해 지정된 키와 값이 있는 온프레미스 인스턴스 태그를 추가합니다.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddTagsToOnPremisesInstances를 참조하세요.

Get-CDApplication

다음 코드 예시는 Get-CDApplication의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대한 정보를 가져옵니다.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

출력:

```
ApplicationId                           ApplicationName             CreateTime
         LinkedToGitHub
-------------                           ---------------             ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication   7/20/2015
 9:49:48 PM    False
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetApplication을 참조하세요.

Get-CDApplicationBatch

다음 코드 예시는 Get-CDApplicationBatch의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대한 정보를 가져옵니다.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetApplications을 참조하세요.

Get-CDApplicationList

다음 코드 예시는 Get-CDApplicationList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 애플리케이션 목록을 가져옵니다.

```
Get-CDApplicationList
```

출력:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListApplications을 참조하세요.

`Get-CDApplicationRevision`

다음 코드 예시는 `Get-CDApplicationRevision`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션 개정에 대한 정보를 가져옵니다.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

출력:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetApplicationRevision을 참조하세요.

`Get-CDApplicationRevisionList`

다음 코드 예시는 `Get-CDApplicationRevisionList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 사용 가능한 개정에 대한 정보를 가져옵니다.

```
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)
>>   }
>> }
>>
```

출력:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListApplicationRevisions을 참조하세요.

`Get-CDDeployment`

다음 코드 예시는 `Get-CDDeployment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포에 대한 요약 정보를 가져옵니다.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 배포에 참여하는 인스턴스의 상태에 대한 정보를 가져옵니다.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

출력:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

예제 3: 이 예제에서는 지정된 배포의 애플리케이션 개정에 대한 정보를 가져옵니다.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

출력:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDeployment를 참조하세요.

`Get-CDDeploymentBatch`

다음 코드 예시는 `Get-CDDeploymentBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포에 대한 정보를 가져옵니다.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetDeployments를 참조하세요.

`Get-CDDeploymentConfig`

다음 코드 예시는 `Get-CDDeploymentConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포 구성에 대한 요약 정보를 가져옵니다.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

출력:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

예제 2: 이 예제에서는 지정된 배포 구성의 정의에 대한 정보를 가져옵니다.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

출력:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDeploymentConfig를 참조하세요.

`Get-CDDeploymentConfigList`

다음 코드 예시는 `Get-CDDeploymentConfigList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 배포 구성 목록을 가져옵니다.

```
Get-CDDeploymentConfigList
```

출력:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDeploymentConfigs를 참조하세요.

Get-CDDeploymentGroup

다음 코드 예시는 Get-CDDeploymentGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포 그룹에 대한 정보를 가져옵니다.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDeploymentGroup을 참조하세요.

Get-CDDeploymentGroupList

다음 코드 예시는 Get-CDDeploymentGroupList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션의 배포 그룹 목록을 가져옵니다.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

출력:

```
ApplicationName              DeployментGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDeploymentGroups을 참조하세요.

`Get-CDDeploymentInstance`

다음 코드 예시는 `Get-CDDeploymentInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포의 지정된 인스턴스에 대한 정보를 가져옵니다.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDeploymentInstance를 참조하세요.

`Get-CDDeploymentInstanceList`

다음 코드 예시는 `Get-CDDeploymentInstanceList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포의 인스턴스 ID 목록을 가져옵니다.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

출력:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDeploymentInstances를 참조하세요.

Get-CDDeploymentList

다음 코드 예시는 Get-CDDeploymentList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션 및 배포 그룹의 배포 ID 목록을 가져옵니다.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

출력:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDeployments를 참조하세요.

Get-CDOnPremiseInstance

다음 코드 예시는 Get-CDOnPremiseInstance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 온프레미스 인스턴스에 대한 정보를 가져옵니다.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetOnPremisesInstance를 참조하세요.

`Get-CDOnPremiseInstanceBatch`

다음 코드 예시는 `Get-CDOnPremiseInstanceBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 온프레미스 인스턴스에 대한 정보를 가져옵니다.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetOnPremisesInstances를 참조하세요.

Get-CDOnPremiseInstanceList

다음 코드 예시는 Get-CDOnPremiseInstanceList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 온프레미스 인스턴스 이름 목록을 가져옵니다.

```
Get-CDOnPremiseInstanceList
```

출력:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListOnPremisesInstances를 참조하세요.

New-CDApplication

다음 코드 예시는 New-CDApplication의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름으로 새로운 애플리케이션을 생성합니다.

```
New-CDApplication -ApplicationName MyNewApplication
```

출력:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateApplication을 참조하세요.

New-CDDeployment

다음 코드 예시는 New-CDDeployment의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포 구성 및 애플리케이션 개정을 사용하여 지정된 애플리케이션 및 배포 그룹에 대한 새 배포를 생성합니다.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

출력:

```
d-ZHROG7UEX
```

예제 2: 이 예제에서는 블루/그린 배포를 위한 대체 환경에 포함시킬 인스턴스를 식별해야 하는 EC2 인스턴스 태그 그룹을 지정하는 방법을 보여줍니다.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
 -S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

출력:

```
d-ZHROG7UEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDeployment를 참조하세요.

`New-CDDeploymentConfig`

다음 코드 예시는 `New-CDDeploymentConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름 및 동작으로 새 배포 구성을 생성합니다.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

출력:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDeploymentConfig를 참조하세요.

New-CDDeploymentGroup

다음 코드 예시는 New-CDDeploymentGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대해 지정된 이름, Auto Scaling 그룹, 배포 구성, 태그, 서비스 역할을 사용하여 배포 그룹을 생성합니다.

```
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
```

출력:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

예제 2: 이 예제에서는 블루/그린 배포를 위한 대체 환경에 포함시킬 인스턴스를 식별해야 하는 EC2 인스턴스 태그 그룹을 지정하는 방법을 보여줍니다.

```
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=
```

출력:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDeploymentGroup을 참조하세요.

`Register-CDApplicationRevision`

다음 코드 예시는 `Register-CDApplicationRevision`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대해 지정된 Amazon S3 위치에 애플리케이션 개정을 등록합니다.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterApplicationRevision을 참조하세요.

`Register-CDOnPremiseInstance`

다음 코드 예시는 `Register-CDOnPremiseInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름과 IAM 사용자를 사용하여 온프레미스 인스턴스를 등록합니다.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterOnPremisesInstance를 참조하세요.

`Remove-CDApplication`

다음 코드 예시는 `Remove-CDApplication`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름의 애플리케이션을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 애플리케이션을 삭제합니다.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteApplication을 참조하세요.

`Remove-CDDeploymentConfig`

다음 코드 예시는 `Remove-CDDeploymentConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름의 배포 구성을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 배포 구성을 삭제합니다.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDeploymentConfig를 참조하세요.

`Remove-CDDeploymentGroup`

다음 코드 예시는 `Remove-CDDeploymentGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대해 지정된 이름의 배포 그룹을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 배포 그룹을 삭제합니다.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDeploymentGroup을 참조하세요.

Remove-CDOnPremiseInstanceTag

다음 코드 예시는 Remove-CDOnPremiseInstanceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름의 온프레미스 인스턴스에 대해 지정된 태그를 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 태그를 삭제합니다.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveTagsFromOnPremisesInstances를 참조하세요.

Stop-CDDeployment

다음 코드 예시는 Stop-CDDeployment의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배포 ID로 배포 중지를 시도합니다.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

출력:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopDeployment를 참조하세요.

Unregister-CDOnPremiseInstance

다음 코드 예시는 Unregister-CDOnPremiseInstance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름의 온프레미스 인스턴스 등록을 취소합니다.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterOnPremisesInstance를 참조하세요.

Update-CDApplication

다음 코드 예시는 Update-CDApplication의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션의 이름을 변경합니다.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateApplication을 참조하세요.

Update-CDDeploymentGroup

다음 코드 예시는 Update-CDDeploymentGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션에 대해 지정된 배포 그룹의 이름을 변경합니다.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

예제 2: 이 예제에서는 블루/그린 배포를 위한 대체 환경에 포함시킬 인스턴스를 식별해야 하는 EC2 인스턴스 태그 그룹을 지정하는 방법을 보여줍니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateDeploymentGroup을 참조하세요.

Tools for PowerShell V4를 사용한 CodePipeline 예제

다음 코드 예제에서는 CodePipeline과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Confirm-CPJob`

다음 코드 예시는 `Confirm-CPJob`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 작업의 상태를 가져옵니다.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

출력:

```
Value
-----
InProgress
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AcknowledgeJob을 참조하세요.

`Disable-CPStageTransition`

다음 코드 예시는 `Disable-CPStageTransition`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인의 지정된 단계에 대한 인바운드 전환을 비활성화합니다.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableStageTransition을 참조하세요.

`Enable-CPStageTransition`

다음 코드 예시는 `Enable-CPStageTransition`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인의 지정된 단계에 대한 인바운드 전환을 활성화합니다.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableStageTransition을 참조하세요.

`Get-CPActionType`

다음 코드 예시는 `Get-CPActionType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 소유자에 대해 사용 가능한 모든 작업의 정보를 가져옵니다.

```
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)
}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListActionTypes을 참조하세요.

`Get-CPActionableJobList`

다음 코드 예시는 `Get-CPActionableJobList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 작업 범주, 소유자, 공급자, 버전, 쿼리 파라미터에 대해 실행 가능한 모든 작업의 정보를 가져옵니다.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

출력:

```
AccountId       Data                                    Id
         Nonce
---------       ----                                    --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData    0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PollForJobs을 참조하세요.

`Get-CPJobDetail`

다음 코드 예시는 `Get-CPJobDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 작업에 대한 일반 정보를 가져옵니다.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

출력:

```
AccountId       Data                                    Id
---------       ----                                    --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

예제 2: 이 예제에서는 지정된 작업에 대한 자세한 정보를 가져옵니다.

```
$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)
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetJobDetails을 참조하세요.

Get-CPPipeline

다음 코드 예시는 Get-CPPipeline의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인에 대한 일반 정보를 가져옵니다.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

출력:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

예제 2: 이 예제에서는 지정된 파이프라인에 대한 자세한 정보를 가져옵니다.

```
$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)
  }
}
```

출력:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
  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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPipeline을 참조하세요.

Get-CPPipelineList

다음 코드 예시는 Get-CPPipelineList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 파이프라인 목록을 가져옵니다.

```
Get-CPPipelineList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListPipelines을 참조하세요.

Get-CPPipelineState

다음 코드 예시는 Get-CPPipelineState의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인의 단계에 대한 일반 정보를 가져옵니다.

```
Get-CPPipelineState -Name CodePipelineDemo
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 파이프라인의 상태에 대한 자세한 정보를 가져옵니다.

```
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)
  }
}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPipelineState를 참조하세요.

New-CPCustomActionType

다음 코드 예시는 New-CPCustomActionType의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 속성을 사용하여 새로운 사용자 지정 작업을 생성합니다.

```
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
```

출력:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCustomActionType을 참조하세요.

New-CPPipeline

다음 코드 예시는 New-CPPipeline의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 설정으로 새로운 파이프라인을 생성합니다.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"
```

```
$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

출력:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreatePipeline을 참조하세요.

`Remove-CPCustomActionType`

다음 코드 예시는 `Remove-CPCustomActionType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 사용자 지정 작업을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 사용자 지정 작업을 삭제합니다.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCustomActionType을 참조하세요.

`Remove-CPPipeline`

다음 코드 예시는 `Remove-CPPipeline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. -Force 파라미터를 추가하여 프롬프트 없이 파이프라인을 삭제합니다.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePipeline을 참조하세요.

`Start-CPPipelineExecution`

다음 코드 예시는 `Start-CPPipelineExecution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파이프라인의 실행을 시작합니다.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartPipelineExecution을 참조하세요.

`Update-CPPipeline`

다음 코드 예시는 `Update-CPPipeline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 기존 파이프라인을 지정된 설정으로 업데이트합니다.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"
```

```
$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

출력:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdatePipeline을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon Cognito ID 예제

다음 코드 예제에서는 Amazon Cognito Identity와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-CGIIdentityPool`

다음 코드 예시는 `Get-CGIIdentityPool`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID별로 특정 ID 풀에 대한 정보를 검색합니다.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeIdentityPool을 참조하세요.

Get-CGIIdentityPoolList

다음 코드 예시는 Get-CGIIdentityPoolList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 기존 ID 풀 목록을 검색합니다.

```
Get-CGIIdentityPoolList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListIdentityPools을 참조하세요.

Get-CGIIdentityPoolRole

다음 코드 예시는 Get-CGIIdentityPoolRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 특정 ID 풀의 역할에 대한 정보를 가져옵니다.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetIdentityPoolRoles을 참조하세요.

New-CGIIdentityPool

다음 코드 예시는 New-CGIIdentityPool의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 인증되지 않은 ID를 허용하는 새 ID 풀을 생성합니다.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateIdentityPool을 참조하세요.

Remove-CGIIdentityPool

다음 코드 예시는 Remove-CGIIdentityPool의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 특정 ID 풀을 삭제합니다.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteIdentityPool을 참조하세요.

`Set-CGIIdentityPoolRole`

다음 코드 예시는 `Set-CGIIdentityPoolRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 인증되지 않은 IAM 역할을 갖도록 특정 ID 풀을 구성합니다.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetIdentityPoolRoles을 참조하세요.

`Update-CGIIdentityPool`

다음 코드 예시는 `Update-CGIIdentityPool`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID 풀 속성 중 일부를 업데이트합니다. 이 케이스에서는 ID 풀의 이름을 업데이트합니다.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateIdentityPool을 참조하세요.

AWS Config Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Config.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-CFGResourceTag`

다음 코드 예시는 `Add-CFGResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 태그를 리소스 ARN, 이 경우 config-rule/config-rule-16iyn0에 연결합니다.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResource를 참조하세요.

`Get-CFGAggregateComplianceByConfigRuleList`

다음 코드 예시는 `Get-CFGAggregateComplianceByConfigRuleList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 구성 규칙에 대한 ConfigurationAggregator 'kaju' 필터링에서 세부 정보를 가져오고 규칙의 '규정 준수'를 확장/반환합니다.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

출력:

```
ComplianceContributorCount                              ComplianceType
--------------------------                              --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

예제 2: 이 예제에서는 지정된 ConfigurationAggregator에서 세부 정보를 가져오고, 이 세부 정보를 지정된 계정과 애그리게이터에 포함된 모든 리전을 기준으로 필터링하고, 모든 규칙에 대한 규정 준수 여부를 추가로 반환합니다.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

출력:

```
ConfigRuleName                      Compliance
--------------                      ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip           NON_COMPLIANT
desired-instance-type               NON_COMPLIANT
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAggregateComplianceByConfigRules을 참조하세요.

Get-CFGAggregateComplianceDetailsByConfigRule

다음 코드 예시는 Get-CFGAggregateComplianceDetailsByConfigRule의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 지정된 계정, 집계자, 리전 및 구성 규칙에 대해 'COMPLIANT' 상태인 AWS Config 규칙 'desired-instance-type'에 대한 resource-id 및 resource-type이 있는 출력을 선택하는 평가 결과를 반환합니다.

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

출력:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAggregateComplianceDetailsByConfigRule을 참조하세요.

Get-CFGAggregateConfigRuleComplianceSummary

다음 코드 예시는 Get-CFGAggregateConfigRuleComplianceSummary의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애그리게이터에 대한 규정 미준수 규칙 수를 반환합니다.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

출력:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAggregateConfigRuleComplianceSummary를 참조하세요.

`Get-CFGAggregateDiscoveredResourceCount`

다음 코드 예시는 `Get-CFGAggregateDiscoveredResourceCount`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 us-east-1 리전에 대해 필터링된 지정된 애그리게이터의 리소스 수를 반환합니다.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

출력:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- -----------------------
           {}                              455
```

예제 2: 이 예제에서는 지정된 애그리게이터의 필터링된 리전에 대해 RESOURCE_TYPE별로 그룹화된 리소스 수를 반환합니다.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAggregateDiscoveredResourceCounts를 참조하세요.

`Get-CFGAggregateDiscoveredResourceList`

다음 코드 예시는 `Get-CFGAggregateDiscoveredResourceList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 '아일랜드' 애그리게이터에 집계된 지정된 리소스 유형에 대한 리소스 식별자를 반환합니다. 리소스 유형 목록은 https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/TConfigServiceResourceType.html&tocid=Amazon_ConfigService_ResourceType 페이지를 확인하세요.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

출력:

```
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
```

예제 2: 이 예제에서는 us-east-1 리전으로 필터링된 지정된 애그리게이터에 대해 'default'라는 **AwsEC2SecurityGroup** 리소스 유형을 반환합니다.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAggregateDiscoveredResources를 참조하세요.

Get-CFGAggregateResourceConfig

다음 코드 예시는 Get-CFGAggregateResourceConfig의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 집계된 지정 리소스의 구성 항목을 반환하고 구성을 확장합니다.

```
(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
```

출력:

```
{"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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAggregateResourceconfig-service를 참조하세요.

`Get-CFGAggregateResourceConfigBatch`

다음 코드 예시는 `Get-CFGAggregateResourceConfigBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애그리게이터에 있는 (식별된) 리소스의 현재 구성 항목을 가져옵니다.

```
$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
```

출력:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- -----------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetAggregateResourceconfig-service를 참조하세요.

`Get-CFGAggregationAuthorizationList`

다음 코드 예시는 `Get-CFGAggregationAuthorizationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 애그리게이터에 부여된 권한을 검색합니다.

```
Get-CFGAggregationAuthorizationList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAggregationAuthorizations을 참조하세요.

`Get-CFGComplianceByConfigRule`

다음 코드 예시는 `Get-CFGComplianceByConfigRule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 규칙에 대한 현재 평가 결과가 없으므로 INSUFFICIENT_DATA를 반환하는 규칙 ebs-optimized-instance에 대한 규정 준수 세부 정보를 검색합니다.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

출력:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

예제 2: 이 예제는 규칙 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK에 대한 비준수 리소스 수를 반환합니다.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

출력:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeComplianceByConfigRule을 참조하세요.

`Get-CFGComplianceByResource`

다음 코드 예시는 `Get-CFGComplianceByResource`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 **`AWS::SSM::ManagedInstanceInventory`** 리소스 유형에 'COMPLIANT' 규정 준수 유형이 있는지 확인합니다.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

출력:

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeComplianceByResource를 참조하세요.

`Get-CFGComplianceDetailsByConfigRule`

다음 코드 예시는 `Get-CFGComplianceDetailsByConfigRule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 규칙 access-keys-rotated에 대한 평가 결과를 얻고 규정 준수 유형별로 그룹화된 출력을 반환합니다.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

출력:

```
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...
```

예제 2: 이 예제는 COMPLIANT 리소스에 대한 규칙 access-keys-rotated에 대한 규정 준수 세부 정보를 쿼리합니다.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

출력:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetComplianceDetailsByConfigRule을 참조하세요.

`Get-CFGComplianceDetailsByResource`

다음 코드 예시는 `Get-CFGComplianceDetailsByResource`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 제공된 리소스에 대한 예제 평가 결과입니다.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetComplianceDetailsByResource를 참조하세요.

`Get-CFGComplianceSummaryByConfigRule`

다음 코드 예시는 `Get-CFGComplianceSummaryByConfigRule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 규정을 준수하지 않는 Config 규칙 수를 반환합니다.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

출력:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetComplianceSummaryByConfigRule을 참조하세요.

Get-CFGComplianceSummaryByResourceType

다음 코드 예시는 Get-CFGComplianceSummaryByResourceType의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 규정 준수 또는 규정 미준수 리소스 수를 반환하고 출력을 json으로 변환합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetComplianceSummaryByResourceType을 참조하세요.

Get-CFGConfigRule

다음 코드 예시는 Get-CFGConfigRule의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에는 선택한 속성과 함께 계정에 대한 구성 규칙이 나열되어 있습니다.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigRules을 참조하세요.

`Get-CFGConfigRuleEvaluationStatus`

다음 코드 예시는 `Get-CFGConfigRuleEvaluationStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 제공된 구성 규칙에 대한 상태 정보를 반환합니다.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigRuleEvaluationStatus를 참조하세요.

`Get-CFGConfigurationAggregatorList`

다음 코드 예시는 `Get-CFGConfigurationAggregatorList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 리전/계정에 대한 모든 애그리게이터를 반환합니다.

```
Get-CFGConfigurationAggregatorList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigurationAggregators를 참조하세요.

`Get-CFGConfigurationAggregatorSourcesStatus`

다음 코드 예시는 `Get-CFGConfigurationAggregatorSourcesStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 애그리게이터의 소스에 대해 요청된 필드를 표시합니다.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigurationAggregatorSourcesStatus를 참조하세요.

Get-CFGConfigurationRecorder

다음 코드 예시는 Get-CFGConfigurationRecorder의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 구성 레코더의 세부 정보를 반환합니다.

```
Get-CFGConfigurationRecorder | Format-List
```

출력:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigurationRecorders를 참조하세요.

Get-CFGConfigurationRecorderStatus

다음 코드 예시는 Get-CFGConfigurationRecorderStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 구성 레코더의 상태를 반환합니다.

```
Get-CFGConfigurationRecorderStatus
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConfigurationRecorderStatus를 참조하세요.

Get-CFGConformancePack

다음 코드 예시는 Get-CFGConformancePack의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 모든 적합성 팩을 나열합니다.

```
Get-CFGConformancePack
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConformancePacks을 참조하세요.

Get-CFGDeliveryChannel

다음 코드 예시는 Get-CFGDeliveryChannel의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 리전의 전송 채널을 검색하고 세부 정보를 표시합니다.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

출력:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDeliveryChannels을 참조하세요.

`Get-CFGResourceTag`

다음 코드 예시는 `Get-CFGResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스에 연결된 태그를 나열합니다.

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

출력:

```
Key     Value
---     -----
Version 1.3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

`Remove-CFGConformancePack`

다음 코드 예시는 `Remove-CFGConformancePack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 팩에 대한 모든 규칙, 문제 해결 작업, 평가 결과와 함께 지정된 적합성 팩을 제거합니다.

```
Remove-CFGConformancePack -ConformancePackName dono
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteConformancePack을 참조하세요.

`Write-CFGConformancePack`

다음 코드 예시는 `Write-CFGConformancePack`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 적합성 팩을 만들어 지정된 yaml 파일에서 템플릿을 가져옵니다.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutConformancePack을 참조하세요.

`Write-CFGDeliveryChannel`

다음 코드 예시는 `Write-CFGDeliveryChannel`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 기존 전송 채널의 deliveryFrequency 속성을 변경합니다.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutDeliveryChannel을 참조하세요.

Tools for PowerShell V4를 사용한 Device Farm 예제

다음 코드 예제에서는 Device Farm과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`New-DFUpload`

다음 코드 예시는 `New-DFUpload`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 Android 앱에 대한 AWS Device Farm 업로드를 생성합니다. New-DFProject 또는 Get-DFProjectList의 출력에서 프로젝트 ARN을 가져올 수 있습니다. New-DFUpload 출력에 포함된 서명된 URL을 사용하여 파일을 Device Farm에 업로드합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateUpload를 참조하세요.

Directory Service Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는 AWS Tools for PowerShell V4를와 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 Directory Service.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-DSIpRoute`

다음 코드 예시는 `Add-DSIpRoute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 Directory-id에 할당된 리소스 태그를 제거합니다.

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddIpRoutes를 참조하세요.

`Add-DSResourceTag`

다음 코드 예시는 `Add-DSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 Directory-id에 리소스 태그를 추가합니다.

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddTagsToResource를 참조하세요.

Approve-DSTrust

다음 코드 예시는 Approve-DSTrust의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 Trustid에 대한 AWS Directory Service VerifyTrust API 작업을 호출합니다.

```
Approve-DSTrust -TrustId t-9067157123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 VerifyTrust를 참조하세요.

Confirm-DSSharedDirectory

다음 코드 예시는 Confirm-DSSharedDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 디렉터리 소유자로부터 전송된 디렉터리 공유 요청을 수락합니다 AWS 계정.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AcceptSharedDirectory를 참조하세요.

Connect-DSDirectory

다음 코드 예시는 Connect-DSDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 AD Connector를 생성하여 온프레미스 디렉터리에 연결합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ConnectDirectory를 참조하세요.

`Deny-DSSharedDirectory`

다음 코드 예시는 `Deny-DSSharedDirectory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리 소유자 계정에서 보낸 디렉터리 공유 요청을 거부합니다.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

출력:

```
d-9067012345
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RejectSharedDirectory를 참조하세요.

`Disable-DSDirectoryShare`

다음 코드 예시는 `Disable-DSDirectoryShare`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리 소유자와 소비자 간의 디렉터리 공유를 중지합니다.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

출력:

```
d-9067012345
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UnshareDirectory를 참조하세요.

`Disable-DSLDAPS`

다음 코드 예시는 `Disable-DSLDAPS`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 디렉터리에 대한 LDAP 보안 직접 호출을 비활성화합니다.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableLDAPS를 참조하세요.

`Disable-DSRadius`

다음 코드 예시는 `Disable-DSRadius`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 AD Connector 또는 Microsoft AD 디렉터리에 대해 구성된 RADIUS 서버를 비활성화합니다.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableRadius를 참조하세요.

`Disable-DSSso`

다음 코드 예시는 `Disable-DSSso`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리에 대한 Single Sign-On을 비활성화합니다.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableSso를 참조하세요.

`Enable-DSDirectoryShare`

다음 코드 예시는 `Enable-DSDirectoryShare`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 핸드셰이크 메서드를 사용하여 AWS 계정의 지정된 디렉터리를 다른 AWS 계정과 공유합니다.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

출력:

```
d-9067012345
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ShareDirectory를 참조하세요.

`Enable-DSLDAPS`

다음 코드 예시는 `Enable-DSLDAPS`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 특정 디렉터리의 스위치를 활성화하여 LDAP 보안 직접 호출을 항상 사용합니다.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableLDAPS를 참조하세요.

`Enable-DSRadius`

다음 코드 예시는 `Enable-DSRadius`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 AD Connector 또는 Microsoft AD 디렉터리에 대해 제공된 RADIUS 서버 구성을 사용하여 다중 인증(MFA)을 활성화합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableRadius를 참조하세요.

`Enable-DSSso`

다음 코드 예시는 `Enable-DSSso`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리에 대한 Single Sign-On을 활성화합니다.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableSso를 참조하세요.

`Get-DSCertificate`

다음 코드 예시는 `Get-DSCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 보안 LDAP 연결을 위해 등록된 인증서에 대한 정보를 표시합니다.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCertificate를 참조하세요.

`Get-DSCertificateList`

다음 코드 예시는 `Get-DSCertificateList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 디렉터리에 대한 보안 LDAP 연결을 위해 등록된 모든 인증서를 나열합니다.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

출력:

```
CertificateId CommonName                 ExpiryDateTime       State
------------- ----------                 --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListCertificates를 참조하세요.

`Get-DSConditionalForwarder`

다음 코드 예시는 `Get-DSConditionalForwarder`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 Directory-id에 대해 구성된 모든 조건부 전달자를 가져옵니다.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

출력:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeConditionalForwarders를 참조하세요.

`Get-DSDirectory`

다음 코드 예시는 `Get-DSDirectory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 이 계정에 속한 디렉터리의 정보를 가져옵니다.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDirectories를 참조하세요.

`Get-DSDirectoryLimit`

다음 코드 예시는 `Get-DSDirectoryLimit`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 us-east-1 리전에 대한 디렉터리 제한 정보를 표시합니다.

```
Get-DSDirectoryLimit -Region us-east-1
```

출력:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
```

```
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDirectoryLimits을 참조하세요.

`Get-DSDomainControllerList`

다음 코드 예시는 `Get-DSDomainControllerList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 언급된 directory-id에 대해 시작된 도메인 컨트롤러의 세부 목록을 가져옵니다.

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDomainControllers를 참조하세요.

`Get-DSEventTopic`

다음 코드 예시는 `Get-DSEventTopic`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 디렉터리 상태가 변경될 때 알림을 보내도록 구성된 SNS 주제의 정보를 보여줍니다.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEventTopics을 참조하세요.

`Get-DSIpRouteList`

다음 코드 예시는 `Get-DSIpRouteList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 디렉터리 IP 라우팅에 구성된 퍼블릭 IP 주소 블록을 가져옵니다.

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListIpRoutes를 참조하세요.

Get-DSLDAPSSetting

다음 코드 예시는 Get-DSLDAPSSetting의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 디렉터리에 대한 LDAP 보안 상태를 설명합니다.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

출력:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLDAPSSettings을 참조하세요.

Get-DSLogSubscriptionList

다음 코드 예시는 Get-DSLogSubscriptionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 directory-id의 로그 구독 정보를 가져옵니다.

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

출력:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 ---------------------------
```

```
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListLogSubscriptions을 참조하세요.

Get-DSResourceTag

다음 코드 예시는 Get-DSResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 디렉터리의 모든 태그를 가져옵니다.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

출력:

```
Key   Value
---   -----
myTag myTagValue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

Get-DSSchemaExtension

다음 코드 예시는 Get-DSSchemaExtension의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Microsoft AD 디렉터리에 적용된 모든 스키마 확장을 나열합니다.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListSchemaExtensions을 참조하세요.

Get-DSSharedDirectory

다음 코드 예시는 Get-DSSharedDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 AWS 계정의 공유 디렉터리를 가져옵니다.

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSharedDirectories를 참조하세요.

Get-DSSnapshot

다음 코드 예시는 Get-DSSnapshot의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 이 계정에 속한 지정된 디렉터리 스냅샷의 정보를 가져옵니다.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSnapshots을 참조하세요.

`Get-DSSnapshotLimit`

다음 코드 예시는 `Get-DSSnapshotLimit`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 디렉터리에 대한 수동 스냅샷 제한을 가져옵니다.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

출력:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetSnapshotLimits을 참조하세요.

Get-DSTrust

다음 코드 예시는 Get-DSTrust의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 directory-id에 대해 생성된 신뢰 관계 정보를 가져옵니다.

```
Get-DSTrust -DirectoryId d-123456abcd
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrusts를 참조하세요.

New-DSAlias

다음 코드 예시는 New-DSAlias의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 디렉터리에 대한 별칭을 생성하고 별칭을 지정된 directory-id에 할당합니다.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

출력:

```
Alias     DirectoryId
-----     -----------
```

```
myorgname d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAlias를 참조하세요.

New-DSComputer

다음 코드 예시는 New-DSComputer의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 새 Active Directory 컴퓨터 객체를 생성합니다.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

출력:

```
ComputerAttributes                  ComputerId
 ComputerName
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateComputer를 참조하세요.

New-DSConditionalForwarder

다음 코드 예시는 New-DSConditionalForwarder의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 AWS Directory-id에 조건부 전달자를 생성합니다.

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateConditionalForwarder를 참조하세요.

New-DSDirectory

다음 코드 예시는 New-DSDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 새 Simple AD 디렉터리를 생성합니다.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDirectory를 참조하세요.

New-DSLogSubscription

다음 코드 예시는 New-DSLogSubscription의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 실시간 Directory Service 도메인 컨트롤러 보안 로그를 AWS 계정에 지정된 Amazon CloudWatch 로그 그룹으로 전달하기 위한 구독을 생성합니다.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLogSubscription을 참조하세요.

New-DSMicrosoftAD

다음 코드 예시는 New-DSMicrosoftAD의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는에서 새 Microsoft AD 디렉터리를 생성합니다 AWS 클라우드.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateMicrosoftAD를 참조하세요.

New-DSSnapshot

다음 코드 예시는 New-DSSnapshot의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리 스냅샷을 생성합니다.

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSnapshot을 참조하세요.

New-DSTrust

다음 코드 예시는 New-DSTrust의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS 관리형 Microsoft AD 디렉터리와 기존 온프레미스 Microsoft Active Directory 간에 양방향 포리스트 전체 신뢰를 생성합니다.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

출력:

```
t-9067157123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTrust를 참조하세요.

Register-DSCertificate

다음 코드 예시는 Register-DSCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 보안 LDAP 연결에 대해 인증서를 등록합니다.

```
$Certificate = Get-Content contoso.cer -Raw
```

```
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

출력:

```
c-906731e350
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterCertificate를 참조하세요.

Register-DSEventTopic

다음 코드 예시는 Register-DSEventTopic의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 디렉터리를 게시자로 하여 SNS 주제에 연결합니다.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterEventTopic을 참조하세요.

Remove-DSConditionalForwarder

다음 코드 예시는 Remove-DSConditionalForwarder의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS Direcotry에 대해 설정된 조건부 전달자를 제거합니다.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteConditionalForwarder를 참조하세요.

Remove-DSDirectory

다음 코드 예시는 Remove-DSDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS 디렉터리 서비스 디렉터리(Simple AD/Microsoft AD/AD Connector)를 삭제합니다.

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDirectory를 참조하세요.

`Remove-DSIpRoute`

다음 코드 예시는 `Remove-DSIpRoute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 Directory-id에 대해 구성된 IP 경로에서 지정된 IP를 제거합니다.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveIpRoutes를 참조하세요.

`Remove-DSLogSubscription`

다음 코드 예시는 `Remove-DSLogSubscription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 Directory-id의 로그 구독을 제거합니다.

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLogSubscription을 참조하세요.

`Remove-DSResourceTag`

다음 코드 예시는 `Remove-DSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 Directory-id에 할당된 리소스 태그를 제거합니다.

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveTagsFromResource를 참조하세요.

`Remove-DSSnapshot`

다음 코드 예시는 `Remove-DSSnapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 수동으로 생성된 스냅샷을 제거합니다.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSnapshot을 참조하세요.

`Remove-DSTrust`

다음 코드 예시는 `Remove-DSTrust`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS Managed AD Directory와 외부 도메인 간에 존재하는 신뢰 관계 힙을 제거합니다.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

출력:

```
t-9067157123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTrust를 참조하세요.

`Reset-DSUserPassword`

다음 코드 예시는 `Reset-DSUserPassword`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS Managed microsoft AD 또는 Simple AD Directory에서 ADUser라는 Active Direcotry 사용자의 암호를 재설정합니다.

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResetUserPassword를 참조하세요.

`Restore-DSFromSnapshot`

다음 코드 예시는 `Restore-DSFromSnapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기존 디렉터리 스냅샷을 사용하여 디렉터리를 복원합니다.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RestoreFromSnapshot을 참조하세요.

`Set-DSDomainControllerCount`

다음 코드 예시는 `Set-DSDomainControllerCount`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 directory-id에 대해 도메인 컨트롤러 수를 3으로 설정합니다.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateNumberOfDomainControllers를 참조하세요.

`Start-DSSchemaExtension`

다음 코드 예시는 `Start-DSSchemaExtension`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Microsoft AD 디렉터리에 스키마 확장을 적용합니다.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

출력:

```
e-9067306643
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartSchemaExtension을 참조하세요.

`Stop-DSSchemaExtension`

다음 코드 예시는 `Stop-DSSchemaExtension`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Microsoft AD 디렉터리에 대해 진행 중인 스키마 확장을 취소합니다.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelSchemaExtension을 참조하세요.

`Unregister-DSCertificate`

다음 코드 예시는 `Unregister-DSCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 보안 LDAP 연결을 위해 등록된 인증서를 시스템에서 삭제합니다.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterCertificate를 참조하세요.

`Unregister-DSEventTopic`

다음 코드 예시는 `Unregister-DSEventTopic`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 디렉터리를 지정된 SNS 주제의 게시자에서 제거합니다.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterEventTopic을 참조하세요.

`Update-DSConditionalForwarder`

다음 코드 예시는 `Update-DSConditionalForwarder`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS 디렉터리에 대해 설정된 조건부 전달자를 업데이트합니다.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateConditionalForwarder를 참조하세요.

`Update-DSRadius`

다음 코드 예시는 `Update-DSRadius`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 AD Connector 또는 Microsoft AD 디렉터리에 대한 RADIUS 서버 정보를 업데이트합니다.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateRadius를 참조하세요.

Update-DSTrust

다음 코드 예시는 Update-DSTrust의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 trust-id의 SelectiveAuth 파라미터를 Disabled에서 Enabled로 업데이트합니다.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

출력:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateTrust를 참조하세요.

AWS DMS Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS DMS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

New-DMSReplicationTask

다음 코드 예시는 New-DMSReplicationTask의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 CdcStartPosition 대신 CdcStartTime을 사용하는 새 AWS 데이터베이스 마이그레이션 서비스 복제 작업을 생성합니다. CdcStartPosition MigrationType은 'full-load-and-cdc'로 설정되어 있습니다. 즉, 대상 테이블이 비어 있어야 합니다. 새 작업에는 키가 Stage이고 키 값이 Test인 태그가 지정됩니다. 이 cmdlet에서 사용하는 값에 대한 자세한 내용은 AWS Database Migration Service 사용 설명서의 작업 생성(https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html)을 참조하세요.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateReplicationTask를 참조하세요.

Tools for PowerShell V4를 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

Add-DDBIndexSchema

다음 코드 예시는 Add-DDBIndexSchema의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 빈 TableSchema 객체를 생성하고 파이프라인에 TableSchema 객체를 쓰기 전에 새 로컬 보조 인덱스 정의를 추가합니다.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

출력:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

예제 2: 파이프라인에 TableSchema 객체를 다시 쓰기 전에 제공된 TableSchema 객체에 새 로컬 보조 인덱스 정의를 추가합니다. TableSchema 객체는 -Schema 파라미터를 사용하여 제공할 수도 있습니다.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

출력:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{LastPostDateTime}                          {}
    {LastPostIndex}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Add-DDBIndexSchema를 참조하세요.

Add-DDBKeySchema

다음 코드 예시는 Add-DDBKeySchema의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 빈 TableSchema 객체를 생성하고 파이프라인에 TableSchema 객체를 쓰기 전에 지정된 키 데이터를 사용하여 키 및 속성 정의 항목을 추가합니다. 키 유형은 기본적으로 'HASH'로 선언됩니다. 값이 'RANGE'인 -KeyType 파라미터를 사용하여 범위 키를 선언합니다.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

출력:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

예제 2: 파이프라인에 TableSchema 객체를 쓰기 전에 제공된 TableSchema 객체에 새 키 및 속성 정의 항목을 추가합니다. 키 유형은 기본적으로 'HASH'로 선언됩니다. 값이 'RANGE'인 -KeyType 파라미터를 사용하여 범위 키를 선언합니다. TableSchema 객체는 -Schema 파라미터를 사용하여 제공할 수도 있습니다.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

출력:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{ForumName}                                 {ForumName}
    {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Add-DDBKeySchema를 참조하세요.

`ConvertFrom-DDBItem`

다음 코드 예시는 `ConvertFrom-DDBItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ConvertFrom-DDBItem은 Get-DDBItem의 결과를 DynamoDB AttributeValues의 해시 테이블에서 string 및 double과 같은 일반적인 유형의 해시 테이블로 변환하는 데 사용됩니다.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

출력:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ConvertFrom-DDBItem을 참조하세요.

`ConvertTo-DDBItem`

다음 코드 예시는 `ConvertTo-DDBItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 해시테이블을 DynamoDB 속성 값의 사전으로 변환하는 예제입니다.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

예제 2: 해시테이블을 DynamoDB 속성 값의 사전으로 변환하는 예제입니다.

```
@{
    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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ConvertTo-DDBItem을 참조하세요.

Get-DDBBatchItem

다음 코드 예시는 Get-DDBBatchItem의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: DynamoDB 테이블 'Music' 및 'Songs'에서 노래 제목이 'Somewhere Down The Road'인 항목을 가져옵니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchGetItem을 참조하세요.

Get-DDBItem

다음 코드 예시는 Get-DDBItem의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 파티션 키 SongTitle과 정렬 키 Artist가 있는 DynamoDB 항목을 반환합니다.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

출력:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetItem을 참조하세요.

Get-DDBTable

다음 코드 예시는 Get-DDBTable의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 테이블의 세부 정보를 반환합니다.

```
Get-DDBTable -TableName "myTable"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTable을 참조하세요.

`Get-DDBTableList`

다음 코드 예시는 `Get-DDBTableList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 서비스에 더 이상 테이블이 없다고 표시될 때까지 자동으로 반복하여 모든 테이블의 세부 정보를 반환합니다.

```
Get-DDBTableList
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTables를 참조하세요.

`Invoke-DDBQuery`

다음 코드 예시는 `Invoke-DDBQuery`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 SongTitle 및 Artist와 함께 DynamoDB 항목을 반환하는 쿼리를 간접 호출합니다.

```
$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
```

출력:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
```

```
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Query를 참조하세요.

`Invoke-DDBScan`

다음 코드 예시는 `Invoke-DDBScan`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: Music 테이블에서 모든 항목을 반환합니다.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

출력:

```
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
```

예 2: Music 테이블에서 CriticRating이 9 이상인 항목을 반환합니다.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
```

```
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

출력:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Scan을 참조하세요.

`New-DDBTable`

다음 코드 예시는 `New-DDBTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예시는 프라이머리 키가 'ForumName'(키 유형 해시) 및 'Subject'(키 유형 범위)로 구성된 Thread라는 테이블을 만듭니다. 테이블을 구성하는 데 사용되는 스키마는 표시된 대로 또는 -Schema 파라미터를 사용하여 지정한 대로 각 cmdlet에 파이프로 연결할 수 있습니다.

```
$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
```

출력:

```
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  : {}
```

예 2: 이 예시는 프라이머리 키가 'ForumName'(키 유형 해시) 및 'Subject'(키 유형 범위)로 구성된 Thread라는 테이블을 만듭니다. 로컬 보조 인덱스도 정의됩니다. 로컬 보조 인덱스의 키는 테이블의 프라이머리 해시 키(ForumName)에서 자동으로 설정됩니다. 테이블을 구성하는 데 사용되는 스키마는 표시된 대로 또는 -Schema 파라미터를 사용하여 지정한 대로 각 cmdlet에 파이프로 연결할 수 있습니다.

```
$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
```

출력:

```
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}
```

예 3: 이 예시는 단일 파이프라인을 사용하여 프라이머리 키가 'ForumName'(키 유형 해시) 및 'Subject'(키 유형 범위)로 구성된 Thread라는 테이블을 만드는 방법을 보여줍니다. Add-DDBKeySchema 및 Add-DDBIndexSchema는 파이프라인 또는 -Schema 파라미터에서 TableSchema 객체가 제공되지 않는 경우 자동으로 새 TableSchema 객체를 만듭니다.

```
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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTable을 참조하세요.

New-DDBTableSchema

다음 코드 예시는 New-DDBTableSchema의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 새 Amazon DynamoDB 테이블 생성에 사용할 키 및 인덱스 정의를 수락할 준비가 된 빈 TableSchema 객체를 생성합니다. 반환된 객체는 Add-DDBKeySchema, Add-DDBIndexSchema, New-DDBTable cmdlet에 파이프하거나 각 cmdlet의 -Schema 파라미터를 사용하여 전달할 수 있습니다.

```
New-DDBTableSchema
```

출력:

```
AttributeSchema                             KeySchema
    LocalSecondaryIndexSchema
---------------                             ---------
    -------------------------
{}                                          {}
    {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 New-DDBTableSchema를 참조하세요.

Remove-DDBItem

다음 코드 예시는 Remove-DDBItem의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 제공된 키와 일치하는 DynamoDB 항목을 제거합니다.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteItem을 참조하세요.

`Remove-DDBTable`

다음 코드 예시는 `Remove-DDBTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 테이블을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-DDBTable -TableName "myTable"
```

예 2: 지정된 테이블을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-DDBTable -TableName "myTable" -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTable을 참조하세요.

`Set-DDBBatchItem`

다음 코드 예시는 `Set-DDBBatchItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 새 항목을 생성하거나, DynamoDB 테이블 Music 및 Songs의 새 항목으로 기존 항목을 바꿉니다.

```
$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

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 BatchWriteItem을 참조하세요.

`Set-DDBItem`

다음 코드 예시는 `Set-DDBItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 새 항목을 생성하거나 새 항목으로 기존 항목을 바꿉니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutItem을 참조하세요.

`Update-DDBItem`

다음 코드 예시는 `Update-DDBItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 파티션 키 SongTitle과 정렬 키 Artist가 있는 DynamoDB 항목에서 장르 속성을 'Rap'으로 설정합니다.

```
$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
```

출력:

```
Name                           Value
----                           -----
Genre                          Rap
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateItem을 참조하세요.

`Update-DDBTable`

다음 코드 예시는 `Update-DDBTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 주어진 테이블의 프로비저닝된 처리량을 업데이트합니다.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateTable을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon EC2 예제

다음 코드 예제에서는 Amazon EC2와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-EC2CapacityReservation`

다음 코드 예시는 `Add-EC2CapacityReservation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 속성을 사용하여 새 용량 예약을 생성합니다.

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCapacityReservation을 참조하세요.

Add-EC2InternetGateway

다음 코드 예시는 Add-EC2InternetGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인터넷 게이트웨이를 특정 VPC에 연결합니다.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

예제 2: 이 예제에서는 VPC와 인터넷 게이트웨이를 생성한 다음 인터넷 게이트웨이를 VPC에 연결합니다.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachInternetGateway를 참조하세요.

Add-EC2NetworkInterface

다음 코드 예시는 Add-EC2NetworkInterface의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1 : 이 예제에서는 지정된 네트워크 인터페이스를 지정된 인스턴스에 연결합니다.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

출력:

```
eni-attach-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachNetworkInterface를 참조하세요.

Add-EC2Volume

다음 코드 예시는 Add-EC2Volume의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨을 지정된 인스턴스에 연결하고 지정된 디바이스 이름으로 노출합니다.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

출력:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachVolume을 참조하세요.

Add-EC2VpnGateway

다음 코드 예시는 Add-EC2VpnGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이를 지정된 VPC에 연결합니다.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

출력:

```
State        VpcId
```

```
-----        -----
attaching    vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachVpnGateway를 참조하세요.

Approve-EC2VpcPeeringConnection

다음 코드 예시는 Approve-EC2VpcPeeringConnection의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 요청된 VpcPeeringConnectionId pcx-1dfad234b56ff78be를 승인합니다.

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AcceptVpcPeeringConnection을 참조하세요.

Confirm-EC2ProductInstance

다음 코드 예시는 Confirm-EC2ProductInstance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 제품 코드가 지정된 인스턴스와 연관되어 있는지 여부를 확인합니다.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ConfirmProductInstance를 참조하세요.

Copy-EC2Image

다음 코드 예시는 Copy-EC2Image의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'EU(아일랜드)' 리전의 지정된 AMI를 '미국 서부(오리건)' 리전에 복사합니다. -Region을 지정하지 않으면 현재 기본 리전이 대상 리전으로 사용됩니다.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2
 -Name "Copy of ami-12345678"
```

출력:

```
ami-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CopyImage를 참조하세요.

Copy-EC2Snapshot

다음 코드 예시는 Copy-EC2Snapshot의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷을 EU(아일랜드) 리전에서 미국 서부(오리건) 리전으로 복사합니다.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-
west-2
```

예제 2: 기본 리전을 설정하고 Region 파라미터를 생략하면 기본 대상 리전이 기본 리전이 됩니다.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CopySnapshot을 참조하세요.

Deny-EC2VpcPeeringConnection

다음 코드 예시는 Deny-EC2VpcPeeringConnection의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 위 예제에서는 VpcPeering request id pcx-01a2b3ce45fe67eb8에 대한 요청을 거부합니다.

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RejectVpcPeeringConnection을 참조하세요.

`Disable-EC2VgwRoutePropagation`

다음 코드 예시는 `Disable-EC2VgwRoutePropagation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VGW가 지정된 라우팅 테이블에 경로를 자동으로 전파하지 못하도록 합니다.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableVgwRoutePropagation을 참조하세요.

`Disable-EC2VpcClassicLink`

다음 코드 예시는 `Disable-EC2VpcClassicLink`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 vpc-01e23c4a5d6db78e9에 대한 EC2VpcClassicLink를 비활성화합니다. 그러면 True 또는 False가 반환됩니다.

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableVpcClassicLink를 참조하세요.

`Disable-EC2VpcClassicLinkDnsSupport`

다음 코드 예시는 `Disable-EC2VpcClassicLinkDnsSupport`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 vpc-0b12d3456a7e8910d에 대한 ClassicLink DNS 지원을 비활성화합니다.

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableVpcClassicLinkDnsSupport를 참조하세요.

`Dismount-EC2InternetGateway`

다음 코드 예시는 `Dismount-EC2InternetGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인터넷 게이트웨이를 지정된 VPC에서 분리합니다.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachInternetGateway를 참조하세요.

`Dismount-EC2NetworkInterface`

다음 코드 예시는 `Dismount-EC2NetworkInterface`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 네트워크 인터페이스와 인스턴스 간에 지정된 연결을 제거합니다.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachNetworkInterface를 참조하세요.

`Dismount-EC2Volume`

다음 코드 예시는 `Dismount-EC2Volume`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨을 분리합니다.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

출력:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

예제 2: 인스턴스 ID와 디바이스 이름을 지정하여 올바른 볼륨을 분리할 수도 있습니다.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachVolume을 참조하세요.

`Dismount-EC2VpnGateway`

다음 코드 예시는 `Dismount-EC2VpnGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이를 지정된 VPC에서 분리합니다.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachVpnGateway를 참조하세요.

`Edit-EC2CapacityReservation`

다음 코드 예시는 `Edit-EC2CapacityReservation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 수를 1로 변경하여 CapacityReservationId cr-0c1f2345db6f7cdba를 수정합니다.

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyCapacityReservation을 참조하세요.

`Edit-EC2Host`

다음 코드 예시는 `Edit-EC2Host`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 전용 호스트 h-01e23f4cd567890f3에 대한 AutoPlacement 설정을 off로 수정합니다.

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

출력:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyHosts를 참조하세요.

`Edit-EC2IdFormat`

다음 코드 예시는 `Edit-EC2IdFormat`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스 유형에 대해 더 긴 ID 형식을 활성화합니다.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

예제 2: 이 예제에서는 지정된 리소스 유형에 대해 더 긴 ID 형식을 비활성화합니다.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyIdFormat을 참조하세요.

`Edit-EC2ImageAttribute`

다음 코드 예시는 `Edit-EC2ImageAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 AMI에 대한 설명을 업데이트합니다.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

예제 2:이 예제에서는 AMI를 퍼블릭으로 설정합니다(예: 누구나 사용할 AWS 계정 수 있도록).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

예제 3: 이 예제에서는 AMI를 프라이빗으로 설정합니다(예를 들어, 소유자만 사용할 수 있도록).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

예제 4:이 예제에서는 지정된에 시작 권한을 부여합니다 AWS 계정.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

예제 5:이 예제는 지정된에서 시작 권한을 제거합니다 AWS 계정.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyImageAttribute를 참조하세요.

Edit-EC2InstanceAttribute

다음 코드 예시는 Edit-EC2InstanceAttribute의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스의 인스턴스 유형을 수정합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

예제 2: 이 예제에서는 'simple'을 단일 루트 I/O 가상화(SR-IOV) 네트워크 지원 파라미터인 -SriovNetSupport 값으로 지정하여 지정된 인스턴스에 대해 향상된 네트워킹을 활성화합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

예제 3: 이 예제에서는 지정된 인스턴스의 보안 그룹을 수정합니다. 인스턴스가 VPC에 있어야 합니다. 이름이 아닌 각 보안 그룹의 ID를 지정해야 합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

예제 4: 이 예제에서는 지정된 인스턴스에 대해 EBS I/O 최적화를 활성화합니다. 이 기능은 일부 인스턴스 유형에서는 사용할 수 없습니다. EBS 최적화 인스턴스를 사용할 때 추가 사용 요금이 적용됩니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

예제 5: 이 예제에서는 지정된 인스턴스에 대한 소스/대상 확인을 활성화합니다. NAT를 수행하는 NAT 인스턴스의 경우 값이 'false'여야 합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

예제 6: 이 예제에서는 지정된 인스턴스에 대한 종료를 비활성화합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

예제 7: 이 예제에서는 인스턴스 내부에서 종료가 시작될 때 인스턴스 자체를 종료하도록 지정된 인스턴스를 변경합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyInstanceAttribute를 참조하세요.

`Edit-EC2InstanceCreditSpecification`

다음 코드 예시는 `Edit-EC2InstanceCreditSpecification`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 i-01234567890abcdef에 대한 T2 무제한 크레딧을 활성화합니다.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyInstanceCreditSpecification을 참조하세요.

`Edit-EC2NetworkInterfaceAttribute`

다음 코드 예시는 `Edit-EC2NetworkInterfaceAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 종료 시 지정된 연결이 삭제되도록 지정된 네트워크 인터페이스를 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

예제 2: 이 예제에서는 지정된 네트워크 인터페이스의 설명을 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

예제 3: 이 예제에서는 지정된 네트워크 인터페이스의 보안 그룹을 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

예제 4: 이 예제에서는 지정된 네트워크 인터페이스에 대한 소스/대상 확인을 비활성화합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyNetworkInterfaceAttribute를 참조하세요.

`Edit-EC2ReservedInstance`

다음 코드 예시는 `Edit-EC2ReservedInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 예약 인스턴스의 가용 영역, 인스턴스 수, 플랫폼을 수정합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyReservedInstances를 참조하세요.

`Edit-EC2SnapshotAttribute`

다음 코드 예시는 `Edit-EC2SnapshotAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 CreateVolumePermission 속성을 설정하여 지정된 스냅샷을 퍼블릭으로 만듭니다.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifySnapshotAttribute를 참조하세요.

`Edit-EC2SpotFleetRequest`

다음 코드 예시는 `Edit-EC2SpotFleetRequest`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 플릿 요청의 목표 용량을 업데이트합니다.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifySpotFleetRequest를 참조하세요.

`Edit-EC2SubnetAttribute`

다음 코드 예시는 `Edit-EC2SubnetAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 서브넷에 대한 퍼블릭 IP 주소 지정을 활성화합니다.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

예제 2: 이 예제에서는 지정된 서브넷에 대한 퍼블릭 IP 주소 지정을 비활성화합니다.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifySubnetAttribute를 참조하세요.

`Edit-EC2VolumeAttribute`

다음 코드 예시는 `Edit-EC2VolumeAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨의 지정된 속성을 수정합니다. 일관성이 떨어질 수 있는 데이터로 인해 일시 중지된 후 볼륨에 대한 I/O 작업이 자동으로 재개됩니다.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyVolumeAttribute를 참조하세요.

`Edit-EC2VpcAttribute`

다음 코드 예시는 `Edit-EC2VpcAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC에 대해 DNS 호스트 이름 지원을 활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

예제 2: 이 예제에서는 지정된 VPC에 대해 DNS 호스트 이름 지원을 비활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

예제 3: 이 예제에서는 지정된 VPC에 대해 DNS 해석 지원을 활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

예제 4: 이 예제에서는 지정된 VPC에 대해 DNS 해석 지원을 비활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyVpcAttribute를 참조하세요.

`Enable-EC2VgwRoutePropagation`

다음 코드 예시는 `Enable-EC2VgwRoutePropagation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VGW가 지정된 라우팅 테이블에 경로를 자동으로 전파할 수 있도록 합니다.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableVgwRoutePropagation을 참조하세요.

`Enable-EC2VolumeIO`

다음 코드 예시는 `Enable-EC2VolumeIO`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 I/O 작업이 비활성화된 경우 지정된 볼륨에 대한 I/O 작업을 활성화합니다.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableVolumeIo를 참조하세요.

`Enable-EC2VpcClassicLink`

다음 코드 예시는 `Enable-EC2VpcClassicLink`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 ClassicLink에 대해 VPC vpc-0123456b789b0d12f를 활성화합니다.

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableVpcClassicLink를 참조하세요.

`Enable-EC2VpcClassicLinkDnsSupport`

다음 코드 예시는 `Enable-EC2VpcClassicLinkDnsSupport`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 vpc-0b12d3456a7e8910d가 ClassicLink에 대한 DNS 호스트 이름 해석 지원을 활성화합니다.

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableVpcClassicLinkDnsSupport를 참조하세요.

`Get-EC2AccountAttribute`

다음 코드 예시는 `Get-EC2AccountAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리전의 EC2-Classic 및 EC2-VPC에서 인스턴스를 시작할 수 있는지 아니면 EC2-VPC에서만 인스턴스를 시작할 수 있는지 설명합니다.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

출력:

```
AttributeValue
--------------
EC2
VPC
```

예제 2: 이 예제에서는 기본 VPC를 설명하며, 해당 리전에 기본 VPC가 없는 경우 'none'으로 표시됩니다.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

출력:

```
AttributeValue
--------------
vpc-12345678
```

예제 3: 이 예제에서는 실행할 수 있는 최대 온디맨드 인스턴스 수를 설명합니다.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

출력:

```
AttributeValue
--------------
20
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAccountAttributes를 참조하세요.

`Get-EC2Address`

다음 코드 예시는 `Get-EC2Address`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-Classic의 인스턴스에 대해 지정된 탄력적 IP 주소를 설명합니다.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 VPC의 인스턴스에 대한 탄력적 IP 주소를 설명합니다. 이 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

예제 3: 이 예제에서는 EC2-Classic의 인스턴스에 대해 지정된 탄력적 IP 주소를 설명합니다.

```
Get-EC2Address -PublicIp 203.0.113.17
```

출력:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

예제 4: 이 예제에서는 EC2-Classic의 인스턴스에 대한 탄력적 IP 주소를 설명합니다. 이 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

예제 5: 이 예제에서는 모든 탄력적 IP 주소를 설명합니다.

```
Get-EC2Address
```

예제 6: 이 예제에서는 필터에 제공된 인스턴스 ID의 퍼블릭 및 프라이빗 IP를 반환합니다.

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

출력:

```
PrivateIpAddress PublicIp
---------------- --------
```

```
10.0.0.99        63.36.5.227
```

예제 7: 이 예제에서는 할당 ID, 연결 ID, 인스턴스 ID가 있는 모든 탄력적 IP를 검색합니다.

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

출력:

```
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
```

예제 8: 이 예제에서는 태그 키가 'Category'이고 값이 'Prod'인 EC2 IP 주소 목록을 가져옵니다.

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAddresses를 참조하세요.

Get-EC2AvailabilityZone

다음 코드 예시는 Get-EC2AvailabilityZone의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 현재 리전의 가용 영역을 설명합니다.

```
Get-EC2AvailabilityZone
```

출력:

```
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
```

예제 2: 이 예제에서는 손상된 상태의 가용 영역을 설명합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

예제 3: PowerShell 버전 2에서 필터를 생성하려면 New-Object를 사용해야 합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAvailabilityZones을 참조하세요.

Get-EC2BundleTask

다음 코드 예시는 Get-EC2BundleTask의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 번들 작업을 설명합니다.

```
Get-EC2BundleTask -BundleId bun-12345678
```

예제 2: 이 예제에서는 상태가 'complete' 또는 'failed'인 번들 작업을 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeBundleTasks를 참조하세요.

Get-EC2CapacityReservation

다음 코드 예시는 Get-EC2CapacityReservation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리전에 대한 하나 이상의 용량 예약을 설명합니다.

```
Get-EC2CapacityReservation -Region eu-west-1
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCapacityReservations을 참조하세요.

`Get-EC2ConsoleOutput`

다음 코드 예시는 `Get-EC2ConsoleOutput`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 Linux 인스턴스에 대한 콘솔 출력을 가져옵니다. 콘솔 출력이 인코딩됩니다.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

출력:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

예제 2: 이 예제에서는 인코딩된 콘솔 출력을 변수에 저장한 다음 디코딩합니다.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetConsoleOutput을 참조하세요.

`Get-EC2CustomerGateway`

다음 코드 예시는 `Get-EC2CustomerGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 고객 게이트웨이를 설명합니다.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

출력:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

예제 2: 이 예제에서는 상태가 pending이거나 available인 모든 고객 게이트웨이를 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

예제 3: 이 예제에서는 모든 고객 게이트웨이를 설명합니다.

```
Get-EC2CustomerGateway
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCustomerGateways를 참조하세요.

`Get-EC2DhcpOption`

다음 코드 예시는 `Get-EC2DhcpOption`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 DHCP 옵션 세트를 나열합니다.

```
Get-EC2DhcpOption
```

출력:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

예제 2: 이 예제에서는 지정된 DHCP 옵션 세트에 대한 구성 세부 정보를 가져옵니다.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

출력:

```
Key                  Values
---                  ------
domain-name          {abc.local}
domain-name-servers  {10.0.0.101, 10.0.0.102}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDhcpOptions을 참조하세요.

Get-EC2FlowLog

다음 코드 예시는 Get-EC2FlowLog의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 로그 대상 유형이 's3'인 하나 이상의 흐름 로그를 설명합니다.

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

출력:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
```

```
LogGroupName              :
ResourceId                : eni-01d2dda3456b7e890
TrafficType               : ALL
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeFlowLogs를 참조하세요.

Get-EC2Host

다음 코드 예시는 Get-EC2Host의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2 호스트 세부 정보를 반환합니다.

```
Get-EC2Host
```

출력:

```
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              : {}
```

예제 2: 이 예제에서는 호스트 h-01e23f4cd567899f1에 대한 AvailableInstanceCapacity를 쿼리합니다.

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

출력:

```
AvailableCapacity InstanceType TotalCapacity
```

```
---------------- ------------ -------------
11               m4.xlarge    11
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeHosts를 참조하세요.

`Get-EC2HostReservationOffering`

다음 코드 예시는 `Get-EC2HostReservationOffering`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 PaymentOption이 'NoUpfront'인 지정된 'instance-family' 필터에 대해 구매할 수 있는 전용 호스트 예약을 설명합니다.

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeHostReservationOfferings을 참조하세요.

`Get-EC2HostReservationPurchasePreview`

다음 코드 예시는 `Get-EC2HostReservationPurchasePreview`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 전용 호스트 h-01e23f4cd567890f1의 구성과 일치하는 구성의 예약 구매를 미리 봅니다.

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

출력:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetHostReservationPurchasePreview를 참조하세요.

`Get-EC2IdFormat`

다음 코드 예시는 `Get-EC2IdFormat`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스 유형의 ID 형식을 설명합니다.

```
Get-EC2IdFormat -Resource instance
```

출력:

```
Resource       UseLongIds
--------       ----------
instance       False
```

예제 2: 이 예제에서는 더 긴 ID를 지원하는 모든 리소스 유형의 ID 형식을 설명합니다.

```
Get-EC2IdFormat
```

출력:

```
Resource       UseLongIds
--------       ----------
```

```
reservation    False
instance       False
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeIdFormat을 참조하세요.

`Get-EC2IdentityIdFormat`

다음 코드 예시는 `Get-EC2IdentityIdFormat`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 역할에 대한 리소스 'image'의 ID 형식을 반환합니다.

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

출력:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeIdentityIdFormat을 참조하세요.

`Get-EC2Image`

다음 코드 예시는 `Get-EC2Image`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 AMI를 설명합니다.

```
Get-EC2Image -ImageId ami-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 사용자가 소유한 AMI를 설명합니다.

```
Get-EC2Image -owner self
```

예제 3: 이 예제에서는 Microsoft Windows Server를 실행하는 퍼블릭 AMI를 설명합니다.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

예제 4: 이 예제에서는 'us-west-2' 리전의 모든 퍼블릭 AMI를 설명합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImages를 참조하세요.

`Get-EC2ImageAttribute`

다음 코드 예시는 `Get-EC2ImageAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 AMI에 대한 설명을 가져옵니다.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

출력:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

예제 2: 이 예제에서는 지정된 AMI에 대한 시작 권한을 가져옵니다.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

출력:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

예제 3: 이 예제에서는 향상된 네트워킹이 활성화되어 있는지 테스트합니다.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

출력:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
```

```
SriovNetSupport     : simple
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImageAttribute를 참조하세요.

`Get-EC2ImageByName`

다음 코드 예시는 `Get-EC2ImageByName`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 현재 지원되는 전체 필터 이름 세트를 설명합니다.

```
Get-EC2ImageByName
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 AMI를 설명합니다. 이 명령을 사용하여 AMI를 찾는 것이 유용합니다.는 매월 최신 업데이트가 포함된 새 Windows AMIs를 AWS 출시하기 때문입니다. New-EC2Instance에 'ImageId'를 지정하여 지정된 필터에 대한 현재 AMI를 사용하여 인스턴스를 시작할 수 있습니다.

```
Get-EC2ImageByName -Names WINDOWS_2016_BASE
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Get-EC2ImageByName을 참조하세요.

`Get-EC2ImportImageTask`

다음 코드 예시는 `Get-EC2ImportImageTask`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이미지 가져오기 작업을 설명합니다.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

출력:

```
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   :
```

예제 2: 이 예제에서는 모든 이미지 가져오기 작업을 설명합니다.

```
Get-EC2ImportImageTask
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImportImageTasks를 참조하세요.

`Get-EC2ImportSnapshotTask`

다음 코드 예시는 `Get-EC2ImportSnapshotTask`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷 가져오기 작업을 설명합니다.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

출력:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------
```

```
Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

예제 2: 이 예제에서는 모든 스냅샷 가져오기 작업을 설명합니다.

```
Get-EC2ImportSnapshotTask
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeImportSnapshotTasks를 참조하세요.

`Get-EC2Instance`

다음 코드 예시는 `Get-EC2Instance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 설명합니다.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

출력:

```
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
```

예제 2: 이 예제에서는 예약별로 그룹화된 현재 리전의 모든 인스턴스를 설명합니다. 인스턴스 세부 정보를 보려면 각 예약 객체 내에서 인스턴스 컬렉션을 확장합니다.

```
Get-EC2Instance
```

출력:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c
```

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

예제 3: 이 예제에서는 필터를 사용하여 VPC의 특정 서브넷에서 EC2 인스턴스를 쿼리하는 방법을 보여줍니다.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

출력:

```
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
```

예제 4: 이 예제에서는 여러 값이 있는 필터를 사용하여 running 및 stopped 상태의 EC2 인스턴스를 쿼리하는 방법을 보여줍니다.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

출력:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
```

```
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

예제 5: 이 예제에서는 여러 값이 있는 필터를 사용하여 running 및 stopped 상태의 EC2 인스턴스를 쿼리하고 Select-Object cmdlet을 사용하여 출력할 특정 값을 선택하는 방법을 보여줍니다.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

출력:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
```

```
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstances를 참조하세요.

Get-EC2InstanceAttribute

다음 코드 예시는 Get-EC2InstanceAttribute의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스의 인스턴스 유형을 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

출력:

```
InstanceType                     : t2.micro
```

예제 2: 이 예제에서는 지정된 인스턴스에 대해 향상된 네트워킹이 활성화되었는지 여부를 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

출력:

```
SriovNetSupport                  : simple
```

예제 3: 이 예제에서는 지정된 인스턴스의 보안 그룹을 설명합니다.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

출력:

```
GroupId
-------
sg-12345678
sg-45678901
```

예제 4: 이 예제에서는 지정된 인스턴스에 대해 EBS 최적화가 활성화되었는지 여부를 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

출력:

```
EbsOptimized                     : False
```

예제 5: 이 예제에서는 지정된 인스턴스의 'disableApiTermination' 속성을 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

출력:

```
DisableApiTermination            : False
```

예제 6: 이 예제에서는 지정된 인스턴스의 'instanceInitiatedShutdownBehavior' 속성을 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

출력:

```
InstanceInitiatedShutdownBehavior : stop
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstanceAttribute를 참조하세요.

`Get-EC2InstanceMetadata`

다음 코드 예시는 `Get-EC2InstanceMetadata`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 쿼리할 수 있는 인스턴스 메타데이터의 사용 가능한 범주를 나열합니다.

```
Get-EC2InstanceMetadata -ListCategory
```

출력:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

예제 2: 인스턴스를 시작하는 데 사용된 Amazon Machine Image(AMI)의 ID를 반환합니다.

```
Get-EC2InstanceMetadata -Category AmiId
```

출력:

```
ami-b2e756ca
```

예제 3: 이 예제에서는 인스턴스에 대한 JSON 형식의 ID 문서를 쿼리합니다.

```
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"
}
```

예제 4: 이 예제에서는 경로 쿼리를 사용하여 인스턴스의 네트워크 인터페이스 macs를 가져옵니다.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

출력:

```
02:80:7f:ef:4c:e0/
```

예제 5: 인스턴스에 IAM 역할이 연결되어 있을 경우, 인스턴스의 LastUpdated date, InstanceProfileArn, InstanceProfileId 등 마지막으로 인스턴스 프로파일이 업데이트된 시간 관련 정보를 반환합니다.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

출력:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Get-EC2InstanceMetadata를 참조하세요.

Get-EC2InstanceStatus

다음 코드 예시는 Get-EC2InstanceStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스의 상태를 설명합니다.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

출력:

```
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
```

출력:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

출력:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

출력:

```
Details           Status
-------           ------
{reachability}    ok
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstanceStatus를 참조하세요.

Get-EC2InternetGateway

다음 코드 예시는 Get-EC2InternetGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인터넷 게이트웨이를 설명합니다.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

출력:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

예제 2: 이 예제에서는 모든 인터넷 게이트웨이를 설명합니다.

```
Get-EC2InternetGateway
```

출력:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInternetGateways를 참조하세요.

Get-EC2KeyPair

다음 코드 예시는 Get-EC2KeyPair의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 키 페어를 설명합니다.

```
Get-EC2KeyPair -KeyName my-key-pair
```

출력:

```
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
```

예제 2: 이 예제에서는 모든 키 페어를 설명합니다.

```
Get-EC2KeyPair
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeKeyPairs를 참조하세요.

`Get-EC2NetworkAcl`

다음 코드 예시는 `Get-EC2NetworkAcl`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL을 설명합니다.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

출력:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

예제 2: 이 예제에서는 지정된 네트워크 ACL에 대한 규칙을 설명합니다.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

출력:

```
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
```

예제 3: 이 예제에서는 모든 네트워크 ACL을 설명합니다.

```
Get-EC2NetworkAcl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeNetworkAcls을 참조하세요.

`Get-EC2NetworkInterface`

다음 코드 예시는 `Get-EC2NetworkInterface`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 모든 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterface
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeNetworkInterfaces를 참조하세요.

`Get-EC2NetworkInterfaceAttribute`

다음 코드 예시는 `Get-EC2NetworkInterfaceAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

출력:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

예제 2: 이 예제에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

출력:

```
Description        : My description
```

예제 3: 이 예제에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

출력:

```
Groups             : {my-security-group}
```

예제 4: 이 예제에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

출력:

```
SourceDestCheck    : True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeNetworkInterfaceAttribute를 참조하세요.

`Get-EC2PasswordData`

다음 코드 예시는 `Get-EC2PasswordData`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Amazon EC2가 지정된 Windows 인스턴스의 관리자 계정에 할당한 암호를 해독합니다. pem 파일이 지정되면 -Decrypt 스위치의 설정이 자동으로 수임됩니다.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

출력:

```
mYZ(PA9?C)Q
```

예제 2: (Windows PowerShell만 해당) 인스턴스를 검사하여 인스턴스를 시작하는 데 사용되는 키 페어의 이름을 확인한 다음 AWS Toolkit for Visual Studio의 구성 스토어에서 해당 키 페어 데이터를 찾으려고 시도합니다. 키 페어 데이터가 발견되면 암호가 해독됩니다.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

출력:

```
mYZ(PA9?C)Q
```

예제 3: 인스턴스에 대해 암호화된 암호 데이터를 반환합니다.

```
Get-EC2PasswordData -InstanceId i-12345678
```

출력:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPasswordData를 참조하세요.

`Get-EC2PlacementGroup`

다음 코드 예시는 `Get-EC2PlacementGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배치 그룹을 설명합니다.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

출력:

```
GroupName            State        Strategy
---------            -----        --------
```

```
my-placement-group    available    cluster
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePlacementGroups을 참조하세요.

Get-EC2PrefixList

다음 코드 예시는 Get-EC2PrefixList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 리전 AWS 서비스 에 대해 접두사 목록 형식으로 사용 가능한를 가져옵니다.

```
Get-EC2PrefixList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePrefixLists를 참조하세요.

Get-EC2Region

다음 코드 예시는 Get-EC2Region의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 사용 가능한 리전을 설명합니다.

```
Get-EC2Region
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeRegions을 참조하세요.

`Get-EC2RouteTable`

다음 코드 예시는 `Get-EC2RouteTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 라우팅 테이블을 설명합니다.

```
Get-EC2RouteTable
```

출력:

```
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  :
```

예제 2: 이 예제에서는 지정된 라우팅 테이블에 대한 세부 정보를 반환합니다.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

예제 3: 이 예제에서는 지정된 VPC의 라우팅 테이블을 설명합니다.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

출력:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeRouteTables을 참조하세요.

`Get-EC2ScheduledInstance`

다음 코드 예시는 `Get-EC2ScheduledInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 일정이 예약된 지정된 인스턴스를 설명합니다.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

출력:

```
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
```

예제 2: 이 예제에서는 일정이 예약된 모든 인스턴스를 설명합니다.

```
Get-EC2ScheduledInstance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScheduledInstances를 참조하세요.

`Get-EC2ScheduledInstanceAvailability`

다음 코드 예시는 `Get-EC2ScheduledInstanceAvailability`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 날짜부터 매주 일요일에 발생하는 일정을 설명합니다.

```
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
```

출력:

```
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

...
```

예제 2: 결과의 범위를 좁히려면 운영 체제, 네트워크, 인스턴스 유형과 같은 기준에 대한 필터를 추가할 수 있습니다.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeScheduledInstanceAvailability를 참조하세요.

`Get-EC2SecurityGroup`

다음 코드 예시는 `Get-EC2SecurityGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC에 지정된 보안 그룹을 설명합니다. VPC에 속한 보안 그룹으로 작업할 때는 이름(-GroupName 파라미터)이 아닌 보안 그룹 ID(-GroupId 파라미터)를 사용하여 그룹을 참조해야 합니다.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 EC2-Classic에 지정된 보안 그룹을 설명합니다. EC2-Classic의 보안 그룹으로 작업할 때 그룹 이름(-GroupName 파라미터) 또는 그룹 ID(-GroupId 파라미터)를 사용하여 보안 그룹을 참조할 수 있습니다.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

출력:

```
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               :
```

예제 3: 이 예제에서는 vpc-0fc1ff23456b789eb에 대한 모든 보안 그룹을 검색합니다.

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSecurityGroups을 참조하세요.

`Get-EC2Snapshot`

다음 코드 예시는 `Get-EC2Snapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷을 설명합니다.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 'Name' 태그가 있는 스냅샷을 설명합니다.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

예제 3: 이 예제에서는 값이 'TestValue'인 'Name' 태그가 있는 스냅샷을 설명합니다.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

예제 4: 이 예제에서는 모든 스냅샷을 설명합니다.

```
Get-EC2Snapshot -Owner self
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSnapshots을 참조하세요.

`Get-EC2SnapshotAttribute`

다음 코드 예시는 `Get-EC2SnapshotAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷의 지정된 속성을 설명합니다.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

출력:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
```

```
{}                          {}              snap-12345678
```

예제 2: 이 예제에서는 지정된 스냅샷의 지정된 속성을 설명합니다.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

출력:

```
Group    UserId
-----    ------
all
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSnapshotAttribute를 참조하세요.

`Get-EC2SpotDatafeedSubscription`

다음 코드 예시는 `Get-EC2SpotDatafeedSubscription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 스팟 인스턴스 데이터 피드를 설명합니다.

```
Get-EC2SpotDatafeedSubscription
```

출력:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotDatafeedSubscription을 참조하세요.

`Get-EC2SpotFleetInstance`

다음 코드 예시는 `Get-EC2SpotFleetInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 플릿 요청과 연결된 인스턴스를 설명합니다.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

출력:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotFleetInstances를 참조하세요.

`Get-EC2SpotFleetRequest`

다음 코드 예시는 `Get-EC2SpotFleetRequest`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 플릿 요청을 설명합니다.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

출력:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

예제 2: 이 예제에서는 모든 스팟 플릿 요청을 설명합니다.

```
Get-EC2SpotFleetRequest
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotFleetRequests를 참조하세요.

Get-EC2SpotFleetRequestHistory

다음 코드 예시는 Get-EC2SpotFleetRequestHistory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 플릿 요청의 기록을 설명합니다.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

출력:

```
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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotFleetRequestHistory를 참조하세요.

Get-EC2SpotInstanceRequest

다음 코드 예시는 Get-EC2SpotInstanceRequest의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 인스턴스 요청을 설명합니다.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 모든 스팟 인스턴스 요청을 설명합니다.

```
Get-EC2SpotInstanceRequest
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotInstanceRequests를 참조하세요.

`Get-EC2SpotPriceHistory`

다음 코드 예시는 `Get-EC2SpotPriceHistory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스 유형 및 가용 영역에 대한 스팟 가격 기록의 마지막 10개 항목을 가져옵니다. -AvailabilityZone 파라미터에 지정된 값은 cmdlet의 -Region 파라미터(예제에 표

시되지 않음)에 제공되거나 쉘의 기본값으로 설정된 리전 값에 대해 유효해야 합니다. 이 예제 명령은 환경에서 기본 리전인 'us-west-2'가 설정되었다고 가정합니다.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSpotPriceHistory를 참조하세요.

`Get-EC2Subnet`

다음 코드 예시는 `Get-EC2Subnet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 서브넷을 설명합니다.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

출력:

```
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
```

예제 2: 이 예제에서는 모든 서브넷을 설명합니다.

```
Get-EC2Subnet
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSubnets을 참조하세요.

Get-EC2Tag

다음 코드 예시는 Get-EC2Tag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 resource-type 'image'에 대한 태그를 가져옵니다.

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

출력:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

예제 2: 이 예제에서는 모든 리소스에 대한 모든 태그를 가져와 리소스 유형별로 그룹화합니다.

```
Get-EC2Tag | Group-Object resourcetype
```

출력:

```
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}
```

예제 3: 이 예제에서는 지정된 리전에 대해 태그가 'auto-delete'이고 값이 'no'인 모든 리소스를 표시합니다.

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

출력:

```
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
```

예제 4: 이 예제에서는 태그가 'auto-delete'이고 값이 'no'인 모든 리소스를 가져와서 다음 파이프에서 추가로 필터링하여 'instance' 리소스 유형만 구문 분석하고 최종적으로 값이 인스턴스 ID 자체인 각 인스턴스 리소스에 대해 'ThisInstance' 태그를 생성합니다.

```
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}}
```

예제 5: 이 예제에서는 모든 인스턴스 리소스와 'Name' 키에 대한 태그를 가져와 테이블 형식으로 표시합니다.

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

출력:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

Get-EC2Volume

다음 코드 예시는 Get-EC2Volume의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 EBS 볼륨을 설명합니다.

```
Get-EC2Volume -VolumeId vol-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 상태가 'available'인 EBS 볼륨을 설명합니다.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

출력:

```
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
...
```

예제 3: 이 예제에서는 모든 EBS 볼륨을 설명합니다.

```
Get-EC2Volume
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVolumes을 참조하세요.

`Get-EC2VolumeAttribute`

다음 코드 예시는 `Get-EC2VolumeAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨의 지정된 속성을 설명합니다.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

출력:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVolumeAttribute를 참조하세요.

Get-EC2VolumeStatus

다음 코드 예시는 Get-EC2VolumeStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨의 상태를 설명합니다.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

출력:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

출력:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

출력:

```
Name                           Status
----                           ------
io-enabled                     passed
io-performance                 not-applicable
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVolumeStatus를 참조하세요.

`Get-EC2Vpc`

다음 코드 예시는 `Get-EC2Vpc`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC를 설명합니다.

```
Get-EC2Vpc -VpcId vpc-12345678
```

출력:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

예제 2: 이 예제에서는 기본 VPC를 설명합니다(리전당 VPC 하나만 있을 수 있음). 계정이 이 리전에서 EC2-Classic을 지원하는 경우 기본 VPC는 존재하지 않습니다.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

출력:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
```

```
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

예제 3: 이 예제에서는 지정된 필터와 일치하는 VPC를 설명합니다(즉, 값이 '10.0.0.0/16'이고 'available' 상태인 CIDR이 있음).

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

예제 4: 이 예제에서는 모든 VPC를 설명합니다.

```
Get-EC2Vpc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcs를 참조하세요.

`Get-EC2VpcAttribute`

다음 코드 예시는 `Get-EC2VpcAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'enableDnsSupport' 속성을 설명합니다.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

출력:

```
EnableDnsSupport
----------------
True
```

예제 2: 이 예제에서는 'enableDnsHostnames' 속성을 설명합니다.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

출력:

```
EnableDnsHostnames
```

```
------------------
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcAttribute를 참조하세요.

`Get-EC2VpcClassicLink`

다음 코드 예시는 `Get-EC2VpcClassicLink`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 위 예제에서는 리전에 대해 ClassicLinkEnabled 상태의 모든 VPC를 반환합니다.

```
Get-EC2VpcClassicLink -Region eu-west-1
```

출력:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcClassicLink를 참조하세요.

`Get-EC2VpcClassicLinkDnsSupport`

다음 코드 예시는 `Get-EC2VpcClassicLinkDnsSupport`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 eu-west-1 리전에 대한 VPC의 ClassicLink DNS 지원 상태를 설명합니다.

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

출력:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcClassicLinkDnsSupport를 참조하세요.

`Get-EC2VpcEndpoint`

다음 코드 예시는 `Get-EC2VpcEndpoint`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 eu-west-1 리전에 대한 하나 이상의 VPC 엔드포인트를 설명합니다. 그런 다음 출력을 다음 명령으로 파이프하여 VpcEndpointId 속성을 선택하고 배열 VPC ID를 문자열 배열로 반환합니다.

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

출력:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

예제 2: 이 예제에서는 eu-west-1 리전의 모든 VPC 엔드포인트를 설명하고 VpcEndpointId, VpcId, ServiceName, PrivateDnsEnabled 속성을 선택하여 테이블 형식으로 표시합니다.

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

출력:

```
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
```

예제 3: 이 예제에서는 VPC 엔드포인트 vpce-01a2ab3f4f5cc6f7d에 대한 정책 문서를 json 파일로 내보냅니다.

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcEndpoints를 참조하세요.

`Get-EC2VpcEndpointService`

다음 코드 예시는 `Get-EC2VpcEndpointService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 필터를 사용하여 EC2 VPC 엔드포인트 서비스를 설명합니다. 이 경우 필터는 com.amazonaws.eu-west-1.ecs입니다. 그리고 ServiceDetails 속성을 확장하고 세부 정보를 표시합니다.

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

출력:

```
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
```

예제 2: 이 예제에서는 모든 EC2 VPC 엔드포인트 서비스를 검색하고 'ssm'과 일치하는 ServiceName을 반환합니다.

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

출력:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpcEndpointServices를 참조하세요.

`Get-EC2VpnConnection`

다음 코드 예시는 `Get-EC2VpnConnection`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPN 연결을 설명합니다.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

출력:

```
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
```

예제 2: 이 예제에서는 상태가 pending 또는 available인 모든 VPN 연결을 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

예제 3: 이 예제에서는 모든 VPN 연결을 설명합니다.

```
Get-EC2VpnConnection
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpnConnections을 참조하세요.

`Get-EC2VpnGateway`

다음 코드 예시는 `Get-EC2VpnGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이를 설명합니다.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

출력:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

예제 2: 이 예제에서는 상태가 pending 또는 available인 모든 가상 프라이빗 게이트웨이를 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )
```

```
Get-EC2VpnGateway -Filter $filter
```

예제 3: 이 예제에서는 모든 가상 프라이빗 게이트웨이를 설명합니다.

```
Get-EC2VpnGateway
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeVpnGateways를 참조하세요.

`Grant-EC2SecurityGroupEgress`

다음 코드 예시는 `Grant-EC2SecurityGroupEgress`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-VPC에 지정된 보안 그룹에 대한 송신 규칙을 정의합니다. 이 규칙은 지정된 IP 주소 범위에 대해 TCP 포트 80 액세스 권한을 부여합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

예제 2: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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
```

예제 3: 이 예제에서는 지정된 소스 보안 그룹에 대해 TCP 포트 80 액세스 권한을 부여합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AuthorizeSecurityGroupEgress를 참조하세요.

`Grant-EC2SecurityGroupIngress`

다음 코드 예시는 `Grant-EC2SecurityGroupIngress`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-VPC의 보안 그룹에 대한 수신 규칙을 정의합니다. 이러한 규칙은 특정 IP 주소에 대해 SSH(포트 22) 및 RDC(포트 3389) 액세스 권한을 부여합니다. 보안 그룹 이름이 아닌 보안 그룹 ID를 사용하여 EC2-VPC의 보안 그룹을 식별해야 한다는 점에 유의하세요. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$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 )
```

예제 2: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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 )
```

예제 3: 이 예제에서는 EC2-Classic의 보안 그룹에 대한 수신 규칙을 정의합니다. 이러한 규칙은 특정 IP 주소에 대해 SSH(포트 22) 및 RDC(포트 3389) 액세스 권한을 부여합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$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 )
```

예제 4: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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 )
```

예제 5: 이 예제에서는 지정된 소스 보안 그룹(sg-1a2b3c4d)에서 지정된 보안 그룹(sg-12345678)으로의 TCP 포트 8081 액세스 권한을 부여합니다.

```
$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 } )
```

예제 6: 이 예제에서는 설명이 포함된 TCP 포트 22 트래픽에 대한 보안 그룹 sg-1234abcd의 수신 규칙에 CIDR 5.5.5.5/32를 추가합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AuthorizeSecurityGroupIngress를 참조하세요.

`Import-EC2Image`

다음 코드 예시는 `Import-EC2Image`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 멱등성 토큰을 사용하여 지정된 Amazon S3 버킷에서 Amazon EC2로 단일 디스크 가상 머신 이미지를 가져옵니다. 이 예제에서는 기본 이름이 'vmimport'인 VM Import 서비스 역할이 있어야 하며, VM Import 사전 조건 주제에 설명된 대로 지정된 버킷에 대한 Amazon EC2 액세스를 허용하는 정책이 있어야 합니다. 사용자 지정 역할을 사용하려면 **`-RoleName`** 파라미터를 사용하여 역할 이름을 지정합니다.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

출력:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
```

```
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ImportImage를 참조하세요.

`Import-EC2KeyPair`

다음 코드 예시는 `Import-EC2KeyPair`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 퍼블릭 키를 EC2로 가져옵니다. 첫 번째 줄은 퍼블릭 키 파일(*.pub)의 내용을 변수 **$publickey**에 저장합니다. 다음 예제에서는 퍼블릭 키 파일의 UTF8 형식을 Base64-encoded 문자열로 변환하고, 변환된 문자열을 **$pkbase64** 변수에 저장합니다. 마지막 줄에서는 변환된 퍼블릭 키를 EC2로 가져옵니다. cmdlet은 키 지문과 이름을 결과로 반환합니다.

```
$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
```

출력:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ImportKeyPair를 참조하세요.

`Import-EC2Snapshot`

다음 코드 예시는 `Import-EC2Snapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'VMDK' 형식의 VM 디스크 이미지를 Amazon EBS 스냅샷으로 가져옵니다. 이 예제에서는 http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/

VMImportPrerequisites.html 페이지의 **VM Import Prequisites** 주제에 설명된 대로 지정된 버킷에 대한 Amazon EC2 액세스를 허용하는 정책과 함께 기본 이름이 'vmimport'인 VM Import 서비스 역할이 필요합니다. 사용자 지정 역할을 사용하려면 **-RoleName** 파라미터를 사용하여 역할 이름을 지정합니다.

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

출력:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ImportSnapshot을 참조하세요.

Move-EC2AddressToVpc

다음 코드 예시는 Move-EC2AddressToVpc의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 퍼블릭 IP 주소가 12.345.67.89인 EC2 인스턴스를 미국 동부(버지니아 북부) 리전의 EC2-VPC 플랫폼으로 이동합니다.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

예제 2: 이 예제에서는 Get-EC2Instance 명령의 결과를 Move-EC2AddressToVpc cmdlet으로 파이프합니다. Get-EC2Instance 명령은 인스턴스 ID로 지정된 인스턴스를 가져온 다음 인스턴스의 퍼블릭 IP 주소 속성을 반환합니다.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 MoveAddressToVpc를 참조하세요.

New-EC2Address

다음 코드 예시는 New-EC2Address의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC의 인스턴스에 사용할 탄력적 IP 주소를 할당합니다.

```
New-EC2Address -Domain Vpc
```

출력:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

예제 2: 이 예제에서는 EC2-Classic의 인스턴스에 사용할 탄력적 IP 주소를 할당합니다.

```
New-EC2Address
```

출력:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AllocateAddress를 참조하세요.

New-EC2CustomerGateway

다음 코드 예시는 New-EC2CustomerGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 고객 게이트웨이를 생성합니다.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

출력:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCustomerGateway를 참조하세요.

New-EC2DhcpOption

다음 코드 예시는 New-EC2DhcpOption의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 DHCP 옵션 세트를 생성합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

출력:

```
DhcpConfigurations                  DhcpOptionsId    Tags
------------------                  -------------    ----
{domain-name, domain-name-servers}  dopt-1a2b3c4d    {}
```

예제 2: PowerShell 버전 2에서 각 DHCP 옵션을 만들려면 New-Object를 사용해야 합니다.

```
$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)
```

출력:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDhcpOptions을 참조하세요.

New-EC2FlowLog

다음 코드 예시는 New-EC2FlowLog의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'Admin' 역할의 권한을 사용하여 모든 'REJECT' 트래픽에 대해 'subnet1-log'라는 cloud-watch-log에 subnet-1d234567 서브넷에 대한 EC2 플로우로그를 생성합니다.

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

출력:

```
ClientToken                                    FlowLogIds             Unsuccessful
-----------                                    ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFlowLogs를 참조하세요.

New-EC2Host

다음 코드 예시는 New-EC2Host의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스 유형 및 가용 영역에 대해 계정에 전용 호스트를 할당합니다.

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

출력:

```
h-01e23f4cd567890f3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AllocateHosts를 참조하세요.

New-EC2HostReservation

다음 코드 예시는 New-EC2HostReservation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 전용 호스트 h-01e23f4cd567890f1의 구성과 일치하는 구성으로 hro-0c1f23456789d0ab를 제공하는 예약을 구매합니다.

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

출력:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PurchaseHostReservation을 참조하세요.

New-EC2Image

다음 코드 예시는 New-EC2Image의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에서 지정된 이름 및 설명으로 AMI를 생성합니다. Amazon EC2는 이미지를 생성하기 전에 인스턴스를 완전히 종료하려고 시도하고 완료 시 인스턴스를 다시 시작합니다.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

예제 2: 이 예제에서는 지정된 인스턴스에서 지정된 이름 및 설명으로 AMI를 생성합니다. Amazon EC2는 인스턴스를 종료했다가 다시 시작하지 않고 이미지를 생성합니다. 따라서 생성된 이미지의 파일 시스템 무결성을 보장할 수 없습니다.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

예제 3: 이 예제에서는 볼륨이 3개인 AMI를 생성합니다. 첫 번째 볼륨은 Amazon EBS 스냅샷을 기반으로 합니다. 두 번째 볼륨은 빈 100GiB Amazon EBS 볼륨입니다. 세 번째 볼륨은 인스턴스 저장소 볼륨입니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateImage를 참조하세요.

`New-EC2Instance`

다음 코드 예시는 `New-EC2Instance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-Classic 또는 기본 VPC에서 지정된 AMI의 단일 인스턴스를 시작합니다.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

예제 2: 이 예제에서는 VPC에서 지정된 AMI의 단일 인스턴스를 시작합니다.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

예제 3: EBS 볼륨 또는 인스턴스 저장소 볼륨을 추가하려면 블록 디바이스 매핑을 정의하고 명령에 추가합니다. 이 예제에서는 인스턴스 저장소 볼륨을 추가합니다.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

예제 4: 현재 Windows AMI 중 하나를 지정하려면 Get-EC2ImageByName을 사용하여 AMI ID를 가져옵니다. 이 예제에서는 Windows Server 2016용 최신 기본 AMI를 기반으로 인스턴스를 시작합니다.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

예제 5: 지정된 전용 호스트 환경에서 인스턴스를 시작합니다.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

예제 6: 이 요청에서는 두 개의 인스턴스를 시작하고 키가 webserver이고 값이 production인 태그를 인스턴스에 적용합니다. 또한 이 요청은 생성된 볼륨(이 경우 각 인스턴스의 루트 볼륨)에 키가 cost-center이고 값이 cc123인 태그를 적용합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RunInstances를 참조하세요.

`New-EC2InstanceExportTask`

다음 코드 예시는 `New-EC2InstanceExportTask`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 중지된 인스턴스인 **`i-0800b00a00EXAMPLE`**을 가상 하드 디스크(VHD) 형태로 내보내 S3 버킷 **`testbucket-export-instances-2019`**에 저장합니다. 대상 환경은 이고**`Microsoft`**, 인스턴스가 리전에 **`us-east-1`** 있는 반면 사용자의 기본 리전은 us-east-1 이 아니기 때문에 AWS 리전 파라미터가 추가됩니다. 내보내기 작업의 상태를 가져오려면 이 명령의 결과에서 **`ExportTaskId`** 값을 복사한 다음 **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`**를 실행합니다.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket" -
TargetEnvironment Microsoft -Region us-east-1
```

출력:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateInstanceExportTask를 참조하세요.

New-EC2InternetGateway

다음 코드 예시는 New-EC2InternetGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인터넷 게이트웨이를 생성합니다.

```
New-EC2InternetGateway
```

출력:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateInternetGateway를 참조하세요.

New-EC2KeyPair

다음 코드 예시는 New-EC2KeyPair의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 키 페어를 생성하고 지정된 이름의 파일에 PEM 인코딩 RSA 프라이빗 키를 캡처합니다. PowerShell을 사용하는 경우 유효한 키를 생성하려면 인코딩을 ascii로 설정해야 합니다. 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 Amazon EC2 키 페어 생성, 표시 및 삭제(https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html)를 참조하세요.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateKeyPair를 참조하세요.

New-EC2NetworkAcl

다음 코드 예시는 New-EC2NetworkAcl의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC에 대한 네트워크 ACL을 만듭니다.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

출력:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateNetworkAcl을 참조하세요.

New-EC2NetworkAclEntry

다음 코드 예시는 New-EC2NetworkAclEntry의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL에 대한 항목을 만듭니다. 이 규칙은 UDP 포트 53(DNS)을 통해 모든 곳(0.0.0.0/0)에서 연결된 모든 서브넷으로 유입되는 트래픽을 허용합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateNetworkAclEntry를 참조하세요.

New-EC2NetworkInterface

다음 코드 예시는 New-EC2NetworkInterface의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스를 생성합니다.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateNetworkInterface를 참조하세요.

New-EC2PlacementGroup

다음 코드 예시는 New-EC2PlacementGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름으로 배치 그룹을 생성합니다.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreatePlacementGroup을 참조하세요.

New-EC2Route

다음 코드 예시는 New-EC2Route의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블에 대한 지정된 경로를 생성합니다. 이 경로는 모든 트래픽을 일치시켜 지정된 인터넷 게이트웨이로 보냅니다.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId
 igw-1a2b3c4d
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRoute를 참조하세요.

New-EC2RouteTable

다음 코드 예시는 New-EC2RouteTable의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC에 대한 라우팅 테이블을 생성합니다.

```
New-EC2RouteTable -VpcId vpc-12345678
```

출력:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRouteTable을 참조하세요.

New-EC2ScheduledInstance

다음 코드 예시는 New-EC2ScheduledInstance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 일정이 예약된 지정된 인스턴스를 시작합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RunScheduledInstances를 참조하세요.

New-EC2ScheduledInstancePurchase

다음 코드 예시는 New-EC2ScheduledInstancePurchase의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 일정이 예약된 인스턴스를 구매합니다.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PurchaseScheduledInstances를 참조하세요.

New-EC2SecurityGroup

다음 코드 예시는 New-EC2SecurityGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC에 대한 보안 그룹을 생성합니다.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

출력:

```
sg-12345678
```

예제 2: 이 예제에서는 EC2-Classic에 대한 보안 그룹을 생성합니다.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

출력:

```
sg-45678901
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSecurityGroup을 참조하세요.

New-EC2Snapshot

다음 코드 예시는 New-EC2Snapshot의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨의 스냅샷을 생성합니다.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSnapshot을 참조하세요.

New-EC2SpotDatafeedSubscription

다음 코드 예시는 New-EC2SpotDatafeedSubscription의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 스팟 인스턴스 데이터 피드를 생성합니다.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

출력:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
```

```
State   : Active
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSpotDatafeedSubscription을 참조하세요.

New-EC2Subnet

다음 코드 예시는 New-EC2Subnet의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 CIDR로 서브넷을 생성합니다.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSubnet을 참조하세요.

New-EC2Tag

다음 코드 예시는 New-EC2Tag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스에 단일 태그를 추가합니다. 태그 키는 'myTag '이고 태그 값은 'myTagValue'입니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

예제 2: 이 예제에서는 지정된 리소스에 대해 지정된 태그를 업데이트하거나 추가합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

예제 3: PowerShell 버전 2에서 Tag 파라미터에 대해 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTags를 참조하세요.

New-EC2Volume

다음 코드 예시는 New-EC2Volume의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨을 생성합니다.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

출력:

```
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
```

예제 2: 이 예제 요청에서는 볼륨을 생성하고 키가 stack이고 값이 production인 태그를 적용합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVolume을 참조하세요.

New-EC2Vpc

다음 코드 예시는 New-EC2Vpc의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 CIDR로 VPC를 생성합니다. Amazon VPC는 또한 VPC에 대해 기본 DHCP 옵션 세트, 기본 라우팅 테이블, 기본 네트워크 ACL을 생성합니다.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

출력:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVpc를 참조하세요.

New-EC2VpcEndpoint

다음 코드 예시는 New-EC2VpcEndpoint의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC vpc-0fc1ff23f45b678eb의 서비스 com.amazonaws.eu-west-1.s3에 대한 새 VPC 엔드포인트를 생성합니다.

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

출력:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVpcEndpoint를 참조하세요.

New-EC2VpnConnection

다음 코드 예시는 New-EC2VpnConnection의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이와 지정된 고객 게이트웨이 간에 VPN 연결을 만듭니다. 출력에는 네트워크 관리자에게 필요한 구성 정보가 XML 형식으로 포함됩니다.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

출력:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
```

```
VpnGatewayId                : vgw-1a2b3c4d
```

예제 2: 이 예제에서는 VPN 연결을 생성하고 지정된 이름의 파일에 구성을 캡처합니다.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

예제 3: 이 예제에서는 지정된 가상 프라이빗 게이트웨이와 지정된 고객 게이트웨이 간에 정적 라우팅을 사용하여 VPN 연결을 만듭니다.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVpnConnection을 참조하세요.

New-EC2VpnConnectionRoute

다음 코드 예시는 New-EC2VpnConnectionRoute의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPN 연결에 대해 지정된 정적 경로를 만듭니다.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVpnConnectionRoute를 참조하세요.

New-EC2VpnGateway

다음 코드 예시는 New-EC2VpnGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이를 생성합니다.

```
New-EC2VpnGateway -Type ipsec.1
```

출력:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVpnGateway를 참조하세요.

Register-EC2Address

다음 코드 예시는 Register-EC2Address의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 탄력적 IP 주소를 VPC의 지정된 인스턴스와 연결합니다.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

출력:

```
eipassoc-12345678
```

예제 2: 이 예제에서는 지정된 탄력적 IP 주소를 EC2-Classic의 지정된 인스턴스와 연결합니다.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateAddress를 참조하세요.

Register-EC2DhcpOption

다음 코드 예시는 Register-EC2DhcpOption의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 DHCP 옵션 세트를 지정된 VPC와 연결합니다.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

예제 2: 이 예제에서는 기본 DHCP 옵션 세트를 지정된 VPC와 연결합니다.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateDhcpOptions을 참조하세요.

Register-EC2Image

다음 코드 예시는 Register-EC2Image의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Amazon S3에 있는 지정된 매니페스트 파일을 사용하여 AMI를 등록합니다.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterImage를 참조하세요.

Register-EC2PrivateIpAddress

다음 코드 예시는 Register-EC2PrivateIpAddress의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 보조 프라이빗 IP 주소를 지정된 네트워크 인터페이스에 할당합니다.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

예제 2: 이 예제에서는 두 개의 보조 프라이빗 IP 주소를 생성하고 이를 지정된 네트워크 인터페이스에 할당합니다.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssignPrivateIpAddresses를 참조하세요.

`Register-EC2RouteTable`

다음 코드 예시는 `Register-EC2RouteTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블을 지정된 서브넷과 연결합니다.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

출력:

```
rtbassoc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateRouteTable을 참조하세요.

`Remove-EC2Address`

다음 코드 예시는 `Remove-EC2Address`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC의 인스턴스에 대해 지정된 탄력적 IP 주소를 릴리스합니다.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

예제 2: 이 예제에서는 EC2-Classic의 인스턴스에 대해 지정된 탄력적 IP 주소를 릴리스합니다.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReleaseAddress를 참조하세요.

`Remove-EC2CapacityReservation`

다음 코드 예시는 `Remove-EC2CapacityReservation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 용량 예약 cr-0c1f2345db6f7cdba를 취소합니다.

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelCapacityReservation을 참조하세요.

`Remove-EC2CustomerGateway`

다음 코드 예시는 `Remove-EC2CustomerGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 고객 게이트웨이를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCustomerGateway를 참조하세요.

Remove-EC2DhcpOption

다음 코드 예시는 Remove-EC2DhcpOption의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 DHCP 옵션 세트를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDhcpOptions을 참조하세요.

Remove-EC2FlowLog

다음 코드 예시는 Remove-EC2FlowLog의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 FlowLogId fl-01a2b3456a789c01을 제거합니다.

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFlowLogs를 참조하세요.

Remove-EC2Host

다음 코드 예시는 Remove-EC2Host의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 호스트 ID h-0badafd1dcb2f3456을 릴리스합니다.

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReleaseHosts를 참조하세요.

Remove-EC2Instance

다음 코드 예시는 Remove-EC2Instance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 종료합니다(인스턴스가 실행 중이거나 'stopped' 상태일 수 있음). 계속하기 전에 cmdlet에 확인 프롬프트가 표시됩니다. 프롬프트를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-EC2Instance -InstanceId i-12345678
```

출력:

```
CurrentState                        InstanceId    PreviousState
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TerminateInstances를 참조하세요.

`Remove-EC2InternetGateway`

다음 코드 예시는 `Remove-EC2InternetGateway`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인터넷 게이트웨이를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteInternetGateway를 참조하세요.

`Remove-EC2KeyPair`

다음 코드 예시는 `Remove-EC2KeyPair`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 키 페어를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteKeyPair를 참조하세요.

`Remove-EC2NetworkAcl`

다음 코드 예시는 `Remove-EC2NetworkAcl`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteNetworkAcl을 참조하세요.

`Remove-EC2NetworkAclEntry`

다음 코드 예시는 `Remove-EC2NetworkAclEntry`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL에서 지정된 규칙을 제거합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteNetworkAclEntry를 참조하세요.

`Remove-EC2NetworkInterface`

다음 코드 예시는 `Remove-EC2NetworkInterface`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteNetworkInterface를 참조하세요.

`Remove-EC2PlacementGroup`

다음 코드 예시는 `Remove-EC2PlacementGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 배치 그룹을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePlacementGroup을 참조하세요.

`Remove-EC2Route`

다음 코드 예시는 `Remove-EC2Route`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블에서 지정된 경로를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRoute를 참조하세요.

Remove-EC2RouteTable

다음 코드 예시는 Remove-EC2RouteTable의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRouteTable을 참조하세요.

Remove-EC2SecurityGroup

다음 코드 예시는 Remove-EC2SecurityGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-VPC에 지정된 보안 그룹을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

출력:

```
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"):
```

예제 2: 이 예제에서는 EC2-Classic에 지정된 보안 그룹을 삭제합니다.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSecurityGroup을 참조하세요.

`Remove-EC2Snapshot`

다음 코드 예시는 `Remove-EC2Snapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSnapshot을 참조하세요.

`Remove-EC2SpotDatafeedSubscription`

다음 코드 예시는 `Remove-EC2SpotDatafeedSubscription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 스팟 인스턴스 데이터 피드를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2SpotDatafeedSubscription
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSpotDatafeedSubscription을 참조하세요.

`Remove-EC2Subnet`

다음 코드 예시는 `Remove-EC2Subnet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 서브넷을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSubnet을 참조하세요.

`Remove-EC2Tag`

다음 코드 예시는 `Remove-EC2Tag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 태그 값에 관계없이 지정된 리소스에서 지정된 태그를 삭제합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

예제 2: 이 예제에서는 태그 값이 일치하는 경우에만 지정된 리소스에서 지정된 태그를 삭제합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

예제 3: 이 예제에서는 태그 값에 관계없이 지정된 리소스에서 지정된 태그를 삭제합니다.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

예제 4: 이 예제에서는 태그 값이 일치하는 경우에만 지정된 리소스에서 지정된 태그를 삭제합니다.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTags를 참조하세요.

`Remove-EC2Volume`

다음 코드 예시는 `Remove-EC2Volume`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 볼륨을 분리합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2Volume -VolumeId vol-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVolume을 참조하세요.

`Remove-EC2Vpc`

다음 코드 예시는 `Remove-EC2Vpc`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVpc를 참조하세요.

`Remove-EC2VpnConnection`

다음 코드 예시는 `Remove-EC2VpnConnection`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPN 연결을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVpnConnection을 참조하세요.

`Remove-EC2VpnConnectionRoute`

다음 코드 예시는 `Remove-EC2VpnConnectionRoute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPN 연결에서 지정된 정적 경로를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVpnConnectionRoute를 참조하세요.

Remove-EC2VpnGateway

다음 코드 예시는 Remove-EC2VpnGateway의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가상 프라이빗 게이트웨이를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVpnGateway를 참조하세요.

Request-EC2SpotFleet

다음 코드 예시는 Request-EC2SpotFleet의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스 유형에 대해 최저 가격으로 가용 영역에 스팟 플릿 요청을 생성합니다. 계정이 EC2-VPC만 지원하는 경우, 스팟 플릿은 기본 서브넷이 있는 최저 가격의 가용 영역에서 인스턴스를 시작합니다. 계정이 EC2-Classic을 지원하는 경우, 스팟 플릿은 최저 가격의 가용 영역에 있는 EC2-Classic에서 인스턴스를 시작합니다. 지불하는 가격은 요청에 지정된 스팟 가격을 초과하지 않습니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RequestSpotFleet을 참조하세요.

`Request-EC2SpotInstance`

다음 코드 예시는 `Request-EC2SpotInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 서브넷에서 일회성 스팟 인스턴스를 요청합니다. 보안 그룹은 지정된 서브넷이 포함된 VPC에 대해 생성해야 하며, 네트워크 인터페이스를 사용할 때는 보안 그룹을 ID로 지정해야 합니다. 네트워크 인터페이스를 지정할 때는 해당 네트워크 인터페이스를 사용하는 서브넷 ID를 포함해야 합니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RequestSpotInstances를 참조하세요.

`Reset-EC2ImageAttribute`

다음 코드 예시는 `Reset-EC2ImageAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'launchPermission' 속성을 기본값으로 재설정합니다. 기본적으로 AMI는 프라이빗으로 설정됩니다.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResetImageAttribute를 참조하세요.

`Reset-EC2InstanceAttribute`

다음 코드 예시는 `Reset-EC2InstanceAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에 대한 'sriovNetSupport' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

예제 2: 이 예제에서는 지정된 인스턴스에 대한 'ebsOptimized' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

예제 3: 이 예제에서는 지정된 인스턴스에 대한 'sourceDestCheck' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

예제 4: 이 예제에서는 지정된 인스턴스에 대한 'disableApiTermination' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

예제 5: 이 예제에서는 지정된 인스턴스의 'instanceInitiatedShutdownBehavior' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResetInstanceAttribute를 참조하세요.

`Reset-EC2NetworkInterfaceAttribute`

다음 코드 예시는 `Reset-EC2NetworkInterfaceAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스에 대한 소스/대상 확인을 재설정합니다.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResetNetworkInterfaceAttribute를 참조하세요.

`Reset-EC2SnapshotAttribute`

다음 코드 예시는 `Reset-EC2SnapshotAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스냅샷의 지정된 속성을 재설정합니다.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResetSnapshotAttribute를 참조하세요.

`Restart-EC2Instance`

다음 코드 예시는 `Restart-EC2Instance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 재부팅합니다.

```
Restart-EC2Instance -InstanceId i-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RebootInstances를 참조하세요.

`Revoke-EC2SecurityGroupEgress`

다음 코드 예시는 `Revoke-EC2SecurityGroupEgress`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-VPC에 지정된 보안 그룹에 대한 규칙을 제거합니다. 이렇게 하면 지정된 IP 주소 범위에 대한 TCP 포트 80 액세스 권한이 취소됩니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

예제 2: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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
```

예제 3: 이 예제에서는 지정된 소스 보안 그룹에 대한 TCP 포트 80 액세스 권한을 취소합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RevokeSecurityGroupEgress를 참조하세요.

`Revoke-EC2SecurityGroupIngress`

다음 코드 예시는 `Revoke-EC2SecurityGroupIngress`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 EC2-VPC의 지정된 보안 그룹에 대해 지정된 주소 범위에서 TCP 포트 22에 대한 액세스 권한을 취소합니다. 보안 그룹 이름이 아닌 보안 그룹 ID를 사용하여 EC2-VPC의 보안 그룹을 식별해야 한다는 점에 유의하세요. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

예제 2: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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
```

예제 3: 이 예제에서는 EC2-Classic의 지정된 보안 그룹에 대해 지정된 주소 범위에서 TCP 포트 22에 대한 액세스 권한을 취소합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

예제 4: PowerShell 버전 2에서 IpPermission 객체를 생성하려면 New-Object를 사용해야 합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RevokeSecurityGroupIngress를 참조하세요.

Send-EC2InstanceStatus

다음 코드 예시는 Send-EC2InstanceStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에 대한 상태 피드백을 보고합니다.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReportInstanceStatus를 참조하세요.

Set-EC2NetworkAclAssociation

다음 코드 예시는 Set-EC2NetworkAclAssociation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL을 지정된 네트워크 ACL 연결의 서브넷과 연결합니다.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

출력:

```
aclassoc-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReplaceNetworkAclAssociation을 참조하세요.

`Set-EC2NetworkAclEntry`

다음 코드 예시는 `Set-EC2NetworkAclEntry`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 ACL에 지정된 항목을 대체합니다. 새 규칙은 지정된 주소에서 연결된 서브넷으로의 인바운드 트래픽을 허용합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReplaceNetworkAclEntry를 참조하세요.

`Set-EC2Route`

다음 코드 예시는 `Set-EC2Route`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블의 지정된 경로를 대체합니다. 새 경로는 지정된 트래픽을 지정된 가상 프라이빗 게이트웨이로 보냅니다.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReplaceRoute를 참조하세요.

`Set-EC2RouteTableAssociation`

다음 코드 예시는 `Set-EC2RouteTableAssociation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 라우팅 테이블을 지정된 라우팅 테이블 연결을 위한 서브넷과 연결합니다.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

출력:

```
rtbassoc-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReplaceRouteTableAssociation을 참조하세요.

`Start-EC2Instance`

다음 코드 예시는 `Start-EC2Instance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 시작합니다.

```
Start-EC2Instance -InstanceId i-12345678
```

출력:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

예제 2: 이 예제에서는 지정된 인스턴스를 시작합니다.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

예제 3: 이 예제에서는 현재 중지된 인스턴스 세트를 시작합니다. Get-EC2Instance에서 반환하는 인스턴스 객체는 Start-EC2Instance로 파이프됩니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

예제 4: PowerShell 버전 2에서 Filter 파라미터에 대해 필터를 생성하려면 New-Object를 사용해야 합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartInstances를 참조하세요.

`Start-EC2InstanceMonitoring`

다음 코드 예시는 `Start-EC2InstanceMonitoring`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에 대한 세부 모니터링을 활성화합니다.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

출력:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 MonitorInstances를 참조하세요.

`Stop-EC2ImportTask`

다음 코드 예시는 `Stop-EC2ImportTask`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 가져오기 작업(스냅샷 또는 이미지 가져오기)을 취소합니다. 필요한 경우 **`-CancelReason`** 파라미터를 사용하여 이유를 제공할 수 있습니다.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelImportTask를 참조하세요.

Stop-EC2Instance

다음 코드 예시는 Stop-EC2Instance의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스를 중지합니다.

```
Stop-EC2Instance -InstanceId i-12345678
```

출력:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopInstances를 참조하세요.

Stop-EC2InstanceMonitoring

다음 코드 예시는 Stop-EC2InstanceMonitoring의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 인스턴스에 대한 세부 모니터링을 비활성화합니다.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

출력:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UnmonitorInstances를 참조하세요.

`Stop-EC2SpotFleetRequest`

다음 코드 예시는 `Stop-EC2SpotFleetRequest`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 스팟 플릿 요청을 취소하고 연결된 스팟 인스턴스를 종료합니다.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

예제 2: 이 예제에서는 연결된 스팟 인스턴스를 종료하지 않고 지정된 스팟 플릿 요청을 취소합니다.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelSpotFleetRequests를 참조하세요.

`Stop-EC2SpotInstanceRequest`

다음 코드 예시는 `Stop-EC2SpotInstanceRequest`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 스팟 인스턴스 요청을 취소합니다.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

출력:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelSpotInstanceRequests를 참조하세요.

`Unregister-EC2Address`

다음 코드 예시는 `Unregister-EC2Address`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC의 지정된 인스턴스에서 지정된 탄력적 IP 주소의 연결을 해제합니다.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

예제 2: 이 예제에서는 EC2-Classic의 지정된 인스턴스에서 지정된 탄력적 IP 주소의 연결을 해제합니다.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisassociateAddress를 참조하세요.

`Unregister-EC2Image`

다음 코드 예시는 `Unregister-EC2Image`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 AMI의 등록을 취소합니다.

```
Unregister-EC2Image -ImageId ami-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterImage를 참조하세요.

`Unregister-EC2PrivateIpAddress`

다음 코드 예시는 `Unregister-EC2PrivateIpAddress`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 네트워크 인터페이스에서 지정된 프라이빗 IP 주소의 할당을 해제합니다.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UnassignPrivateIpAddresses를 참조하세요.

`Unregister-EC2RouteTable`

다음 코드 예시는 `Unregister-EC2RouteTable`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 라우팅 테이블과 서브넷 간의 지정된 연결을 제거합니다.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisassociateRouteTable을 참조하세요.

`Update-EC2SecurityGroupRuleIngressDescription`

다음 코드 예시는 `Update-EC2SecurityGroupRuleIngressDescription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 기존 수신(인바운드) 보안 그룹 규칙에 대한 설명을 업데이트합니다.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}
```

```
Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

예제 2: 요청에서 파라미터를 생략하여 기존 수신(인바운드) 보안 그룹 규칙에 대한 설명을 제거합니다.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithoutDescription
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateSecurityGroupRuleDescriptionsIngress를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon ECR 예제

다음 코드 예제에서는 Amazon ECR에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-ECRLoginCommand`

다음 코드 예시는 `Get-ECRLoginCommand`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: IAM 위탁자에게 액세스 권한이 있는 Amazon ECR 레지스트리에 인증하는 데 사용할 수 있는 로그인 정보가 포함된 PSObject를 반환합니다. 직접 호출이 권한 부여 토큰을 얻는 데 필요한 자격 증명 및 리전 엔드포인트는 쉘 기본값(**Set-AWSCredential/Set-DefaultAWSRegion** 또는 **Initialize-AWSDefaultConfiguration** cmdlet에 의해 설정됨)에서 가져옵니다. Invoke-Expression과 함께 Command 속성을 사용하여 지정된 레지스트리에 로그인하거나 로그인이 필요한 다른 도구에서 반환된 자격 증명을 사용할 수 있습니다.

```
Get-ECRLoginCommand
```

출력:

```
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
```

예제 2: Docker 로그인 명령에 대한 입력으로 사용하는 로그인 정보가 포함된 PSObject를 검색합니다. IAM 위탁자에 해당 레지스트리에 대한 액세스 권한이 있으면 인증할 Amazon ECR 레지스트리 URI를 지정할 수 있습니다.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Get-ECRLoginCommand를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon ECS 예제

다음 코드 예제에서는 Amazon ECS에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-ECSClusterDetail`

다음 코드 예시는 `Get-ECSClusterDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 하나 이상의 ECS 클러스터를 설명합니다.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

출력:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeClusters를 참조하세요.

`Get-ECSClusterList`

다음 코드 예시는 `Get-ECSClusterList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 기존 ECS 클러스터 목록을 반환합니다.

```
Get-ECSClusterList
```

출력:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListClusters를 참조하세요.

`Get-ECSClusterService`

다음 코드 예시는 `Get-ECSClusterService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기본 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
Get-ECSClusterService
```

예제 2: 이 예제에서는 지정된 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
Get-ECSClusterService -Cluster myCluster
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListServices를 참조하세요.

`Get-ECSService`

다음 코드 예시는 `Get-ECSService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기본 클러스터에서 지정된 서비스의 세부 정보를 검색하는 방법을 보여줍니다.

```
Get-ECSService -Service my-hhtp-service
```

예제 2: 이 예제에서는 명명된 클러스터에서 실행되는 지정된 서비스의 세부 정보를 검색하는 방법을 보여줍니다.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeServices를 참조하세요.

New-ECSCluster

다음 코드 예시는 New-ECSCluster의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 새 Amazon ECS 클러스터를 생성합니다.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCluster를 참조하세요.

New-ECSService

다음 코드 예시는 New-ECSService의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제 명령에서는 `ecs-simple-service`라는 서비스를 기본 클러스터에 생성합니다. 이 서비스는 `ecs-demo` 태스크 정의를 사용하며 태스크의 인스턴스화 10회를 유지 관리합니다.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

예제 2: 이 예제 명령에서는 `ecs-simple-service`라는 기본 클러스터의 로드 밸런서 뒤에 서비스를 생성합니다. 이 서비스는 `ecs-demo` 태스크 정의를 사용하며 태스크의 인스턴스화 10회를 유지 관리합니다.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateService를 참조하세요.

`Remove-ECSCluster`

다음 코드 예시는 `Remove-ECSCluster`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 지정된 ECS 클러스터를 삭제합니다. 삭제하기 전에 이 클러스터에서 모든 컨테이너 인스턴스의 등록을 취소해야 합니다.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCluster를 참조하세요.

`Remove-ECSService`

다음 코드 예시는 `Remove-ECSService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 기본 클러스터에서 'my-http-service'라는 서비스를 삭제합니다. 서비스를 삭제하려면 먼저 원하는 개수와 실행 개수가 0이어야 합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 우회하려면 명령에 -Force 스위치를 추가합니다.

```
Remove-ECSService -Service my-http-service
```

예제 2: 명명된 클러스터에서 'my-http-service'라는 서비스를 삭제합니다.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteService를 참조하세요.

`Update-ECSClusterSetting`

다음 코드 예시는 `Update-ECSClusterSetting`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 ECS 클러스터에 사용할 설정을 수정합니다.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateClusterSettings을 참조하세요.

Update-ECSService

다음 코드 예시는 Update-ECSService의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제 명령에서는 `amazon-ecs-sample` 태스크 정의를 사용하도록 `my-http-service` 서비스를 업데이트합니다.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

예제 2: 이 예제 명령에서는 `my-http-service` 서비스의 원하는 개수를 10으로 업데이트합니다.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateService를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon EFS 예제

다음 코드 예제에서는 Amazon EFS에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Edit-EFSMountTargetSecurityGroup`

다음 코드 예시는 `Edit-EFSMountTargetSecurityGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 탑재 대상에 적용되는 보안 그룹을 업데이트합니다. 'sg-xxxxxxxx' 형식으로 최대 5개까지 지정할 수 있습니다.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-
group1,sg-group3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyMountTargetSecurityGroups을 참조하세요.

`Get-EFSFileSystem`

다음 코드 예시는 `Get-EFSFileSystem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 리전에서 호출자의 계정이 소유한 모든 파일 시스템 컬렉션을 반환합니다.

```
Get-EFSFileSystem
```

출력:

```
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
```

```
...
```

예제 2: 지정된 파일 시스템의 세부 정보를 반환합니다.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

예제 3: 파일 시스템이 생성될 때 지정된 멱등성 생성 토큰을 사용하여 파일 시스템의 세부 정보를 반환합니다.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeFileSystems을 참조하세요.

`Get-EFSMountTarget`

다음 코드 예시는 `Get-EFSMountTarget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 파일 시스템과 연결된 탑재 대상 컬렉션을 반환합니다.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

출력:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMountTargets을 참조하세요.

`Get-EFSMountTargetSecurityGroup`

다음 코드 예시는 `Get-EFSMountTargetSecurityGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 탑재 대상과 연결된 네트워크 인터페이스에 현재 할당된 보안 그룹의 ID를 반환합니다.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

출력:

```
sg-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMountTargetSecurityGroups을 참조하세요.

`Get-EFSTag`

다음 코드 예시는 `Get-EFSTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 파일 시스템에 현재 연결된 태그 모음을 반환합니다.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

출력:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

`New-EFSFileSystem`

다음 코드 예시는 `New-EFSFileSystem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 빈 파일 시스템을 새로 생성합니다. 멱등성 생성을 보장하는 데 사용되는 토큰은 자동으로 생성되며 반환된 객체의 **`CreationToken`** 멤버에서 액세스할 수 있습니다.

```
New-EFSFileSystem
```

출력:

```
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
```

예제 2: 멱등성 생성을 보장하기 위해 사용자 지정 토큰을 사용하여 새로운 빈 파일 시스템을 생성합니다.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFileSystem을 참조하세요.

New-EFSMountTarget

다음 코드 예시는 New-EFSMountTarget의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 파일 시스템의 새 탑재 대상을 생성합니다. 지정된 서브넷은 탑재 대상이 생성될 가상 프라이빗 클라우드(VPC)와 (서브넷의 주소 범위에서) 자동 할당될 IP 주소를 결정합니다. 할당된 IP 주소를 사용하여 이 파일 시스템을 Amazon EC2 인스턴스에 탑재할 수 있습니다. 보안 그룹이 지정되지 않았으므로 대상에 대해 생성된 네트워크 인터페이스는 서브넷의 VPC에 대한 기본 보안 그룹과 연결됩니다.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

출력:

```
FileSystemId       : fs-1a2b3c4d
```

```
IpAddress           : 10.0.0.131
LifeCycleState      : creating
MountTargetId       : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId             : 123456789012
SubnetId            : subnet-1a2b3c4d
```

예제 2: 자동 할당된 IP 주소를 사용하여 지정된 파일 시스템에 대한 새 탑재 대상을 생성합니다. 탑재 대상에 대해 생성된 네트워크 인터페이스는 지정된 보안 그룹과 연결됩니다(보안 그룹은 'sg-xxxxxxxx' 형식으로 최대 5개까지 지정할 수 있음).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

예제 3: 지정된 IP 주소를 사용하여 지정된 파일 시스템에 대한 새 탑재 대상을 생성합니다.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateMountTarget을 참조하세요.

New-EFSTag

다음 코드 예시는 New-EFSTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 파일 시스템에 태그 모음을 적용합니다. 키가 지정된 태그가 파일 시스템에 이미 있는 경우 태그 값이 업데이트됩니다.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

예제 2: 지정된 파일 시스템의 이름 태그를 설정합니다. 이 값은 Get-EFSFileSystem cmdlet을 사용할 때 다른 파일 시스템 세부 정보와 함께 반환됩니다.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTags를 참조하세요.

`Remove-EFSFileSystem`

다음 코드 예시는 `Remove-EFSFileSystem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 더 이상 사용되지 않는 지정된 파일 시스템을 삭제합니다(파일 시스템에 탑재 대상이 있는 경우 먼저 제거해야 함). cmdlet이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 **`-Force`** 스위치를 사용하세요.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFileSystem을 참조하세요.

`Remove-EFSMountTarget`

다음 코드 예시는 `Remove-EFSMountTarget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 탑재 대상을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 프롬프트를 차단하려면 **`-Force`** 스위치를 사용합니다. 이 작업은 대상을 통해 파일 시스템의 탑재를 강제로 중단합니다. 상황에 따라 이 명령을 실행하기 전에 파일 시스템의 탑재 해제를 고려할 수 있습니다.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteMountTarget을 참조하세요.

`Remove-EFSTag`

다음 코드 예시는 `Remove-EFSTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 파일 시스템에서 하나 이상의 태그 모음을 삭제합니다. cmdlet이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 **`-Force`** 스위치를 사용하세요.

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTags를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon EKS 예제

다음 코드 예제에서는 Amazon EKS에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-EKSResourceTag`

다음 코드 예시는 `Add-EKSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 지정된 태그를 지정된 resourceArn이 있는 리소스에 연결합니다.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResource를 참조하세요.

`Get-EKSCluster`

다음 코드 예시는 `Get-EKSCluster`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 클러스터에 대한 설명 정보를 반환합니다.

```
Get-EKSCluster -Name "PROD"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCluster를 참조하세요.

`Get-EKSClusterList`

다음 코드 예시는 `Get-EKSClusterList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 지정된 리전의에 AWS 계정 있는 Amazon EKS 클러스터를 나열합니다.

```
Get-EKSClusterList
```

출력:

```
PROD
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListClusters를 참조하세요.

`Get-EKSFargateProfile`

다음 코드 예시는 `Get-EKSFargateProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 AWS Fargate 프로필에 대한 설명 정보를 반환합니다.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeFargateProfile을 참조하세요.

`Get-EKSFargateProfileList`

다음 코드 예시는 `Get-EKSFargateProfileList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 지정된 리전의에서 지정된 클러스터와 연결된 AWS Fargate 프로파일을 나열 AWS 계정 합니다.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

출력:

```
EKSFargate
EKSFargateProfile
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListFargateProfiles을 참조하세요.

`Get-EKSNodegroup`

다음 코드 예시는 `Get-EKSNodegroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 노드 그룹에 대한 설명 정보를 반환합니다.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeNodegroup을 참조하세요.

`Get-EKSNodegroupList`

다음 코드 예시는 `Get-EKSNodegroupList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 지정된 리전의에서 지정된 클러스터와 연결된 Amazon EKS 노드 그룹을 나열 AWS 계정 합니다.

```
Get-EKSNodegroupList -ClusterName PROD
```

출력:

```
 ProdEKSNodeGroup
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListNodegroups을 참조하세요.

`Get-EKSResourceTag`

다음 코드 예시는 `Get-EKSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 리소스의 태그를 나열합니다.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

출력:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

`Get-EKSUpdate`

다음 코드 예시는 `Get-EKSUpdate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 클러스터 또는 연결된 관리형 노드 그룹의 업데이트에 대한 설명 정보를 반환합니다.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeUpdate를 참조하세요.

`Get-EKSUpdateList`

다음 코드 예시는 `Get-EKSUpdateList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 지정된 리전의에 있는 Amazon EKS 클러스터 또는 관리형 노드 그룹과 연결된 업데이트를 나열 AWS 계정합니다.

```
Get-EKSUpdateList -Name "PROD"
```

출력:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListUpdates를 참조하세요.

`New-EKSCluster`

다음 코드 예시는 `New-EKSCluster`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'prod'라는 새 클러스터를 생성합니다.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCluster를 참조하세요.

`New-EKSFargateProfile`

다음 코드 예시는 `New-EKSFargateProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 Amazon EKS 클러스터에 대한 AWS Fargate 프로파일을 생성합니다. Fargate 인프라에서 포드를 예약할 수 있으려면 클러스터에 하나 이상의 Fargate 프로파일이 있어야 합니다.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFargateProfile을 참조하세요.

New-EKSNodeGroup

다음 코드 예시는 New-EKSNodeGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 cmdlet은 Amazon EKS 클러스터에 대한 워커 노드 그룹을 생성합니다. 클러스터에 대한 현재 Kubernetes 버전과 동일한 클러스터에 대해서만 노드 그룹을 생성할 수 있습니다. 모든 노드 그룹은 클러스터의 해당 마이너 Kubernetes 버전에 대한 최신 AMI 릴리스 버전으로 생성됩니다.

```
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"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateNodegroup을 참조하세요.

`Remove-EKSCluster`

다음 코드 예시는 `Remove-EKSCluster`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 클러스터 컨트롤 플레인을 삭제합니다.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCluster를 참조하세요.

Remove-EKSFargateProfile

다음 코드 예시는 Remove-EKSFargateProfile의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 cmdlet은 AWS Fargate 프로파일을 삭제합니다. Fargate 프로파일을 삭제하면 해당 프로파일을 사용하여 생성된 Fargate에서 실행 중인 모든 포드가 삭제됩니다.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFargateProfile을 참조하세요.

`Remove-EKSNodegroup`

다음 코드 예시는 `Remove-EKSNodegroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 클러스터에 대한 Amazon EKS 노드 그룹을 삭제합니다.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteNodegroup을 참조하세요.

`Remove-EKSResourceTag`

다음 코드 예시는 `Remove-EKSResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 EKS 리소스에서 지정된 태그를 삭제합니다.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagResource를 참조하세요.

`Update-EKSClusterConfig`

다음 코드 예시는 `Update-EKSClusterConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Amazon EKS 클러스터 구성을 업데이트합니다. 업데이트 중에도 클러스터가 계속 작동합니다.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateClusterConfig를 참조하세요.

`Update-EKSClusterVersion`

다음 코드 예시는 `Update-EKSClusterVersion`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 cmdlet은 Amazon EKS 클러스터를 지정된 Kubernetes 버전으로 업데이트합니다. 업데이트 중에도 클러스터가 계속 작동합니다.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateClusterVersion을 참조하세요.

Tools for PowerShell V4를 사용한 Elastic Load Balancing - 버전 1 예제

다음 코드 예제에서는 Elastic Load Balancing - 버전 1과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-ELBLoadBalancerToSubnet`

다음 코드 예시는 `Add-ELBLoadBalancerToSubnet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대해 구성된 서브넷 세트에 지정된 서브넷을 추가합니다. 출력에는 전체 서브넷 목록이 포함됩니다.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

출력:

```
subnet-12345678
subnet-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachLoadBalancerToSubnets을 참조하세요.

`Add-ELBResourceTag`

다음 코드 예시는 `Add-ELBResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 태그를 지정된 로드 밸런서에 추가합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

예제 2: PowerShell 버전 2에서 Tag 파라미터에 대해 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddTags를 참조하세요.

`Disable-ELBAvailabilityZoneForLoadBalancer`

다음 코드 예시는 `Disable-ELBAvailabilityZoneForLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에서 지정된 가용 영역을 제거합니다. 출력에는 나머지 가용 영역이 포함됩니다.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

출력:

```
us-west-2b
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisableAvailabilityZonesForLoadBalancer를 참조하세요.

`Dismount-ELBLoadBalancerFromSubnet`

다음 코드 예시는 `Dismount-ELBLoadBalancerFromSubnet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대해 구성된 서브넷 세트에서 지정된 서브넷을 제거합니다. 출력에는 나머지 서브넷이 포함됩니다.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

출력:

```
subnet-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachLoadBalancerFromSubnets을 참조하세요.

`Edit-ELBLoadBalancerAttribute`

다음 코드 예시는 `Edit-ELBLoadBalancerAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대해 교차 영역 로드 밸런싱을 활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

예제 2: 이 예제에서는 지정된 로드 밸런서에 대한 연결 드레이닝을 비활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

예제 3: 이 예제에서는 지정된 로드 밸런서에 대한 액세스 로깅을 활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyLoadBalancerAttributes를 참조하세요.

`Enable-ELBAvailabilityZoneForLoadBalancer`

다음 코드 예시는 `Enable-ELBAvailabilityZoneForLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 지정된 가용 영역을 추가합니다. 출력에는 가용 영역의 전체 목록이 포함됩니다.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

출력:

```
us-west-2a
us-west-2b
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableAvailabilityZonesForLoadBalancer를 참조하세요.

`Get-ELBInstanceHealth`

다음 코드 예시는 `Get-ELBInstanceHealth`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 등록된 인스턴스의 상태를 설명합니다.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

출력:

```
Description                  InstanceId                   ReasonCode
      State
-----------                  ----------                   ----------
      -----
N/A                          i-87654321                   N/A
      InService
Instance has failed at lea... i-12345678                  Instance
      OutOfService
```

예제 2: 이 예제에서는 지정된 로드 밸런서에 등록되어 있는 지정된 인스턴스의 상태를 설명합니다.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

예제 3: 이 예제에서는 지정된 인스턴스의 상태에 대한 전체 설명을 표시합니다.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

출력:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstanceHealth를 참조하세요.

`Get-ELBLoadBalancer`

다음 코드 예시는 `Get-ELBLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 로드 밸런서의 이름을 나열합니다.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

출력:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

예제 2: 이 예제에서는 지정된 로드 밸런서를 설명합니다.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

출력:

```
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
```

예제 3:이 예제에서는 현재 AWS 리전의 모든 로드 밸런서를 설명합니다.

```
Get-ELBLoadBalancer
```

예제 4: 이 예제에서는 사용 가능한 모든 AWS 리전의 모든 로드 밸런서를 설명합니다.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancers를 참조하세요.

`Get-ELBLoadBalancerAttribute`

다음 코드 예시는 `Get-ELBLoadBalancerAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서의 속성을 설명합니다.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

출력:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
```

```
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancerAttributes를 참조하세요.

`Get-ELBLoadBalancerPolicy`

다음 코드 예시는 `Get-ELBLoadBalancerPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서와 연결된 정책을 설명합니다.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

출력:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
----------------------------            ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

예제 2: 이 예제에서는 지정된 정책의 속성을 설명합니다.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

출력:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

예제 3: 이 예제에서는 샘플 정책을 포함하여 사전 정의된 정책을 설명합니다. 샘플 정책의 이름에는 ELBSample- 접두사가 있습니다.

```
Get-ELBLoadBalancerPolicy
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancerPolicies를 참조하세요.

`Get-ELBLoadBalancerPolicyType`

다음 코드 예시는 `Get-ELBLoadBalancerPolicyType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Elastic Load Balancing에서 지원하는 정책 유형을 가져옵니다.

```
Get-ELBLoadBalancerPolicyType
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 정책 유형을 설명합니다.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

출력:

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

예제 3: 이 예제에서는 지정된 정책 유형에 대한 전체 설명을 표시합니다.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancerPolicyTypes을 참조하세요.

`Get-ELBResourceTag`

다음 코드 예시는 `Get-ELBResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서의 태그를 나열합니다.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

출력:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

예제 2: 이 예제에서는 지정된 로드 밸런서의 태그를 설명합니다.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

출력:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

`Join-ELBSecurityGroupToLoadBalancer`

다음 코드 예시는 `Join-ELBSecurityGroupToLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대한 현재 보안 그룹을 지정된 보안 그룹으로 대체합니다.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

출력:

```
sg-87654321
```

예제 2: 현재 보안 그룹을 유지하고 추가 보안 그룹을 지정하려면 기존 보안 그룹과 새 보안 그룹을 모두 지정합니다.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

출력:

```
sg-12345678
sg-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ApplySecurityGroupsToLoadBalancer를 참조하세요.

`New-ELBAppCookieStickinessPolicy`

다음 코드 예시는 `New-ELBAppCookieStickinessPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 애플리케이션 생성 쿠키의 스티키 세션 수명을 따르는 스티키 정책을 생성합니다.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAppCookieStickinessPolicy를 참조하세요.

New-ELBLBCookieStickinessPolicy

다음 코드 예시는 New-ELBLBCookieStickinessPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 스티키 세션 수명이 지정된 만료 기간(초)으로 제어되는 스티키 정책을 생성합니다.

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy -CookieExpirationPeriod 60
```

예제 2: 이 예제에서는 스티키 세션 수명이 브라우저의 수명(사용자 에이전트)으로 제어되는 스티키 정책을 생성합니다.

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLbCookieStickinessPolicy를 참조하세요.

New-ELBLoadBalancer

다음 코드 예시는 New-ELBLoadBalancer의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 VPC에 HTTP 리스너가 있는 로드 밸런서를 생성합니다.

```
$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
```

예제 2: 이 예제에서는 EC2-Classic에 HTTP 리스너가 있는 로드 밸런서를 생성합니다.

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

출력:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

예제 3: 이 예제에서는 HTTPS 리스너가 있는 로드 밸런서를 생성합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLoadBalancer를 참조하세요.

`New-ELBLoadBalancerListener`

다음 코드 예시는 `New-ELBLoadBalancerListener`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 HTTPS 리스너를 지정된 로드 밸런서에 추가합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLoadBalancerListeners를 참조하세요.

New-ELBLoadBalancerPolicy

다음 코드 예시는 New-ELBLoadBalancerPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대한 새 프록시 프로토콜 정책을 생성합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLoadBalancerPolicy를 참조하세요.

Register-ELBInstanceWithLoadBalancer

다음 코드 예시는 Register-ELBInstanceWithLoadBalancer의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 EC2 인스턴스를 지정된 로드 밸런서에 등록합니다.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterInstancesWithLoadBalancer를 참조하세요.

`Remove-ELBInstanceFromLoadBalancer`

다음 코드 예시는 `Remove-ELBInstanceFromLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 EC2 인스턴스를 지정된 로드 밸런서에서 제거합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterInstancesFromLoadBalancer를 참조하세요.

`Remove-ELBLoadBalancer`

다음 코드 예시는 `Remove-ELBLoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLoadBalancer를 참조하세요.

`Remove-ELBLoadBalancerListener`

다음 코드 예시는 `Remove-ELBLoadBalancerListener`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서의 포트 80에서 리스너를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLoadBalancerListeners를 참조하세요.

Remove-ELBLoadBalancerPolicy

다음 코드 예시는 Remove-ELBLoadBalancerPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에서 지정된 정책을 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLoadBalancerPolicy를 참조하세요.

Remove-ELBResourceTag

다음 코드 예시는 Remove-ELBResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 태그를 지정된 로드 밸런서에서 제거합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 프롬프트가 표시됩니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

출력:

```
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"):
```

예제 2: PowerShell 버전 2에서 Tag 파라미터에 대해 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveTags를 참조하세요.

`Set-ELBHealthCheck`

다음 코드 예시는 `Set-ELBHealthCheck`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 대한 상태 확인 설정을 구성합니다.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

출력:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ConfigureHealthCheck를 참조하세요.

Set-ELBLoadBalancerListenerSSLCertificate

다음 코드 예시는 Set-ELBLoadBalancerListenerSSLCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너에 대한 SSL 연결을 종료하는 인증서를 대체합니다.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetLoadBalancerListenerSslCertificate를 참조하세요.

Set-ELBLoadBalancerPolicyForBackendServer

다음 코드 예시는 Set-ELBLoadBalancerPolicyForBackendServer의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 포트의 정책을 지정된 정책으로 대체합니다.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

예제 2: 이 예제에서는 지정된 포트와 연결된 모든 정책을 제거합니다.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetLoadBalancerPoliciesForBackendServer를 참조하세요.

Set-ELBLoadBalancerPolicyOfListener

다음 코드 예시는 Set-ELBLoadBalancerPolicyOfListener의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너의 정책을 지정된 정책으로 대체합니다.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

예제 2: 이 예제에서는 지정된 리스너와 연결된 모든 정책을 제거합니다.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetLoadBalancerPoliciesOfListener를 참조하세요.

Tools for PowerShell V4를 사용한 Elastic Load Balancing - 버전 2 예제

다음 코드 예제에서는 Elastic Load Balancing - 버전 2와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-ELB2ListenerCertificate`

다음 코드 예시는 `Add-ELB2ListenerCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너에 추가 인증서를 추가합니다.

```
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'}
```

출력:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddListenerCertificates를 참조하세요.

`Add-ELB2Tag`

다음 코드 예시는 `Add-ELB2Tag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 **`AWS.Tools.ElasticLoadBalancingV2`** 리소스에 새 태그를 추가합니다.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddTags를 참조하세요.

`Edit-ELB2Listener`

다음 코드 예시는 `Edit-ELB2Listener`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너 기본 작업을 고정 응답으로 수정합니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyListener를 참조하세요.

`Edit-ELB2LoadBalancerAttribute`

다음 코드 예시는 `Edit-ELB2LoadBalancerAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서의 속성을 수정합니다.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyLoadBalancerAttributes를 참조하세요.

`Edit-ELB2Rule`

다음 코드 예시는 `Edit-ELB2Rule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된 리스너 규칙 구성을 수정합니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyRule을 참조하세요.

Edit-ELB2TargetGroup

다음 코드 예시는 Edit-ELB2TargetGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹의 속성을 수정합니다.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyTargetGroup을 참조하세요.

Edit-ELB2TargetGroupAttribute

다음 코드 예시는 Edit-ELB2TargetGroupAttribute의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹의 deregistration_delay 속성을 수정합니다.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyTargetGroupAttributes를 참조하세요.

`Get-ELB2AccountLimit`

다음 코드 예시는 `Get-ELB2AccountLimit`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 리전에 대한 ELB2 계정 한도를 나열합니다.

```
Get-ELB2AccountLimit
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAccountLimits을 참조하세요.

`Get-ELB2Listener`

다음 코드 예시는 `Get-ELB2Listener`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 ALB/NLB의 리스너를 설명합니다.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeListeners를 참조하세요.

`Get-ELB2ListenerCertificate`

다음 코드 예시는 `Get-ELB2ListenerCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너의 인증서를 설명합니다.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

출력:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeListenerCertificates를 참조하세요.

`Get-ELB2LoadBalancer`

다음 코드 예시는 `Get-ELB2LoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 리전의 모든 로드 밸런서를 표시합니다.

```
Get-ELB2LoadBalancer
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancers를 참조하세요.

`Get-ELB2LoadBalancerAttribute`

다음 코드 예시는 `Get-ELB2LoadBalancerAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 로드 밸런서의 속성을 설명합니다.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeLoadBalancerAttributes를 참조하세요.

Get-ELB2Rule

다음 코드 예시는 Get-ELB2Rule의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너 ARN의 리스너 규칙을 설명합니다.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeRules을 참조하세요.

Get-ELB2SSLPolicy

다음 코드 예시는 Get-ELB2SSLPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 ElasticLoadBalancingV2에 사용할 수 있는 모든 리스너 정책을 나열합니다.

```
Get-ELB2SSLPolicy
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSslPolicies를 참조하세요.

Get-ELB2Tag

다음 코드 예시는 Get-ELB2Tag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스의 태그를 나열합니다.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

출력:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

Get-ELB2TargetGroup

다음 코드 예시는 Get-ELB2TargetGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹을 설명합니다.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTargetGroups을 참조하세요.

`Get-ELB2TargetGroupAttribute`

다음 코드 예시는 `Get-ELB2TargetGroupAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹의 속성을 설명합니다.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTargetGroupAttributes를 참조하세요.

`Get-ELB2TargetHealth`

다음 코드 예시는 `Get-ELB2TargetHealth`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹에 있는 대상의 상태를 반환합니다.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

출력:

```
HealthCheckPort Target                                                  TargetHealth
--------------- ------                                                  ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTargetHealth를 참조하세요.

`New-ELB2Listener`

다음 코드 예시는 `New-ELB2Listener`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹으로 트래픽을 전송하기 위해 기본 작업 'Forward'를 사용하여 새 ALB 리스너를 생성합니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateListener를 참조하세요.

New-ELB2LoadBalancer

다음 코드 예시는 New-ELB2LoadBalancer의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 서브넷이 2개인 새 인터넷 연결 Application Load Balancer를 생성합니다.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLoadBalancer를 참조하세요.

New-ELB2Rule

다음 코드 예시는 New-ELB2Rule의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너의 고객 헤더 값을 기반으로 고정 응답 작업을 사용하여 새 리스너 규칙을 생성합니다.

```
$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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRule을 참조하세요.

New-ELB2TargetGroup

다음 코드 예시는 New-ELB2TargetGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 제공된 파라미터를 사용하여 새 대상 그룹을 생성합니다.

```
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'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTargetGroup을 참조하세요.

Register-ELB2Target

다음 코드 예시는 Register-ELB2Target의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹에 'i-0672a4c4cdeae3111' 인스턴스를 등록합니다.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterTargets을 참조하세요.

Remove-ELB2Listener

다음 코드 예시는 Remove-ELB2Listener의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너를 삭제합니다.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

출력:

```
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
```

예제 2: 이 예제에서는 로드 밸런서에서 지정된 리스너를 제거합니다.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteListener를 참조하세요.

`Remove-ELB2ListenerCertificate`

다음 코드 예시는 `Remove-ELB2ListenerCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹에서 지정된 인증서를 제거합니다.

```
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'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveListenerCertificates를 참조하세요.

`Remove-ELB2LoadBalancer`

다음 코드 예시는 `Remove-ELB2LoadBalancer`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서를 삭제합니다.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLoadBalancer를 참조하세요.

`Remove-ELB2Rule`

다음 코드 예시는 `Remove-ELB2Rule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리스너에서 지정된 규칙을 제거합니다.

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRule을 참조하세요.

Remove-ELB2Tag

다음 코드 예시는 Remove-ELB2Tag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 키의 태그를 제거합니다.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveTags를 참조하세요.

Remove-ELB2TargetGroup

다음 코드 예시는 Remove-ELB2TargetGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹을 제거합니다.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssss/4e0b6076bc6483a7'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTargetGroup을 참조하세요.

`Set-ELB2IpAddressType`

다음 코드 예시는 `Set-ELB2IpAddressType`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 로드 밸런서 IP 주소 유형을 'IPv4'에서 'DualStack'으로 변경합니다.

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

출력:

```
Value
-----
dualstack
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetIpAddressType을 참조하세요.

`Set-ELB2RulePriority`

다음 코드 예시는 `Set-ELB2RulePriority`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리스너 규칙의 우선순위를 변경합니다.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetRulePriorities를 참조하세요.

`Set-ELB2SecurityGroup`

다음 코드 예시는 `Set-ELB2SecurityGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서에 보안 그룹 'sg-07c3414abb8811cbd'를 추가합니다.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

출력:

```
sg-07c3414abb8811cbd
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetSecurityGroups을 참조하세요.

`Set-ELB2Subnet`

다음 코드 예시는 `Set-ELB2Subnet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 로드 밸런서의 서브넷을 수정합니다.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

출력:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetSubnets을 참조하세요.

`Unregister-ELB2Target`

다음 코드 예시는 `Unregister-ELB2Target`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대상 그룹에서 인스턴스 'i-0672a4c4cdeae3111'의 등록을 취소합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterTargets을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon FSx 예제

다음 코드 예제에서는 Amazon FSx에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

Add-FSXResourceTag

다음 코드 예시는 Add-FSXResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스에 태그를 추가합니다.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"} -PassThru
```

출력:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResource를 참조하세요.

Get-FSXBackup

다음 코드 예시는 Get-FSXBackup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파일 시스템 ID에 대해 어제 이후 생성된 백업을 가져옵니다.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeBackups을 참조하세요.

`Get-FSXFileSystem`

다음 코드 예시는 `Get-FSXFileSystem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 filesystemId에 대한 설명을 반환합니다.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeFileSystems을 참조하세요.

`Get-FSXResourceTagList`

다음 코드 예시는 `Get-FSXResourceTagList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 제공된 리소스 ARN에 대한 태그를 나열합니다.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

출력:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

`New-FSXBackup`

다음 코드 예시는 `New-FSXBackup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 파일 시스템의 백업을 생성합니다.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateBackup을 참조하세요.

New-FSXFileSystem

다음 코드 예시는 New-FSXFileSystem의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 초당 최대 8MB의 처리량을 지원하는 지정된 서브넷의 액세스를 허용하는 새 300GB Windows 파일 시스템을 생성합니다. 새 파일 시스템은 지정된 Microsoft Active Directory에 자동으로 조인됩니다.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFileSystem을 참조하세요.

New-FSXFileSystemFromBackup

다음 코드 예시는 New-FSXFileSystemFromBackup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기존 Amazon FSx for Windows File Server 백업에서 새 Amazon FSx 파일 시스템을 생성합니다.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFileSystemFromBackup을 참조하세요.

Remove-FSXBackup

다음 코드 예시는 Remove-FSXBackup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 backup-id를 제거합니다.

```
Remove-FSXBackup -BackupId $backupID
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBackup을 참조하세요.

Remove-FSXFileSystem

다음 코드 예시는 Remove-FSXFileSystem의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 FSX 파일 시스템 ID를 제거합니다.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFileSystem을 참조하세요.

`Remove-FSXResourceTag`

다음 코드 예시는 `Remove-FSXResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 FSX 파일 시스템 리소스 ARN에 대한 리소스 태그를 제거합니다.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagResource를 참조하세요.

`Update-FSXFileSystem`

다음 코드 예시는 `Update-FSXFileSystem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 UpdateFileSystemWindowsConfiguration을 통해 FSX 파일 시스템 자동 백업 보존 기간을 업데이트합니다.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
```

```
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateFileSystem을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon Glacier 예제

다음 코드 예제에서는 Amazon Glacier에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

Get-GLCJob

다음 코드 예시는 Get-GLCJob의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 작업의 세부 정보를 반환합니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 작업 콘텐츠(아카이브 또는 인벤토리 목록)를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeJob을 참조하세요.

`New-GLCVault`

다음 코드 예시는 `New-GLCVault`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 사용자 계정용 새 볼트를 생성합니다. -AccountId 파라미터에 값이 제공되지 않았으므로 cmdlet은 현재 계정을 나타내는 기본값인 '-'를 사용합니다.

```
New-GLCVault -VaultName myvault
```

출력:

```
/01234567812/vaults/myvault
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVault를 참조하세요.

`Read-GLCJobOutput`

다음 코드 예시는 `Read-GLCJobOutput`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 작업에서 가져오도록 예약된 아카이브 콘텐츠를 다운로드하고 콘텐츠를 디스크의 파일에 저장합니다. 다운로드 시 체크섬(있는 경우)이 검증됩니다. 원하는 경우 **`-Select '*'`**를 지정하여 체크섬을 포함한 전체 응답을 반환할 수 있습니다.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetJobOutput을 참조하세요.

`Start-GLCJob`

다음 코드 예시는 `Start-GLCJob`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 사용자가 소유한 지정된 볼트에서 아카이브를 가져오는 작업을 시작합니다. Get-GLCJob cmdlet을 사용하여 작업 상태를 확인할 수 있습니다. 작업이 성공적으로 완료되면 Read-GCJobOutput cmdlet을 사용하여 아카이브 콘텐츠를 로컬 파일 시스템으로 가져올 수 있습니다.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

출력:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 InitiateJob을 참조하세요.

`Write-GLCArchive`

다음 코드 예시는 `Write-GLCArchive`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 단일 파일을 지정된 볼트에 업로드하여 아카이브 ID와 계산된 체크섬을 반환합니다.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

출력:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

예제 2: 폴더 계층 구조의 콘텐츠를 사용자 계정의 지정된 볼트에 업로드합니다. 업로드된 각 파일에 대해 cmdlet은 파일 이름, 해당 아카이브 ID, 아카이브의 계산된 체크섬을 내보냅니다.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UploadArchive를 참조하세요.

AWS Glue Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Glue.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`New-GLUEJob`

다음 코드 예시는 `New-GLUEJob`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 AWS Glue에서 새 작업을 생성합니다. 명령 이름 값은 항상 입니다**`glueetl`**. AWS Glue는 Python 또는 Scala로 작성된 실행 중인 작업 스크립트를 지원합니다. 이 예제에서는 작업 스크립트(MyTestGlueJob.py)가 Python으로 작성되었습니다. Python 파라미터는 **`$DefArgs`** 변수에 지정된 다음 해시 테이블을 허용하는 **`DefaultArguments`** 파라미터의 PowerShell 명령에 전달됩니다. **`$JobParams`** 변수의 파라미터는 AWS Glue API 참조의 작업 (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) 주제에 설명된 CreateJob API에서 가져옵니다.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
```

```
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateJob을 참조하세요.

AWS Health Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Health.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-HLTHEvent`

다음 코드 예시는 `Get-HLTHEvent`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령은 AWS Personal Health Dashboard에서 이벤트를 반환합니다. 사용자는 -Region 파라미터를 추가하여 미국 동부(버지니아 북부) 리전에서 서비스에 사용할 수 있는 이벤트를 볼 수 있지만, -Filter_Region 파라미터는 EU(런던) 및 미국 서부(오리건) 리전(eu-west-2 및 us-west-2)에 로깅된 이벤트를 필터링합니다. -Filter_StartTime 파라미터는 이벤트가 시작되는 시간 범위를 기준으로 필터링하고, -Filter_EndTime 파라미터는 이벤트가 종료되는 시간 범위를 기준으로 필터링합니다. 지정된 -Filter_StartTime 범위 내에서 시작되어 예약된 -Filter_EndTime 범위 내에서 끝나는 RDS에 대한 예약된 유지 관리 이벤트가 결과로 반환됩니다.

```
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"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEvents를 참조하세요.

Tools for PowerShell V4를 사용한 IAM 예제

다음 코드 예제에서는 IAM과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-IAMClientIDToOpenIDConnectProvider`

다음 코드 예시는 `Add-IAMClientIDToOpenIDConnectProvider`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 클라이언트 ID 또는 대상 **`my-application-ID`**를 **`server.example.com`**이라는 기존 OIDC 제공업체에 추가합니다.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddClientIdToOpenIdConnectProvider를 참조하세요.

`Add-IAMRoleTag`

다음 코드 예시는 `Add-IAMRoleTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ID 관리 서비스의 역할에 태그를 추가합니다.

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagRole을 참조하세요.

`Add-IAMRoleToInstanceProfile`

다음 코드 예시는 `Add-IAMRoleToInstanceProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 **`webserver`**라는 기존 인스턴스 프로파일에 **`S3Access`**라는 역할을 추가합니다. 인스턴스 프로파일을 생성하려면 **`New-IAMInstanceProfile`** 명령을 사용합니다. 이 명령을 사용하여 인스턴스 프로파일을 생성하고 이를 역할과 연결한 후 EC2 인스턴스에 연결할 수 있습니다. 이를 수행하려면 **`InstanceProfile_Arn`** 또는 **`InstanceProfile-Name`** 파라미터와 함께 **`New-EC2Instance`** cmdlet을 사용하여 새 인스턴스를 시작합니다.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddRoleToInstanceProfile을 참조하세요.

`Add-IAMUserTag`

다음 코드 예시는 `Add-IAMUserTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ID 관리 서비스의 사용자에게 태그를 추가합니다.

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagUser를 참조하세요.

`Add-IAMUserToGroup`

다음 코드 예시는 `Add-IAMUserToGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 **Bob**이라는 사용자를 **Admins**라는 그룹에 추가합니다.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddUserToGroup을 참조하세요.

`Disable-IAMMFADevice`

다음 코드 예시는 `Disable-IAMMFADevice`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 일련 번호 **123456789012**를 가진 사용자 **Bob**과 연결된 하드웨어 MFA 디바이스를 비활성화합니다.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

예제 2: 이 명령은 ARN **arn:aws:iam::210987654321:mfa/David**를 가진 사용자 **David**과 연결된 가상 MFA 디바이스를 비활성화합니다. 단, 가상 MFA 디바이스는 계정에서 삭제되지 않습니다. 가상 디바이스는 여전히 존재하며 **Get-IAMVirtualMFADevice** 명령 출력에 나타납니다. 동일한 사용자를 위한 새 가상 MFA 디바이스를 생성하려면 먼저 **Remove-IAMVirtualMFADevice** 명령을 사용하여 이전 디바이스를 삭제해야 합니다.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeactivateMfaDevice를 참조하세요.

`Edit-IAMPassword`

다음 코드 예시는 `Edit-IAMPassword`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 명령을 실행하는 사용자의 암호를 변경합니다. 이 명령은 IAM 사용자만 호출할 수 있습니다. AWS 계정(루트) 자격 증명으로 로그인할 때이 명령이 호출되면 명령이 **InvalidUserType** 오류를 반환합니다.

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ChangePassword를 참조하세요.

Enable-IAMMFADevice

다음 코드 예시는 Enable-IAMMFADevice의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 일련 번호 **987654321098**을 사용하여 하드웨어 MFA 디바이스를 활성화하고 디바이스를 사용자 **Bob**과 연결합니다. 여기에는 디바이스의 처음 두 코드가 순서대로 포함됩니다.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

예제 2: 이 예제는 가상 MFA 디바이스를 생성하고 활성화합니다. 첫 번째 명령은 가상 디바이스를 생성하고 **$MFADevice** 변수에 디바이스의 객체 표현을 반환합니다. **.Base32StringSeed** 또는 **QRCodePng** 속성을 사용하여 사용자의 소프트웨어 애플리케이션을 구성할 수 있습니다. 마지막 명령은 사용자 **David**에게 디바이스를 할당하고 일련 번호로 디바이스를 식별합니다. 또한 명령은 가상 MFA 디바이스의 처음 두 코드를 순서대로 AWS 포함하여 디바이스를와 동기화합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 EnableMfaDevice를 참조하세요.

Get-IAMAccessKey

다음 코드 예시는 Get-IAMAccessKey의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 **Bob**이라는 IAM 사용자의 액세스 키를 나열합니다. IAM 사용자의 시크릿 액세스 키는 나열할 수 없습니다. 시크릿 액세스 키를 분실한 경우 **New-IAMAccessKey** cmdlet을 사용하여 새 액세스 키를 생성해야 합니다.

```
Get-IAMAccessKey -UserName "Bob"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAccessKeys를 참조하세요.

Get-IAMAccessKeyLastUsed

다음 코드 예시는 `Get-IAMAccessKeyLastUsed`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 제공된 액세스 키의 소유 사용자 이름과 마지막 사용 정보를 반환합니다.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccessKeyLastUsed를 참조하세요.

Get-IAMAccountAlias

다음 코드 예시는 `Get-IAMAccountAlias`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 AWS 계정에 대한 계정 별칭을 반환합니다.

```
Get-IAMAccountAlias
```

출력:

```
ExampleCo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAccountAliases를 참조하세요.

`Get-IAMAccountAuthorizationDetail`

다음 코드 예시는 `Get-IAMAccountAuthorizationDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS 계정의 자격 증명에 대한 권한 부여 세부 정보를 가져오고 사용자, 그룹 및 역할을 포함하여 반환된 객체의 요소 목록을 표시합니다. 예를 들어, **`UserDetailList`** 속성은 사용자에 대한 세부 정보를 표시합니다. **`RoleDetailList`** 및 **`GroupDetailList`** 속성에서 유사한 정보를 확인할 수 있습니다.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

출력:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccountAuthorizationDetails를 참조하세요.

`Get-IAMAccountPasswordPolicy`

다음 코드 예시는 `Get-IAMAccountPasswordPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 계정의 암호 정책에 대한 세부 정보를 반환합니다. 계정에 대해 정의된 암호 정책이 없는 경우 명령은 **`NoSuchEntity`** 오류를 반환합니다.

```
Get-IAMAccountPasswordPolicy
```

출력:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
```

```
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccountPasswordPolicy를 참조하세요.

`Get-IAMAccountSummary`

다음 코드 예시는 `Get-IAMAccountSummary`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 AWS 계정의 현재 IAM 엔터티 사용량과 현재 IAM 엔터티 할당량에 대한 정보를 반환합니다.

```
Get-IAMAccountSummary
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccountSummary를 참조하세요.

`Get-IAMAttachedGroupPolicyList`

다음 코드 예시는 `Get-IAMAttachedGroupPolicyList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령은 AWS 계정에서 라는 IAM 그룹에 연결된 관리형 정책의 이름과 ARNs**Admins**을 반환합니다. 그룹에 포함된 인라인 정책의 목록을 보려면 **`Get-IAMGroupPolicyList`** 명령을 사용합니다.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

출력:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAttachedGroupPolicies를 참조하세요.

Get-IAMAttachedRolePolicyList

다음 코드 예시는 Get-IAMAttachedRolePolicyList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 AWS 계정의 이름이 **SecurityAuditRole**인 IAM 역할에 연결된 관리형 정책의 이름과 ARN을 반환합니다. 역할에 포함된 인라인 정책의 목록을 보려면 **Get-IAMRolePolicyList** 명령을 사용합니다.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

출력:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAttachedRolePolicies를 참조하세요.

Get-IAMAttachedUserPolicyList

다음 코드 예시는 Get-IAMAttachedUserPolicyList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령은 AWS 계정에서 이름이 인 IAM 사용자**Bob**에 대한 관리형 정책의 이름과 ARNs을 반환합니다. IAM 사용자에게 포함된 인라인 정책의 목록을 보려면 **Get-IAMUserPolicyList** 명령을 사용합니다.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

출력:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/TesterPolicy                      TesterPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAttachedUserPolicies를 참조하세요.

`Get-IAMContextKeysForCustomPolicy`

다음 코드 예시는 `Get-IAMContextKeysForCustomPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키를 가져옵니다. 여러 정책을 제공하려면 쉼표로 구분된 값 목록으로 제공합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetContextKeysForCustomPolicy를 참조하세요.

`Get-IAMContextKeysForPrincipalPolicy`

다음 코드 예시는 `Get-IAMContextKeysForPrincipalPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키와 IAM 엔티티(사용자/역할 등)에 연결된 정책을 가져옵니다. -PolicyInputList의 경우 여러 값 목록을 쉼표로 구분된 값으로 제공할 수 있습니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetContextKeysForPrincipalPolicy를 참조하세요.

`Get-IAMCredentialReport`

다음 코드 예시는 `Get-IAMCredentialReport`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 반환된 보고서를 열고 파이프라인에 텍스트 라인 배열로 출력합니다. 줄 1은 쉼표로 구분된 열 이름이 있는 헤더입니다. 그 후의 각 행은 한 사용자에 대한 세부 정보 행이며 각 필드는 쉼표로 구분됩니다. 보고서를 보려면 먼저 **Request-IAMCredentialReport** cmdlet을 사용하여 보고서를 생성해야 합니다. 보고서를 단일 문자열로 검색하려면 **-AsTextArray** 대신 **-Raw**를 사용합니다. **-AsTextArray** 스위치에는 **-SplitLines** 별칭도 허용됩니다. 출력의 전체 열 목록은 서비스 API 참조를 참조하세요. **-AsTextArray** 또는 **-SplitLines**를 사용하지 않는 경우 .NET **StreamReader** 클래스를 사용하여 **.Content** 속성에서 텍스트를 추출해야 합니다.

```
Request-IAMCredentialReport
```

출력:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

출력:

```
 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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCredentialReport를 참조하세요.

`Get-IAMEntitiesForPolicy`

다음 코드 예시는 `Get-IAMEntitiesForPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`arn:aws:iam::123456789012:policy/TestPolicy`** 정책이 연결된 IAM 그룹, 역할 및 사용자 목록을 반환합니다.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

출력:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListEntitiesForPolicy를 참조하세요.

`Get-IAMGroup`

다음 코드 예시는 `Get-IAMGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 그룹에 속한 모든 IAM 사용자의 모음을 포함하여 IAM 그룹 **`Testers`**에 대한 세부 정보를 반환합니다.

```
$results = Get-IAMGroup -GroupName "Testers"
```

```
$results
```

출력:

```
Group                                    IsTruncated           Marker
  Users
-----                                    -----------           ------
  -----
Amazon.IdentityManagement.Model.Group    False
  {Theresa, David}
```

```
$results.Group
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetGroup을 참조하세요.

Get-IAMGroupForUser

다음 코드 예시는 Get-IAMGroupForUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자 **David**가 속한 IAM 그룹 목록을 반환합니다.

```
Get-IAMGroupForUser -UserName David
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListGroupsForUser를 참조하세요.

Get-IAMGroupList

다음 코드 예시는 Get-IAMGroupList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 현재에 정의된 모든 IAM 그룹의 모음을 반환합니다 AWS 계정.

```
Get-IAMGroupList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListGroups를 참조하세요.

`Get-IAMGroupPolicy`

다음 코드 예시는 `Get-IAMGroupPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`Testers`** 그룹에 대해 **`PowerUserAccess-Testers`**라는 포함된 인라인 정책에 대한 세부 정보를 반환합니다. **`PolicyDocument`** 속성은 URL로 인코딩됩니다. 이 예제에서는 **`UrlDecode`** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

출력:

```
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",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetGroupPolicy를 참조하세요.

`Get-IAMGroupPolicyList`

다음 코드 예시는 `Get-IAMGroupPolicyList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 그룹 **`Testers`**에 포함된 인라인 정책의 목록을 반환합니다. 그룹에 연결된 관리형 정책을 가져오려면 **`Get-IAMAttachedGroupPolicyList`** 명령을 사용합니다.

```
Get-IAMGroupPolicyList -GroupName Testers
```

출력:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListGroupPolicies를 참조하세요.

`Get-IAMInstanceProfile`

다음 코드 예시는 `Get-IAMInstanceProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제**ec2instancerole**는 현재 AWS 계정에 정의된 인스턴스 프로파일의 세부 정보를 반환합니다.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetInstanceProfile을 참조하세요.

`Get-IAMInstanceProfileForRole`

다음 코드 예시는 `Get-IAMInstanceProfileForRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **ec2instancerole** 역할과 연결된 인스턴스 프로파일의 세부 정보를 반환합니다.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

출력:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListInstanceProfilesForRole을 참조하세요.

`Get-IAMInstanceProfileList`

다음 코드 예시는 `Get-IAMInstanceProfileList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 현재에 정의된 인스턴스 프로파일의 컬렉션을 반환합니다 AWS 계정.

```
Get-IAMInstanceProfileList
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListInstanceProfiles를 참조하세요.

`Get-IAMLoginProfile`

다음 코드 예시는 `Get-IAMLoginProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 암호 생성 날짜와 IAM 사용자 **David**에 대해 암호 재설정이 필요한지 여부를 반환합니다.

```
Get-IAMLoginProfile -UserName David
```

출력:

```
CreateDate                  PasswordResetRequired                UserName
```

```
----------                    ---------------------                --------
12/10/2014 3:39:44 PM         False                                David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetLoginProfile을 참조하세요.

Get-IAMMFADevice

다음 코드 예시는 Get-IAMMFADevice의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자 **David**에게 할당된 MFA 디바이스에 대한 세부 정보를 반환합니다. 이 예제에서는 **SerialNumber**가 물리적 디바이스의 실제 일련 번호가 아닌 ARN이므로 가상 장치임을 알 수 있습니다.

```
Get-IAMMFADevice -UserName David
```

출력:

```
EnableDate                SerialNumber                          UserName
----------                ------------                          --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David   David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListMfaDevices를 참조하세요.

Get-IAMOpenIDConnectProvider

다음 코드 예시는 Get-IAMOpenIDConnectProvider의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com**인 OpenID Connect 제공업체에 대한 세부 정보를 반환합니다. **ClientIDList** 속성은 이 제공업체에 대해 정의된 모든 클라이언트 ID를 포함하는 컬렉션입니다.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

출력:

```
ClientIDList         CreateDate                ThumbprintList
        Url
------------         ----------                --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetOpenIdConnectProvider를 참조하세요.

`Get-IAMOpenIDConnectProviderList`

다음 코드 예시는 `Get-IAMOpenIDConnectProviderList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 AWS 계정에 정의된 모든 OpenID Connect 제공업체의 ARNS 목록을 반환합니다.

```
Get-IAMOpenIDConnectProviderList
```

출력:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListOpenIdConnectProviders를 참조하세요.

`Get-IAMPolicy`

다음 코드 예시는 `Get-IAMPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **`arn:aws:iam::123456789012:policy/MySamplePolicy`**인 관리형 정책에 대한 세부 정보를 반환합니다.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPolicy를 참조하세요.

`Get-IAMPolicyList`

다음 코드 예시는 `Get-IAMPolicyList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 현재 AWS 계정에서 사용 가능한 처음 3개의 관리형 정책 모음을 반환합니다. **-scope**는 지정되지 않았으므로 기본값은 **all** 이며 AWS 관리형 정책과 고객 관리형 정책을 모두 포함합니다.

```
Get-IAMPolicyList -MaxItem 3
```

출력:

```
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
```

예제 2:이 예제는 현재 AWS 계정에서 사용할 수 있는 처음 두 개의 고객 관리형 정책 모음을 반환합니다. **`-Scope local`**을 사용하여 고객 관리형 정책으로만 출력을 제한합니다.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListPolicies를 참조하세요.

`Get-IAMPolicyVersion`

다음 코드 예시는 `Get-IAMPolicyVersion`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**인 정책의 **`v2`** 버전에 대한 정책 문서를 반환합니다. **`Document`** 속성의 정책 문서는 URL로 인코딩되며 이 예제에서는 **`UrlDecode`** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

출력:

```
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": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPolicyVersion을 참조하세요.

Get-IAMPolicyVersionList

다음 코드 예시는 Get-IAMPolicyVersionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyManagedPolicy**인 정책의 사용 가능한 버전 목록을 반환합니다. 특정 버전에 대한 정책 문서를 가져오려면 **Get-IAMPolicyVersion** 명령을 사용하고 원하는 버전의 **VersionId**를 지정합니다.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

출력:

```
CreateDate                  Document                IsDefaultVersion
    VersionId
----------                  --------                ----------------
    ---------
2/12/2015 9:39:53 AM                                True
    v2
2/12/2015 9:39:09 AM                                False
    v1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListPolicyVersions를 참조하세요.

Get-IAMRole

다음 코드 예시는 Get-IAMRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **lamda_exec_role**의 세부 정보를 반환합니다. 여기에는 이 역할을 수임할 수 있는 사용자를 지정하는 신뢰 정책 문서가 포함됩니다. 정책 문서는 URL로 인코딩되며 .NET **UrlDecode** 메서드를 사용하여 디코딩할 수 있습니다. 이 예제에서는 원래 정책에 업로드되기 전에 모든 공백이 제거됩니다. 역할을 수임한 사람이 수행할 수 있는 작업을 결정하는 권한 정책 문서를 보려면 인라인 정책에는 **Get-IAMRolePolicy**를 사용하고 연결된 관리형 정책에는 **Get-IAMPolicyVersion**을 사용합니다.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

출력:

```
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
```

출력:

```
{"Version":"2012-10-17",       "Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetRole을 참조하세요.

Get-IAMRoleList

다음 코드 예시는 Get-IAMRoleList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 AWS 계정의 모든 IAM 역할 목록을 검색합니다.

```
Get-IAMRoleList
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListRoles를 참조하세요.

`Get-IAMRolePolicy`

다음 코드 예시는 `Get-IAMRolePolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 역할 **`lamda_exec_role`**에 포함된 **`oneClick_lambda_exec_role_policy`**라는 정책에 대한 권한 정책 문서를 반환합니다. 결과 정책 문서는 URL로 인코딩됩니다. 이 예제에서는 **`UrlDecode`** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

출력:

```
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)
```

출력:

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetRolePolicy를 참조하세요.

`Get-IAMRolePolicyList`

다음 코드 예시는 `Get-IAMRolePolicyList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 역할 **`lamda_exec_role`**에 포함된 인라인 정책 이름의 목록을 반환합니다. 인라인 정책의 세부 정보를 보려면 **`Get-IAMRolePolicy`** 명령을 사용합니다.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

출력:

```
oneClick_lambda_exec_role_policy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListRolePolicies를 참조하세요.

Get-IAMRoleTagList

다음 코드 예시는 Get-IAMRoleTagList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 역할과 연결된 태그를 가져옵니다.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListRoleTags를 참조하세요.

Get-IAMSAMLProvider

다음 코드 예시는 Get-IAMSAMLProvider의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARM이 arn:aws:iam::123456789012:saml-provider/SAMLADFS인 SAML 2.0 제공업체에 대한 세부 정보를 검색합니다. 응답에는 AWS SAML 공급자 개체를 생성하기 위해 자격 증명 공급자로부터 받은 메타데이터 문서와 생성 및 만료 날짜가 포함됩니다.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetSamlProvider를 참조하세요.

Get-IAMSAMLProviderList

다음 코드 예시는 Get-IAMSAMLProviderList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 AWS 계정에서 생성된 SAML 2.0 제공업체 목록을 검색합니다. 각 SAML 제공업체의 ARN, 생성 날짜 및 만료 날짜를 반환합니다.

```
Get-IAMSAMLProviderList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListSAMLProviders를 참조하세요.

Get-IAMServerCertificate

다음 코드 예시는 Get-IAMServerCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **MyServerCertificate**라는 서버 인증서에 대한 세부 정보를 검색합니다. **CertificateBody** 및 **ServerCertificateMetadata** 속성에서 인증서 세부 정보를 찾을 수 있습니다.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

출력:

```
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
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetServerCertificate를 참조하세요.

`Get-IAMServerCertificateList`

다음 코드 예시는 `Get-IAMServerCertificateList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 AWS 계정에 업로드된 서버 인증서 목록을 검색합니다.

```
Get-IAMServerCertificateList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListServerCertificates를 참조하세요.

`Get-IAMServiceLastAccessedDetail`

다음 코드 예시는 `Get-IAMServiceLastAccessedDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예에서는 요청 직접 호출과 연결된 IAM 엔터티(사용자, 그룹, 역할 또는 정책)가 마지막으로 액세스한 서비스에 대한 세부 정보를 제공합니다.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

출력:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetServiceLastAccessedDetails를 참조하세요.

`Get-IAMServiceLastAccessedDetailWithEntity`

다음 코드 예시는 `Get-IAMServiceLastAccessedDetailWithEntity`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 해당 IAM 엔터티의 요청에 있는 서비스에 대해 마지막으로 액세스한 타임스탬프를 제공합니다.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

출력:

```
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
```

출력:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

출력:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetServiceLastAccessedDetailsWithEntities를 참조하세요.

`Get-IAMSigningCertificate`

다음 코드 예시는 `Get-IAMSigningCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`Bob`**이라는 사용자와 연결된 서명 인증서에 대한 세부 정보를 검색합니다.

```
Get-IAMSigningCertificate -UserName Bob
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListSigningCertificates를 참조하세요.

Get-IAMUser

다음 코드 예시는 Get-IAMUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **David**라는 사용자에 대한 세부 정보를 검색합니다.

```
Get-IAMUser -UserName David
```

출력:

```
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
```

예제 2: 이 예제는 현재 로그인한 IAM 사용자에 대한 세부 정보를 검색합니다.

```
Get-IAMUser
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetUser를 참조하세요.

Get-IAMUserList

다음 코드 예시는 Get-IAMUserList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 현재의 사용자 모음을 검색합니다 AWS 계정.

```
Get-IAMUserList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListUsers를 참조하세요.

`Get-IAMUserPolicy`

다음 코드 예시는 `Get-IAMUserPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **David**라는 IAM 사용자에게 포함된 **Davids_IAM_Admin_Policy**라는 인라인 정책의 세부 정보를 검색합니다. 정책 문서는 URL로 인코딩됩니다.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
```

```
$results
```

출력:

```
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:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetUserPolicy를 참조하세요.

`Get-IAMUserPolicyList`

다음 코드 예시는 `Get-IAMUserPolicyList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`David`**라는 IAM 사용자에게 포함된 인라인 정책의 이름 목록을 검색합니다.

```
Get-IAMUserPolicyList -UserName David
```

출력:

```
Davids_IAM_Admin_Policy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListUserPolicies를 참조하세요.

`Get-IAMUserTagList`

다음 코드 예시는 `Get-IAMUserTagList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 사용자와 연결된 태그를 가져옵니다.

```
Get-IAMUserTagList -UserName joe
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListUserTags를 참조하세요.

`Get-IAMVirtualMFADevice`

다음 코드 예시는 `Get-IAMVirtualMFADevice`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는 AWS 계정의 사용자에게 할당된 가상 MFA 디바이스의 모음을 검색합니다. 각각의 **User** 속성은 해당 디바이스가 할당된 IAM 사용자의 세부 정보가 포함된 객체입니다.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListVirtualMfaDevices를 참조하세요.

New-IAMAccessKey

다음 코드 예시는 New-IAMAccessKey의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 새 액세스 키와 시크릿 액세스 키 페어를 생성하여 사용자 **David**에게 할당합니다. 이때만 **SecretAccessKey**를 얻을 수 있으므로 **AccessKeyId** 및 **SecretAccessKey** 값을 파일에 저장해야 합니다. 나중에 검색할 수 없습니다. 보안 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

```
New-IAMAccessKey -UserName David
```

출력:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAccessKey를 참조하세요.

New-IAMAccountAlias

다음 코드 예시는 New-IAMAccountAlias의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 계정의 AWS 계정 별칭을 로 변경합니다**mycompanyaws**. 사용자 로그온 페이지의 주소가 https://mycompanyaws.signin.aws.amazon.com/console로 변경됩니다. 별칭 대신 계정 ID 번호를 사용하는 원래 URL(https://<accountidnumber>.signin.aws.amazon.com/console)은 계속 작동합니다. 하지만 이전에 정의된 별칭 기반 URL은 모두 작동이 중지됩니다.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAccountAlias를 참조하세요.

New-IAMGroup

다음 코드 예시는 New-IAMGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Developers**라는 새 IAM 그룹을 생성합니다.

```
New-IAMGroup -GroupName Developers
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateGroup을 참조하세요.

New-IAMInstanceProfile

다음 코드 예시는 New-IAMInstanceProfile의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **ProfileForDevEC2Instance**라는 새 IAM 인스턴스 프로파일을 생성합니다. **Add-IAMRoleToInstanceProfile** 명령을 별도로 실행하여 인스턴스에 권한을 제공하는 기존 IAM 역할과 인스턴스 프로파일을 연결해야 합니다. 마지막으로 EC2 인스턴스를 시작할 때 인스턴스 프로파일을 EC2 인스턴스에 연결합니다. 이를 수행하려면 **InstanceProfile_Arn** 또는 **InstanceProfile_Name** 파라미터와 함께 **New-EC2Instance** cmdlet을 사용합니다.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateInstanceProfile을 참조하세요.

New-IAMLoginProfile

다음 코드 예시는 New-IAMLoginProfile의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 Bob이라는 IAM 사용자에 대한 (임시) 암호를 생성하고 다음에 **Bob**이 로그인할 때 사용자가 암호를 변경하도록 요구하는 플래그를 설정합니다.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

출력:

```
CreateDate                  PasswordResetRequired               UserName
----------                  ---------------------               --------
4/14/2015 12:26:30 PM       True                                Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateLoginProfile을 참조하세요.

New-IAMOpenIDConnectProvider

다음 코드 예시는 New-IAMOpenIDConnectProvider의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 URL **https://example.oidcprovider.com** 및 클라이언트 ID **my-testapp-1**에 있는 OIDC 호환 제공업체 서비스와 연결된 IAM OIDC 제공업체를 생성합니다.

OIDC 제공업체가 지문을 제공합니다. 지문을 인증하려면 http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html의 단계를 따르세요.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

출력:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateOpenIdConnectProvider를 참조하세요.

New-IAMPolicy

다음 코드 예시는 New-IAMPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 라는 현재 AWS 계정에 새 IAM 정책을 생성합니다. **MySamplePolicy** 파일은 정책 콘텐츠를 **MySamplePolicy.json** 제공합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreatePolicy를 참조하세요.

New-IAMPolicyVersion

다음 코드 예시는 New-IAMPolicyVersion의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyPolicy**인 IAM 정책의 새 'v2' 버전을 생성하고 이를 기본 버전으로 만듭니다. **NewPolicyVersion.json** 파일은 정책 콘텐츠를 제공합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

출력:

```
CreateDate                          Document                IsDefaultVersion
        VersionId
----------                          --------                ----------------
        ---------
4/15/2015 10:54:54 AM                                       True
        v2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreatePolicyVersion을 참조하세요.

New-IAMRole

다음 코드 예시는 New-IAMRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **MyNewRole**이라는 새 역할을 생성하고 여기에 **NewRoleTrustPolicy.json** 파일에 있는 정책을 연결합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다. 출력에 표시된 정책 문서는 URL로 인코딩됩니다. 이 예제에서는 **UrlDecode** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

출력:

```
Arn                      : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D
%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A
%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate               : 4/15/2015 11:04:23 AM
Path                     : /
RoleId                   : V5PAJI2KPN4EAEXAMPLE1
RoleName                 : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRole을 참조하세요.

New-IAMSAMLProvider

다음 코드 예시는 New-IAMSAMLProvider의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM에 새 SAML 제공업체를 생성합니다. 이름은 **MySAMLProvider**이며 SAML 서비스 제공업체의 웹 사이트에서 별도로 다운로드한 **SAMLMetaData.xml** 파일에 있는 SAML 메타데이터 문서에 설명되어 있습니다.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

출력:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateSAMLProvider를 참조하세요.

New-IAMServiceLinkedRole

다음 코드 예시는 New-IAMServiceLinkedRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 AutoScaling 서비스에 대한 서비스 연결 역할을 생성합니다.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateServiceLinkedRole을 참조하세요.

New-IAMUser

다음 코드 예시는 New-IAMUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Bob**이라는 IAM 사용자를 생성합니다. Bob이 AWS 콘솔에 로그인해야 하는 경우 명령을 별도로 실행**New-IAMLoginProfile**하여 암호로 로그인 프로필을 생성해야 합니다. Bob

이 AWS PowerShell 또는 교차 플랫폼 CLI 명령을 실행하거나 AWS API 호출을 수행해야 하는 경우 **New-IAMAccessKey** 명령을 별도로 실행하여 액세스 키를 생성해야 합니다.

```
New-IAMUser -UserName Bob
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateUser를 참조하세요.

New-IAMVirtualMFADevice

다음 코드 예시는 New-IAMVirtualMFADevice의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 새 가상 MFA 디바이스를 생성합니다. 줄 2와 3은 가상 MFA 소프트웨어 프로그램에서 계정을 생성하는 데 필요한 **Base32StringSeed** 값을 QR 코드의 대안으로 추출합니다. 값으로 프로그램을 구성한 후 프로그램에서 두 개의 순차적 인증 코드를 받습니다. 끝으로 마지막 명령을 사용하여 가상 MFA 디바이스를 IAM 사용자 **Bob**에게 연결하고 계정을 두 개의 인증 코드와 동기화합니다.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

출력:

```
-- 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
```

예제 2: 이 예제는 새 가상 MFA 디바이스를 생성합니다. 줄 2와 3은 **QRCodePNG** 값을 추출하여 파일에 씁니다. Base32StringSeed 값을 수동으로 입력하는 대신 가상 MFA 소프트웨어 프로그램에서 이 이미지를 스캔하여 계정을 생성할 수 있습니다. 가상 MFA 프로그램에서 계정을 생성한 후 두 개의 순차적 인증 코드를 받아 마지막 명령에 입력하여 가상 MFA 디바이스를 IAM 사용자 **Bob**에게 연결하고 계정을 동기화합니다.

```
$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
```

출력:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateVirtualMfaDevice를 참조하세요.

`Publish-IAMServerCertificate`

다음 코드 예시는 `Publish-IAMServerCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 새 서버 인증서를 IAM 계정에 업로드합니다. 인증서 본문, 프라이빗 키 및 인증서 체인(선택 사항)이 포함된 파일은 모두 PEM 인코딩되어야 합니다. 파라미터에는 파일 이름 대신 파일의 실제 내용이 필요합니다. 파일 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UploadServerCertificate를 참조하세요.

Publish-IAMSigningCertificate

다음 코드 예시는 Publish-IAMSigningCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 새로운 X.509 서명 인증서를 업로드하고 이를 **Bob**이라는 IAM 사용자와 연결합니다. 인증서 본문이 포함된 파일은 PEM으로 인코딩됩니다. **CertificateBody** 파라미터에는 파일 이름이 아닌 인증서 파일의 실제 내용이 필요합니다. 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UploadSigningCertificate를 참조하세요.

Register-IAMGroupPolicy

다음 코드 예시는 Register-IAMGroupPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **TesterPolicy**라는 고객 관리형 정책을 IAM 그룹 **Testers**에 연결합니다. 해당 그룹의 사용자는 해당 정책의 기본 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

예제 2:이 예제에서는 라는 AWS 관리형 정책을 IAM 그룹 **AdministratorAccess**에 연결합니다**Admins**. 해당 그룹의 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachGroupPolicy를 참조하세요.

Register-IAMRolePolicy

다음 코드 예시는 Register-IAMRolePolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 라는 AWS 관리형 정책을 IAM 역할 **SecurityAudit**에 연결합니다**CoSecurityAuditors**. 해당 역할을 수임하는 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachRolePolicy를 참조하세요.

`Register-IAMUserPolicy`

다음 코드 예시는 `Register-IAMUserPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 라는 AWS 관리형 정책을 IAM 사용자 **AmazonCognitoPowerUser**에 연결합니다**Bob**. 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AttachUserPolicy를 참조하세요.

`Remove-IAMAccessKey`

다음 코드 예시는 `Remove-IAMAccessKey`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 이름이 인 사용자**AKIAIOSFODNN7EXAMPLE**로부터 키 ID가 인 AWS 액세스 키 페어를 삭제합니다**Bob**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAccessKey를 참조하세요.

`Remove-IAMAccountAlias`

다음 코드 예시는 `Remove-IAMAccountAlias`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제는에서 계정 별칭을 제거합니다 AWS 계정. https://mycompanyaws.signin.aws.amazon.com/console에서 별칭을 사용하는 사용자 로그인 페이지가 더

이상 작동하지 않습니다. 대신 https://<accountidnumber>.signin.aws.amazon.com/console AWS 계정 ID 번호와 함께 원래 URL을 사용해야 합니다.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAccountAlias를 참조하세요.

`Remove-IAMAccountPasswordPolicy`

다음 코드 예시는 `Remove-IAMAccountPasswordPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는에 대한 암호 정책을 삭제 AWS 계정 하고 모든 값을 원래 기본값으로 재설정합니다. 암호 정책이 현재 존재하지 않는 경우 다음과 같은 오류 메시지가 나타납니다. PasswordPolicy라는 계정 정책을 찾을 수 없습니다.

```
Remove-IAMAccountPasswordPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAccountPasswordPolicy를 참조하세요.

`Remove-IAMClientIDFromOpenIDConnectProvider`

다음 코드 예시는 `Remove-IAMClientIDFromOpenIDConnectProvider`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`**인 IAM OIDC 제공업체와 연결된 클라이언트 ID 목록에서 클라이언트 ID **`My-TestApp-3`**을 제거합니다.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveClientIdFromOpenIdConnectProvider를 참조하세요.

Remove-IAMGroup

다음 코드 예시는 Remove-IAMGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **MyTestGroup**이라는 IAM 그룹을 삭제합니다. 첫 번째 명령은 그룹 멤버인 IAM 사용자를 모두 제거하고, 두 번째 명령은 IAM 그룹을 삭제합니다. 두 명령 모두 확인 프롬프트 없이 작동합니다.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteGroup을 참조하세요.

Remove-IAMGroupPolicy

다음 코드 예시는 Remove-IAMGroupPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 그룹 **Testers**에서 **TesterPolicy**라는 인라인 정책을 제거합니다. 해당 그룹의 사용자는 해당 정책에 정의된 권한을 즉시 잃게 됩니다.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteGroupPolicy를 참조하세요.

Remove-IAMInstanceProfile

다음 코드 예시는 Remove-IAMInstanceProfile의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **MyAppInstanceProfile**이라는 EC2 인스턴스 프로파일을 삭제합니다. 첫 번째 명령은 인스턴스 프로파일에서 모든 역할을 분리하고, 두 번째 명령은 인스턴스 프로파일을 삭제합니다.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteInstanceProfile을 참조하세요.

`Remove-IAMLoginProfile`

다음 코드 예시는 `Remove-IAMLoginProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`Bob`**이라는 IAM 사용자로부터 로그인 프로파일을 삭제합니다. 이렇게 하면 사용자가 AWS 콘솔에 로그인할 수 없습니다. 이는 사용자가 여전히 사용자 계정에 연결될 수 있는 AWS 액세스 키를 사용하여 AWS CLI, PowerShell 또는 API 호출을 실행하는 것을 방지하지 않습니다.

```
Remove-IAMLoginProfile -UserName Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLoginProfile을 참조하세요.

`Remove-IAMOpenIDConnectProvider`

다음 코드 예시는 `Remove-IAMOpenIDConnectProvider`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 제공업체 **`example.oidcprovider.com`**에 연결되는 IAM OIDC 제공업체를 삭제합니다. 역할 신뢰 정책의 **`Principal`** 요소에서 이 제공업체를 참조하는 모든 역할을 업데이트하거나 삭제해야 합니다.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteOpenIdConnectProvider를 참조하세요.

Remove-IAMPolicy

다음 코드 예시는 Remove-IAMPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MySamplePolicy**인 정책을 삭제합니다. 정책을 삭제하려면 먼저 **Remove-IAMPolicyVersion**을 실행하여 기본값을 제외한 모든 버전을 삭제해야 합니다. 또한 모든 IAM 사용자, 그룹 또는 역할에서 정책을 분리해야 합니다.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

예제 2: 이 예제는 먼저 기본이 아닌 모든 정책 버전을 삭제하고 연결된 모든 IAM 엔티티에서 분리한 다음 마지막으로 정책 자체를 삭제하여 정책을 삭제합니다. 줄 1은 정책 객체를 검색합니다. 줄 2는 기본값으로 플래그가 지정되지 않은 모든 정책 버전을 컬렉션으로 검색한 다음 컬렉션의 각 정책을 삭제합니다. 줄 3은 정책이 연결된 모든 IAM 사용자, 그룹 및 역할을 검색합니다. 줄 4~6은 연결된 각 엔터티에서 정책을 분리합니다. 마지막 줄은 이 명령을 사용하여 관리형 정책과 나머지 기본 버전을 제거합니다. 이 예제에는 확인 프롬프트를 표시하지 않는 데 필요한 모든 줄에 **-Force** 스위치 파라미터가 포함되어 있습니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePolicy를 참조하세요.

Remove-IAMPolicyVersion

다음 코드 예시는 Remove-IAMPolicyVersion의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MySamplePolicy**인 정책에서 **v2**로 식별된 버전을 삭제합니다.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

예제 2: 이 예제는 먼저 기본이 아닌 모든 정책 버전을 삭제한 다음 정책 자체를 삭제하여 정책을 삭제합니다. 줄 1은 정책 객체를 검색합니다. 줄 2는 기본값으로 플래그가 지정되지 않은 모든 정책 버전을 컬렉션으로 검색한 다음 이 명령을 사용하여 컬렉션의 각 정책을 삭제합니다. 마지막 줄은 나머지 기본 버전뿐만 아니라 정책 자체도 제거합니다. 관리형 정책을 성공적으로 삭제하려면 **Unregister-IAMUserPolicy**, **Unregister-IAMGroupPolicy** 및 **Unregister-IAMRolePolicy** 명령을 사용하여 모든 사용자, 그룹 또는 역할에서 정책을 분리해야 합니다. **Remove-IAMPolicy** cmdlet의 예를 참조하세요.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePolicyVersion을 참조하세요.

Remove-IAMRole

다음 코드 예시는 Remove-IAMRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 IAM 계정에서 **MyNewRole**이라는 역할을 삭제합니다. 역할을 삭제하려면 먼저 **Unregister-IAMRolePolicy** 명령을 사용하여 관리형 정책을 분리해야 합니다. 인라인 정책은 역할과 함께 삭제됩니다.

```
Remove-IAMRole -RoleName MyNewRole
```

예제 2: 이 예제는 **MyNewRole**이라는 역할에서 관리형 정책을 분리한 다음 역할을 삭제합니다. 줄 1은 역할에 연결된 모든 관리형 정책을 컬렉션으로 검색한 다음 컬렉션의 각 정책을 역할에서 분리합니다. 줄 2는 역할 자체를 삭제합니다. 인라인 정책은 역할과 함께 삭제됩니다.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRole을 참조하세요.

`Remove-IAMRoleFromInstanceProfile`

다음 코드 예시는 `Remove-IAMRoleFromInstanceProfile`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`MyNewRole`**이라는 EC2 인스턴스 프로파일에서 **`MyNewRole`**이라는 역할을 삭제합니다. IAM 콘솔에서 생성되는 인스턴스 프로파일은 이 예제와 같이 항상 역할과 동일한 이름을 갖습니다. API 또는 CLI에서 생성하는 경우 서로 다른 이름을 가질 수 있습니다.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveRoleFromInstanceProfile을 참조하세요.

`Remove-IAMRolePermissionsBoundary`

다음 코드 예시는 `Remove-IAMRolePermissionsBoundary`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 역할에 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRolePermissionsBoundary를 참조하세요.

`Remove-IAMRolePolicy`

다음 코드 예시는 `Remove-IAMRolePolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 역할 **`S3BackupRole`**에 포함된 인라인 정책 **`S3AccessPolicy`**를 삭제합니다.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteRolePolicy를 참조하세요.

`Remove-IAMRoleTag`

다음 코드 예시는 `Remove-IAMRoleTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 태그 키가 'abac'인 'MyRoleName' 역할에서 태그를 제거합니다. 여러 태그를 제거하려면 쉼표로 구분된 태그 키 목록을 제공합니다.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagRole을 참조하세요.

`Remove-IAMSAMLProvider`

다음 코드 예시는 `Remove-IAMSAMLProvider`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**인 IAM SAML 2.0 제공업체를 삭제합니다.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSAMLProvider를 참조하세요.

`Remove-IAMServerCertificate`

다음 코드 예시는 `Remove-IAMServerCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`MyServerCert`**라는 서버 인증서를 삭제합니다.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteServerCertificate를 참조하세요.

`Remove-IAMServiceLinkedRole`

다음 코드 예시는 `Remove-IAMServiceLinkedRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 서비스 연결 역할을 삭제했습니다. 서비스에서 여전히 이 역할을 사용하고 있는 경우 이 명령을 실행하면 실패합니다.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteServiceLinkedRole을 참조하세요.

`Remove-IAMSigningCertificate`

다음 코드 예시는 `Remove-IAMSigningCertificate`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Bob**이라는 IAM 사용자로부터 ID가 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**인 서명 인증서를 삭제합니다.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteSigningCertificate를 참조하세요.

`Remove-IAMUser`

다음 코드 예시는 `Remove-IAMUser`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Bob**이라는 IAM 사용자를 삭제합니다.

```
Remove-IAMUser -UserName Bob
```

예제 2: 이 예제는 먼저 삭제해야 하는 모든 요소와 함께 **Theresa**라는 IAM 사용자를 삭제합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteUser를 참조하세요.

Remove-IAMUserFromGroup

다음 코드 예시는 Remove-IAMUserFromGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 그룹 **Testers**에서 IAM 사용자 **Bob**을 제거합니다.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

예제 2: 이 예제는 IAM 사용자 **Theresa**가 구성원으로 속한 그룹을 찾은 다음 해당 그룹에서 **Theresa**를 제거합니다.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

예제 3: 이 예제는 **Testers** 그룹에서 IAM 사용자 **Bob**을 제거하는 다른 방법을 보여줍니다.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveUserFromGroup을 참조하세요.

Remove-IAMUserPermissionsBoundary

다음 코드 예시는 Remove-IAMUserPermissionsBoundary의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자에게 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteUserPermissionsBoundary를 참조하세요.

`Remove-IAMUserPolicy`

다음 코드 예시는 `Remove-IAMUserPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **`Bob`**이라는 IAM 사용자에게 포함된 **`AccessToEC2Policy`**라는 인라인 정책을 삭제합니다.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

예제 2: 이 예제는 **`Theresa`**라는 IAM 사용자에게 포함된 모든 인라인 정책을 찾아 삭제합니다.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteUserPolicy를 참조하세요.

`Remove-IAMUserTag`

다음 코드 예시는 `Remove-IAMUserTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 태그 키가 'abac' 및 'xyzw'인 'joe'라는 사용자에게서 태그를 제거합니다. 여러 태그를 제거하려면 쉼표로 구분된 태그 키 목록을 제공합니다.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagUser를 참조하세요.

Remove-IAMVirtualMFADevice

다음 코드 예시는 Remove-IAMVirtualMFADevice의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:mfa/bob**인 IAM 가상 MFA 디바이스를 삭제합니다.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

예제 2: 이 예제는 IAM 사용자 Theresa에게 MFA 디바이스가 할당되었는지 확인합니다. 발견된 디바이스는 IAM 사용자에 대해 비활성화됩니다. 디바이스가 가상이면 삭제됩니다.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteVirtualMfaDevice를 참조하세요.

Request-IAMCredentialReport

다음 코드 예시는 Request-IAMCredentialReport의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 4시간마다 수행할 수 있는 새 보고서 생성을 요청합니다. 마지막 보고서가 아직 최신인 경우 State 필드에 **COMPLETE**가 표시됩니다. 완성된 보고서를 보려면 **Get-IAMCredentialReport**를 사용합니다.

```
Request-IAMCredentialReport
```

출력:

```
Description                                                State
```

```
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GenerateCredentialReport를 참조하세요.

`Request-IAMServiceLastAccessedDetail`

다음 코드 예시는 `Request-IAMServiceLastAccessedDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 GenerateServiceLastAccessedDetails API와 동등한 cmdlet입니다. 이는 Get-IAMServiceLastAccessedDetail과 Get-IAMServiceLastAccessedDetailWithEntity에서 사용할 수 있는 작업 ID를 제공합니다.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GenerateServiceLastAccessedDetails를 참조하세요.

`Set-IAMDefaultPolicyVersion`

다음 코드 예시는 `Set-IAMDefaultPolicyVersion`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **`arn:aws:iam::123456789012:policy/MyPolicy`**인 정책의 **`v2`** 버전을 기본 활성 버전으로 설정합니다.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetDefaultPolicyVersion을 참조하세요.

`Set-IAMRolePermissionsBoundary`

다음 코드 예시는 `Set-IAMRolePermissionsBoundary`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 역할의 권한 경계를 설정하는 방법을 보여줍니다. AWS 관리형 정책 또는 사용자 지정 정책을 권한 경계로 설정할 수 있습니다.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutRolePermissionsBoundary를 참조하세요.

`Set-IAMUserPermissionsBoundary`

다음 코드 예시는 `Set-IAMUserPermissionsBoundary`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 사용자의 권한 경계를 설정하는 방법을 보여줍니다. AWS 관리형 정책 또는 사용자 지정 정책을 권한 경계로 설정할 수 있습니다.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutUserPermissionsBoundary를 참조하세요.

`Sync-IAMMFADevice`

다음 코드 예시는 `Sync-IAMMFADevice`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자 **Bob**과 연결되어 있고 ARN이 **arn:aws:iam::123456789012:mfa/bob**인 MFA 디바이스를 두 개의 인증 코드를 제공한 인증 프로그램과 동기화합니다.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

예제 2: 이 예제는 IAM 사용자 **Theresa**와 연결된 IAM MFA 디바이스를 일련 번호가 **ABCD12345678**이고 두 개의 인증 코드를 제공한 물리적 디바이스와 동기화합니다.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResyncMfaDevice를 참조하세요.

Unregister-IAMGroupPolicy

다음 코드 예시는 Unregister-IAMGroupPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/TesterAccessPolicy**인 관리형 그룹 정책을 **Testers**라는 그룹에서 분리합니다.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

예제 2: 이 예제는 **Testers**라는 그룹에 연결된 모든 관리형 정책을 찾아 그룹에서 분리합니다.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachGroupPolicy를 참조하세요.

Unregister-IAMRolePolicy

다음 코드 예시는 Unregister-IAMRolePolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy**인 관리형 그룹 정책을 **FedTesterRole**라는 역할에서 분리합니다.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

예제 2: 이 예제는 **FedTesterRole**라는 역할에 연결된 모든 관리형 정책을 찾아 역할에서 분리합니다.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachRolePolicy를 참조하세요.

Unregister-IAMUserPolicy

다음 코드 예시는 Unregister-IAMUserPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/TesterPolicy**인 관리형 정책을 **Bob**라는 IAM 사용자에게서 분리합니다.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

예제 2: 이 예제는 **Theresa**라는 IAM 사용자에게 연결된 모든 관리형 정책을 찾아 사용자에게서 분리합니다.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DetachUserPolicy를 참조하세요.

Update-IAMAccessKey

다음 코드 예시는 Update-IAMAccessKey의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Bob**이라는 IAM 사용자에 대한 액세스 키 **AKIAIOSFODNN7EXAMPLE**의 상태를 **Inactive**로 변경합니다.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAccessKey를 참조하세요.

Update-IAMAccountPasswordPolicy

다음 코드 예시는 Update-IAMAccountPasswordPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 계정의 암호 정책을 지정된 설정으로 업데이트합니다. 단, 명령에 포함되지 않은 파라미터는 수정되지 않은 채로 남아 있지 않습니다. 대신 기본값으로 재설정됩니다.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAccountPasswordPolicy를 참조하세요.

Update-IAMAssumeRolePolicy

다음 코드 예시는 Update-IAMAssumeRolePolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 이름이 **ClientRole**인 IAM 역할을 새 신뢰 정책으로 업데이트하고, 그 내용은 **ClientRolePolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAssumeRolePolicy를 참조하세요.

Update-IAMGroup

다음 코드 예시는 Update-IAMGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 그룹 이름을 **Testers**에서 **AppTesters**로 변경합니다.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

예제 2: 이 예제는 IAM 그룹 **AppTesters**의 경로를 **/Org1/Org2/**로 변경합니다. 그러면 그룹의 ARN이 **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**로 변경됩니다.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateGroup을 참조하세요.

Update-IAMLoginProfile

다음 코드 예시는 Update-IAMLoginProfile의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자 **Bob**에 대한 새 임시 암호를 설정하고 사용자가 다음에 로그인할 때 암호를 변경하도록 요구합니다.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateLoginProfile을 참조하세요.

Update-IAMOpenIDConnectProviderThumbprint

다음 코드 예시는 Update-IAMOpenIDConnectProviderThumbprint의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**인 OIDC 제공업체에 대한 인증서 지문 목록을 업데이트하여 새 지문을 사용합니다. OIDC 제공업체는 제공업체와 연결된 인증서가 변경될 때 새 값을 공유합니다.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateOpenIdConnectProviderThumbprint를 참조하세요.

Update-IAMRole

다음 코드 예시는 Update-IAMRole의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 역할 설명과 역할 세션을 요청할 수 있는 최대 세션 기간 값 (초)을 업데이트합니다.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateRole을 참조하세요.

Update-IAMRoleDescription

다음 코드 예시는 Update-IAMRoleDescription의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 계정의 IAM 역할 설명을 업데이트합니다.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateRoleDescription을 참조하세요.

Update-IAMSAMLProvider

다음 코드 예시는 Update-IAMSAMLProvider의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:saml-provider/SAMLADFS**인 IAM의 SAML 제공업체를 **SAMLMetaData.xml** 파일의 새 SAML 메타데이터 문서로 업데이트합니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateSamlProvider를 참조하세요.

Update-IAMServerCertificate

다음 코드 예시는 Update-IAMServerCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 인증서 이름을 **MyServerCertificate**에서 **MyRenamedServerCertificate**로 변경합니다.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

예제 2: 이 예제는 **MyServerCertificate**라는 인증서를 /Org1/Org2/ 경로로 이동합니다. 그러면 리소스의 ARN이 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**로 변경됩니다.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateServerCertificate를 참조하세요.

Update-IAMSigningCertificate

다음 코드 예시는 Update-IAMSigningCertificate의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **Bob**이라는 IAM 사용자와 연결되어 있고 인증서 ID가 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**인 인증서를 업데이트하여 비활성으로 표시합니다.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateSigningCertificate를 참조하세요.

Update-IAMUser

다음 코드 예시는 Update-IAMUser의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 IAM 사용자 이름을 **Bob**에서 **Robert**로 변경합니다.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

예제 2: 이 예제는 IAM 사용자 **Bob**의 경로를 **/Org1/Org2/**로 변경합니다. 그러면 해당 사용자의 ARN이 **arn:aws:iam::123456789012:user/Org1/Org2/bob**으로 효과적으로 변경됩니다.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateUser를 참조하세요.

Write-IAMGroupPolicy

다음 코드 예시는 Write-IAMGroupPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **AppTesterPolicy**라는 인라인 정책을 생성하고 이를 IAM 그룹 **AppTesters**에 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁

니다. JSON 정책 내용은 **apptesterpolicy.json** 파일로 제공됩니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutGroupPolicy를 참조하세요.

Write-IAMRolePolicy

다음 코드 예시는 Write-IAMRolePolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **FedTesterRolePolicy**라는 인라인 정책을 생성하고 이를 IAM 역할 **FedTesterRole**에 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. JSON 정책 내용은 **FedTesterPolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutRolePolicy를 참조하세요.

Write-IAMUserPolicy

다음 코드 예시는 Write-IAMUserPolicy의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 **EC2AccessPolicy**라는 인라인 정책을 생성하고 이를 IAM 사용자 **Bob**에게 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. JSON 정책 내용은 **EC2AccessPolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutUserPolicy를 참조하세요.

Tools for PowerShell V4를 사용한 Kinesis 예제

다음 코드 예제에서는 Kinesis와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-KINRecord`

다음 코드 예시는 `Get-KINRecord`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 일련의 하나 이상 레코드에서 데이터를 반환 및 추출하는 방법을 보여줍니다. Get-KINRecord에 제공되는 반복자는 반환할 레코드의 시작 위치를 결정하며, 이 예제에서 이들 레코드,는 변수인 $records에 캡처됩니다. 그런 다음 $records 컬렉션을 인덱싱하여 개별 레코드에 액세스할 수 있습니다. 레코드의 데이터가 UTF-8 인코딩된 텍스트라고 가정하면 최종 명령은 객체의 MemoryStream에서 데이터를 추출하여 콘솔에 텍스트로 반환하는 방법을 보여줍니다.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

출력:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

출력:

```
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())
```

출력:

```
test data from string
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetRecords를 참조하세요.

`Get-KINShardIterator`

다음 코드 예시는 `Get-KINShardIterator`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 샤드 및 시작 위치에 대한 샤드 반복자를 반환합니다. 샤드 식별자 및 시퀀스 번호에 대한 세부 정보는 Get-KINStream cmdlet의 출력에서 반환된 스트림 객체의 샤드 컬렉션을 참조하여 얻을 수 있습니다. 반환된 반복자를 Get-KINRecord cmdlet과 함께 사용하여 샤드의 데이터 레코드를 가져올 수 있습니다.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

출력:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetShardIterator를 참조하세요.

`Get-KINStream`

다음 코드 예시는 `Get-KINStream`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 스트림의 세부 정보를 반환합니다.

```
Get-KINStream -StreamName "mystream"
```

출력:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeStream을 참조하세요.

`New-KINStream`

다음 코드 예시는 `New-KINStream`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 새 스트림을 생성합니다. 기본적으로 이 cmdlet은 출력을 반환하지 않으므로 이후에 사용할 수 있게 -StreamName 파라미터에 제공된 값을 반환하기 위해 -PassThru 스위치가 추가됩니다.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateStream을 참조하세요.

`Remove-KINStream`

다음 코드 예시는 `Remove-KINStream`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 지정된 스트림을 삭제합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 사용합니다.

```
Remove-KINStream -StreamName "mystream"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteStream을 참조하세요.

`Write-KINRecord`

다음 코드 예시는 `Write-KINRecord`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: -Text 파라미터에 제공된 문자열이 포함된 레코드를 씁니다.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

예제 2: 지정된 파일에 포함된 데이터가 포함된 레코드를 씁니다. 이 파일은 바이트 시퀀스로 취급되므로 텍스트가 포함된 경우 이 cmdlet과 함께 사용하기 전에 필요한 인코딩을 사용하여 작성해야 합니다.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutRecord를 참조하세요.

Tools for PowerShell V4를 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-LMResourceTag`

다음 코드 예시는 `Add-LMResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 3개의 태그(워싱턴, 오리건, 캘리포니아)와 관련 값을 ARN으로 식별되는 지정된 함수에 추가합니다.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResource를 참조하세요.

`Get-LMAccountSetting`

다음 코드 예시는 `Get-LMAccountSetting`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 계정 제한과 계정 사용량을 비교하기 위해 표시됩니다.

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

출력:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccountSettings를 참조하세요.

`Get-LMAlias`

다음 코드 예시는 `Get-LMAlias`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 특정 Lambda 함수 별칭에 대한 라우팅 구성 가중치를 검색합니다.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

출력:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAlias를 참조하세요.

`Get-LMFunctionConcurrency`

다음 코드 예시는 `Get-LMFunctionConcurrency`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수에 대한 예약된 동시성을 가져옵니다.

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

출력:

```
ReservedConcurrentExecutions
----------------------------
100
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetFunctionConcurrency를 참조하세요.

`Get-LMFunctionConfiguration`

다음 코드 예시는 `Get-LMFunctionConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수의 버전별 구성을 반환합니다.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetFunctionConfiguration을 참조하세요.

`Get-LMFunctionList`

다음 코드 예시는 `Get-LMFunctionList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 정렬된 코드 크기로 모든 Lambda 함수를 표시합니다.

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

출력:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
 243
MylambdaFunction123                                             python3.8     600
 659
myfuncpython1                                                   python3.8     303
 675
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListFunctions를 참조하세요.

`Get-LMPolicy`

다음 코드 예시는 `Get-LMPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플은 Lambda 함수의 함수 정책을 표시합니다.

```
Get-LMPolicy -FunctionName test -Select Policy
```

출력:

```
{"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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPolicy를 참조하세요.

Get-LMProvisionedConcurrencyConfig

다음 코드 예시는 Get-LMProvisionedConcurrencyConfig의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수의 지정된 별칭에 대해 프로비저닝된 동시성 구성을 가져옵니다.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

출력:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetProvisionedConcurrencyConfig를 참조하세요.

Get-LMProvisionedConcurrencyConfigList

다음 코드 예시는 Get-LMProvisionedConcurrencyConfigList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수에 대해 프로비저닝된 동시성 구성 목록을 검색합니다.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListProvisionedConcurrencyConfigs를 참조하세요.

Get-LMResourceTag

다음 코드 예시는 Get-LMResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 함수에 현재 설정된 태그와 해당 값을 검색합니다.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

출력:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTags를 참조하세요.

`Get-LMVersionsByFunction`

다음 코드 예시는 `Get-LMVersionsByFunction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수의 각 버전에 대한 버전별 구성 목록을 반환합니다.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListVersionsByFunction을 참조하세요.

New-LMAlias

다음 코드 예시는 New-LMAlias의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 버전 및 라우팅 구성에 대한 새 Lambda 별칭을 생성하여 수신하는 간접 호출 요청의 비율을 지정합니다.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAlias를 참조하세요.

Publish-LMFunction

다음 코드 예시는 Publish-LMFunction의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 AWS Lambda에서 MyFunction이라는 새 C#(dotnetcore1.0 런타임) 함수를 생성하여 로컬 파일 시스템의 zip 파일에서 함수에 대해 컴파일된 바이너리를 제공합니다(상대 경로 또는 절대 경로를 사용할 수 있음). C# Lambda 함수는 AssemblyName::Namespace.ClassName::MethodName 지정을 사용하여 함수에 대한 핸들러를 지정합니다. 핸들러 사양의 어셈블리 이름 (.dll 접미사 제외), 네임스페이스, 클래스 이름, 메서드 이름 부분을 적절하게 바꿔야 합니다. 새 함수에는 제공된 값으로부터 환경 변수 'envvar1' 및 'envvar2'가 설정됩니다.

```
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" }
```

출력:

```
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        :
```

예제 2: 이 예제는 함수 바이너리가 먼저 Amazon S3 버킷(의도한 Lambda 함수와 동일한 리전에 있어야 함)에 업로드되고 함수 생성 시 결과 S3 객체가 참조된다는 점을 제외하면 이전 예제와 유사합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateFunction을 참조하세요.

`Publish-LMVersion`

다음 코드 예시는 `Publish-LMVersion`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수 코드의 기존 스냅샷에 대한 버전을 생성합니다.

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PublishVersion을 참조하세요.

`Remove-LMAlias`

다음 코드 예시는 `Remove-LMAlias`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 명령에 언급된 Lambda 함수 별칭을 삭제합니다.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAlias를 참조하세요.

`Remove-LMFunction`

다음 코드 예시는 `Remove-LMFunction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 특정 버전의 Lambda 함수를 삭제합니다.

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFunction를 참조하세요.

`Remove-LMFunctionConcurrency`

다음 코드 예시는 `Remove-LMFunctionConcurrency`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수의 함수 동시성을 제거합니다.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteFunctionConcurrency를 참조하세요.

`Remove-LMPermission`

다음 코드 예시는 `Remove-LMPermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Lambda 함수의 지정된 StatementId에 대한 함수 정책을 제거합니다.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemovePermission을 참조하세요.

`Remove-LMProvisionedConcurrencyConfig`

다음 코드 예시는 `Remove-LMProvisionedConcurrencyConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 특정 별칭에 대한 프로비저닝된 동시성 구성을 제거합니다.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteProvisionedConcurrencyConfig를 참조하세요.

`Remove-LMResourceTag`

다음 코드 예시는 `Remove-LMResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 함수에서 제공된 태그를 제거합니다. -Force 스위치가 지정되지 않은 경우 cmdlet은 진행하기 전에 확인 프롬프트를 표시합니다. 태그를 제거하려면 서비스를 한 번만 직접적으로 호출하면 됩니다.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

예제 2: 함수에서 제공된 태그를 제거합니다. -Force 스위치가 지정되지 않은 경우 cmdlet은 진행하기 전에 확인 프롬프트를 표시합니다. 제공된 태그별로 서비스에 대한 호출은 한 번만 이루어집니다.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagResource를 참조하세요.

`Update-LMAlias`

다음 코드 예시는 `Update-LMAlias`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기존 Lambda 함수 별칭의 구성을 업데이트합니다. 트래픽의 60%(0.6)를 버전 1로 이동하도록 RoutingConfiguration 값을 업데이트합니다.

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAlias를 참조하세요.

`Update-LMFunctionCode`

다음 코드 예시는 `Update-LMFunctionCode`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 zip 파일에 포함된 새 콘텐츠로 'MyFunction'이라는 함수를 업데이트합니다. C# .NET Core Lambda 함수의 경우 zip 파일에는 컴파일된 어셈블리가 포함되어야 합니다.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

예제 2: 이 예제는 이전 예제와 유사하지만 업데이트된 코드가 포함된 Amazon S3 객체를 사용하여 함수를 업데이트합니다.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key
 UpdatedCode.zip
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateFunctionCode를 참조하세요.

`Update-LMFunctionConfiguration`

다음 코드 예시는 `Update-LMFunctionConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기존 Lambda 함수의 구성을 업데이트합니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateFunctionConfiguration을 참조하세요.

`Write-LMFunctionConcurrency`

다음 코드 예시는 `Write-LMFunctionConcurrency`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 함수 전체에 대해 동시성 설정을 적용합니다.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutFunctionConcurrency를 참조하세요.

`Write-LMProvisionedConcurrencyConfig`

다음 코드 예시는 `Write-LMProvisionedConcurrencyConfig`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 함수의 별칭에 프로비저닝된 동시성 구성을 추가합니다.

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutProvisionedConcurrencyConfig를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon ML 예제

다음 코드 예제에서는 Amazon ML에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Get-MLBatchPrediction`

다음 코드 예시는 `Get-MLBatchPrediction`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 배치 예측에 대한 세부 메타데이터를 반환합니다.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBatchPrediction을 참조하세요.

`Get-MLBatchPredictionList`

다음 코드 예시는 `Get-MLBatchPredictionList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 요청에 지정된 검색 기준과 일치하는 모든 BatchPredictions 및 관련 데이터 레코드의 목록을 반환합니다.

```
Get-MLBatchPredictionList
```

예제 2: 상태가 COMPLETED인 모든 BatchPredictions 목록을 반환합니다.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeBatchPredictions을 참조하세요.

`Get-MLDataSource`

다음 코드 예시는 `Get-MLDataSource`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 DataSource의 메타데이터, 상태, 데이터 파일 정보를 반환합니다.

```
Get-MLDataSource -DataSourceId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDataSource를 참조하세요.

`Get-MLDataSourceList`

다음 코드 예시는 `Get-MLDataSourceList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 DataSources 및 관련 데이터 레코드의 목록을 반환합니다.

```
Get-MLDataSourceList
```

예제 2: 상태가 COMPLETED인 모든 DataSources 목록을 반환합니다.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDataSources를 참조하세요.

`Get-MLEvaluation`

다음 코드 예시는 `Get-MLEvaluation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 Evaluation에 대한 메타데이터 및 상태를 반환합니다.

```
Get-MLEvaluation -EvaluationId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetEvaluation을 참조하세요.

`Get-MLEvaluationList`

다음 코드 예시는 `Get-MLEvaluationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 Evaluation 리소스 목록을 반환합니다.

```
Get-MLEvaluationList
```

예제 2: 상태가 COMPLETED인 모든 Evaluation 목록을 반환합니다.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEvaluations을 참조하세요.

`Get-MLModel`

다음 코드 예시는 `Get-MLModel`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 MLModel에 대한 세부 메타데이터, 상태, 스키마, 데이터 파일 정보를 반환합니다.

```
Get-MLModel -ModelId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetMLModel을 참조하세요.

`Get-MLModelList`

다음 코드 예시는 `Get-MLModelList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 모델 및 관련 데이터 레코드의 목록을 반환합니다.

```
Get-MLModelList
```

예제 2: 상태가 COMPLETED인 모든 모델의 목록을 반환합니다.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMLModels을 참조하세요.

Get-MLPrediction

다음 코드 예시는 Get-MLPrediction의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 모델의 실시간 예측 엔드포인트 URL로 레코드를 전송합니다.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Predict를 참조하세요.

New-MLBatchPrediction

다음 코드 예시는 New-MLBatchPrediction의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID가 ID인 모델에 대한 새 배치 예측 요청을 생성하고 출력을 지정된 S3 위치에 배치합니다.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateBatchPrediction을 참조하세요.

New-MLDataSourceFromS3

다음 코드 예시는 New-MLDataSourceFromS3의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 NAME이고 스키마가 SCHEMA인 S3 위치의 데이터를 사용하여 데이터 소스를 생성합니다.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDataSourceFromS3를 참조하세요.

New-MLEvaluation

다음 코드 예시는 New-MLEvaluation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 데이터 소스 ID 및 모델 ID에 대한 평가를 생성합니다.

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateEvaluation을 참조하세요.

New-MLModel

다음 코드 예시는 New-MLModel의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 학습 데이터로 새 모델을 생성합니다.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateMLModel을 참조하세요.

New-MLRealtimeEndpoint

다음 코드 예시는 New-MLRealtimeEndpoint의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 모델 ID에 대한 새 실시간 예측 엔드포인트를 생성합니다.

```
New-MLRealtimeEndpoint -ModelId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateRealtimeEndpoint를 참조하세요.

Tools for PowerShell V4를 사용한 Macie 예제

다음 코드 예제에서는 Macie에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-MAC2FindingList`

다음 코드 예시는 `Get-MAC2FindingList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 'CREDIT_CARD_NUMBER' 또는 'US_SOCIAL_SECURITY_NUMBER' 유형의 민감한 데이터 감지가 포함된 조사 결과에 대한 FindingIds 목록을 반환합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListFindings을 참조하세요.

AWS 가격표 Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS 가격표.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-PLSAttributeValue`

다음 코드 예시는 `Get-PLSAttributeValue`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: us-east-1 리전의 Amazon EC2에 대한 'volumeType' 속성의 값을 반환합니다.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

출력:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAttributeValues를 참조하세요.

`Get-PLSProduct`

다음 코드 예시는 `Get-PLSProduct`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Amazon EC2의 모든 제품에 대한 세부 정보를 반환합니다.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

출력:

```
{"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"},...
```

예제 2: SSD 지원 'General Purpose' 볼륨 유형으로 필터링된 us-east-1 리전의 Amazon EC2에 대한 데이터를 반환합니다.

```
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
```

출력:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetProducts를 참조하세요.

`Get-PLSService`

다음 코드 예시는 `Get-PLSService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: us-east-1 리전에서 사용 가능한 모든 서비스 코드의 메타데이터를 반환합니다.

```
Get-PLSService -Region us-east-1
```

출력:

```
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
...
```

예제 2: us-east-1 리전의 Amazon EC2 서비스에 대한 메타데이터를 반환합니다.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

출력:

```
AttributeNames                                                          ServiceCode
--------------                                                          -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeServices를 참조하세요.

Tools for PowerShell V4를 사용한 Resource Groups 예제

다음 코드 예제에서는 Resource Groups와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

Add-RGResourceTag

다음 코드 예시는 Add-RGResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 태그 키 'Instances'와 값 'workboxes'를 지정된 리소스 그룹 ARN에 추가합니다.

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

출력:

```
Arn                                                              Tags
---                                                              ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Tag를 참조하세요.

Find-RGResource

다음 코드 예시는 Find-RGResource의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 태그 필터를 사용하여 인스턴스 리소스 유형에 대한 ResourceQuery를 생성하고 리소스를 찾습니다.

```
$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
```

출력:

```
ResourceArn                                                     ResourceType
-----------                                                     ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SearchResources를 참조하세요.

Get-RGGroup

다음 코드 예시는 Get-RGGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 그룹 이름에 따라 리소스 그룹을 검색합니다.

```
Get-RGGroup -GroupName auto-no
```

출력:

```
Description GroupArn                                                Name
----------- --------                                                ----
```

```
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetGroup을 참조하세요.

`Get-RGGroupList`

다음 코드 예시는 `Get-RGGroupList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 이미 생성된 리소스 그룹을 나열합니다.

```
Get-RGGroupList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListGroups를 참조하세요.

`Get-RGGroupQuery`

다음 코드 예시는 `Get-RGGroupQuery`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스 그룹에 대한 리소스 쿼리를 가져옵니다.

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

출력:

```
Query
         Type
-----
         ----
```

```
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetGroupQuery를 참조하세요.

Get-RGGroupResourceList

다음 코드 예시는 Get-RGGroupResourceList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리소스 유형별 필터링을 기준으로 그룹 리소스를 나열합니다.

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListGroupResources를 참조하세요.

Get-RGResourceTag

다음 코드 예시는 Get-RGResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스 그룹 ARN에 대한 태그를 나열합니다.

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

출력:

```
Key       Value
---       -----
Instances workboxes
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetTags를 참조하세요.

New-RGGroup

다음 코드 예시는 New-RGGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 TestPowerShellGroup이라는 새 태그 기반 AWS Resource Groups 리소스 그룹을 생성합니다. 그룹에는 키가 'Name'이고 값이 'test2'인 태그가 지정된 현재 리전의 Amazon EC2 인스턴스가 포함됩니다. 명령은 쿼리와 그룹 유형, 작업 결과를 반환합니다.

```
$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."
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateGroup을 참조하세요.

Remove-RGGroup

다음 코드 예시는 Remove-RGGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 명명된 리소스 그룹을 제거합니다.

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteGroup을 참조하세요.

Remove-RGResourceTag

다음 코드 예시는 Remove-RGResourceTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리소스 그룹에서 언급된 태그를 제거합니다.

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Untag를 참조하세요.

Update-RGGroup

다음 코드 예시는 Update-RGGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 그룹의 설명을 업데이트합니다.

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

출력:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateGroup을 참조하세요.

Update-RGGroupQuery

다음 코드 예시는 Update-RGGroupQuery의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 쿼리 객체를 생성하고 그룹에 대한 쿼리를 업데이트합니다.

```
$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
```

출력:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateGroupQuery를 참조하세요.

Tools for PowerShell V4를 사용한 Resource Groups Tagging API 예제

다음 코드 예제에서는 Resource Groups Tagging API와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-RGTResourceTag`

다음 코드 예시는 `Add-RGTResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 태그 키가 'stage' 및 'version'이고 태그 값이 'beta' 및 'preprod_test'인 태그를 Amazon S3 버킷과 Amazon DynamoDB 테이블에 추가합니다. 태그를 적용하려면 서비스를 한 번만 직접적으로 호출하면 됩니다.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

예제 2: 이 예제에서는 지정된 태그와 값을 Amazon S3 버킷과 Amazon DynamoDB 테이블에 추가합니다. cmdlet에 파이프된 각 리소스 ARN에 대해 한 번씩, 두 번의 직접 호출이 서비스에 대해 이루어집니다.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TagResources를 참조하세요.

`Get-RGTResource`

다음 코드 예시는 `Get-RGTResource`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 리전에서 태그가 지정된 모든 리소스 및 리소스와 연결된 태그 키를 반환합니다. cmdlet에 -Region 파라미터가 제공되지 않으면 cmdlet이 쉘 또는 EC2 인스턴스 메타데이터에서 리전을 추론하려고 시도합니다.

```
Get-RGTResource
```

출력:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

예제 2: 리전에서 태그가 있는 지정된 유형의 모든 리소스를 반환합니다. 각 서비스 이름 및 리소스 유형의 문자열은 리소스의 Amazon 리소스 이름(ARN)에 포함된 문자열과 동일합니다.

```
Get-RGTResource -ResourceType "s3"
```

출력:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

예제 3: 리전에서 태그가 있는 지정된 유형의 모든 리소스를 반환합니다. 리소스 유형이 cmdlet에 파이프될 때 제공된 각 리소스 유형에 대해 서비스를 한 번 직접 호출합니다.

```
"dynamodb","s3" | Get-RGTResource
```

출력:

```
ResourceARN                                                          Tags
-----------                                                          ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable                {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

예제 4: 지정된 필터와 일치하고 태그가 있는 모든 리소스를 반환합니다.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

출력:

```
ResourceARN                                                          Tags
```

```
-----------                                                    ----
arn:aws:s3:::amzn-s3-demo-bucket                                        {stage,
 version, othertag}
```

예제 5: 지정된 필터 및 리소스 유형과 일치하고 태그가 지정된 모든 리소스를 반환합니다.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

출력:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

예제 6: 지정된 필터와 일치하고 태그가 있는 모든 리소스를 반환합니다.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

출력:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetResources를 참조하세요.

Get-RGTTagKey

다음 코드 예시는 Get-RGTTagKey의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 리전의 모든 태그 키를 반환합니다. -Region 파라미터를 지정하지 않으면 cmdlet은 기본 쉘 리전 또는 EC2 인스턴스 메타데이터에서 리전을 추론하려고 시도합니다. 태그 키는 특정 순서로 반환되지 않습니다.

```
Get-RGTTagKey -region us-west-2
```

출력:

```
version
stage
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetTagKeys를 참조하세요.

`Get-RGTTagValue`

다음 코드 예시는 `Get-RGTTagValue`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 리전에서 지정된 태그의 값을 반환합니다. -Region 파라미터를 지정하지 않으면 cmdlet은 기본 쉘 리전 또는 EC2 인스턴스 메타데이터에서 리전을 추론하려고 시도합니다.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

출력:

```
beta
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetTagValues를 참조하세요.

`Remove-RGTResourceTag`

다음 코드 예시는 `Remove-RGTResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Amazon S3 버킷 및 Amazon DynamoDB 테이블에서 태그 키 'stage' 및 'version'과 여기에 연결된 값을 제거합니다. 태그를 제거하려면 서비스를 한 번만 직접적으로 호출하면 됩니다. 태그를 제거하기 전에 cmdlet이 확인 프롬프트를 표시합니다. 확인을 우회하려면 -Force 파라미터를 추가합니다.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

예제 2: Amazon S3 버킷 및 Amazon DynamoDB 테이블에서 태그 키 'stage' 및 'version'과 여기에 연결된 값을 제거합니다. cmdlet에 파이프된 각 리소스 ARN에 대해 한 번씩, 두 번의 직접 호출이

서비스에 대해 이루어집니다. 각 직접 호출이 이루어지기 전에 cmdlet이 확인 프롬프트를 표시합니다. 확인을 우회하려면 -Force 파라미터를 추가합니다.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UntagResources를 참조하세요.

Tools for PowerShell V4를 사용한 Route 53 예제

다음 코드 예제에서는 Route 53에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Edit-R53ResourceRecordSet`

다음 코드 예시는 `Edit-R53ResourceRecordSet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 www.example.com의 A 레코드를 생성하고 test.example.com의 A 레코드를 192.0.2.3에서 192.0.2.1로 변경합니다. 변경 TXT 유형 레코드의 값은 큰따옴표로 묶여야 합니다. 자세한 내용은 Amazon Route 53 설명서를 참조하세요. Get-R53Change cmdlet을 사용하여 변경 사항이 완료되는 시기를 확인할 수 있습니다.

```
$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
```

예제 2: 이 예제에서는 별칭 리소스 레코드 세트를 생성하는 방법을 보여줍니다. 'Z222222222'는 별칭 리소스 레코드 세트를 생성하는 Amazon Route 53 호스팅 영역의 ID입니다. 'example.com'은 별칭을 생성하려는 zone apex이고 'www.example.com'은 역시 별칭을 생성하려는 하위 도메인입니다. 'Z1111111111111'은 로드 밸런서의 호스팅 영역 ID의 예이고 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com'은 Amazon Route 53가 example.com 및 www.example.com에 대한 쿼리에 응답하는 로드 밸런서 도메인 이름의 예입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하세요. Get-R53Change cmdlet을 사용하여 변경 사항이 완료되는 시기를 확인할 수 있습니다.

```
$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
```

예제 3: 이 예제는 www.example.com에 대한 두 개의 A 레코드를 생성합니다. 1/4(1/(1+3))의 경우 Amazon Route 53는 첫 번째 리소스 레코드 세트(192.0.2.9 및 192.0.2.10)에 대한 두 값으로 www.example.com 쿼리에 응답합니다. 3/4(3/(1+3))의 경우 Amazon Route 53는 두 번째 리소스 레코드 세트(192.0.2.11 및 192.0.2.12)의 두 값으로 www.example.com 쿼리에 응답합니다. 자세한 내용은 Amazon Route 53 설명서를 참조하세요. Get-R53Change cmdlet을 사용하여 변경 사항이 완료되는 시기를 확인할 수 있습니다.

```
$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
```

예제 4: 이 예제는 example.com이 가중치 기반 별칭 리소스 레코드 세트를 생성하려는 도메인이라고 가정하여 가중치 기반 별칭 리소스 레코드 세트를 생성하는 방법을 보여줍니다. SetIdentifier는 두 개의 가중치 기반 별칭 리소스 레코드 세트를 서로 구분합니다. 이름 및 유형 요소는 두 리소스 레코드 세트에 대해 동일한 값을 갖기 때문에 이 요소가 필요합니다. Z1111111111111 및 Z3333333333333은 DNSName 값으로 지정된 ELB 로드 밸런서에 대한 호스팅 영역 ID의 예입니다. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com 및 example-load-balancer-4444444444.us-east-1.elb.amazonaws.com은 Amazon Route 53가 example.com 쿼리에 응답하는 Elastic Load Balancing 도메인의 예입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하세요. Get-R53Change cmdlet을 사용하여 변경 사항이 완료되는 시기를 확인할 수 있습니다.

```
$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
```

예제 5: 이 예제에서는 두 개의 지연 시간 별칭 리소스 레코드 세트를 생성합니다. 하나는 미국 서부(오리건) 리전(us-west-2)의 ELB 로드 밸런서용이고 다른 하나는 아시아 태평양(싱가포르) 리전(ap-southeast-1)의 로드 밸런서용입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하세요. Get-R53Change cmdlet을 사용하여 변경 사항이 완료되는 시기를 확인할 수 있습니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ChangeResourceRecordSets를 참조하세요.

`Get-R53AccountLimit`

다음 코드 예시는 `Get-R53AccountLimit`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 현재 계정을 사용하여 생성할 수 있는 최대 호스팅 영역 수를 반환합니다.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

출력:

```
15
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAccountLimit을 참조하세요.

`Get-R53CheckerIpRanges`

다음 코드 예시는 `Get-R53CheckerIpRanges`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Route53 상태 확인기의 CIDR을 반환합니다.

```
Get-R53CheckerIpRanges
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCheckerIpRanges를 참조하세요.

Get-R53HostedZone

다음 코드 예시는 Get-R53HostedZone의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: ID Z1D633PJN98FT9를 사용하여 호스팅 영역의 세부 정보를 반환합니다.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetHostedZone을 참조하세요.

Get-R53HostedZoneCount

다음 코드 예시는 Get-R53HostedZoneCount의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 현재의 총 퍼블릭 및 프라이빗 호스팅 영역 수를 반환합니다 AWS 계정.

```
Get-R53HostedZoneCount
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetHostedZoneCount를 참조하세요.

Get-R53HostedZoneLimit

다음 코드 예시는 Get-R53HostedZoneLimit의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 호스팅 영역에서 생성할 수 있는 최대 레코드 수 한도를 반환합니다.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

출력:

```
5
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetHostedZoneLimit을 참조하세요.

Get-R53HostedZoneList

다음 코드 예시는 Get-R53HostedZoneList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 퍼블릭 및 프라이빗 호스팅 영역을 출력합니다.

```
Get-R53HostedZoneList
```

예제 2: ID NZ8X2CISAMPLE이 있는 재사용 가능한 위임 세트와 연결된 호스팅된 영역을 모두 출력합니다.

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListHostedZones을 참조하세요.

Get-R53HostedZonesByName

다음 코드 예시는 Get-R53HostedZonesByName의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 퍼블릭 및 프라이빗 호스팅 영역을 도메인 이름별로 ASCII 순서로 반환합니다.

```
Get-R53HostedZonesByName
```

예제 2: 지정된 DNS 이름부터 시작하여 도메인 이름별로 ASCII 순서로 퍼블릭 및 프라이빗 호스팅 영역을 반환합니다.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListHostedZonesByName을 참조하세요.

`Get-R53QueryLoggingConfigList`

다음 코드 예시는 `Get-R53QueryLoggingConfigList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 현재 AWS 계정와 연결된 DNS 쿼리 로깅에 대한 모든 구성을 반환합니다.

```
Get-R53QueryLoggingConfigList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListQueryLoggingConfigs를 참조하세요.

`Get-R53ReusableDelegationSet`

다음 코드 예시는 `Get-R53ReusableDelegationSet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 위임 세트에 할당된 네 개의 이름 서버를 포함하여 지정된 위임 세트에 대한 정보를 검색합니다.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetReusableDelegationSet를 참조하세요.

New-R53HostedZone

다음 코드 예시는 New-R53HostedZone의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 재사용 가능한 위임 세트와 연결된 'example.com'이라는 새 호스팅 영역을 생성합니다. 작업을 두 번 실행할 위험 없이 필요한 경우 요청을 재시도하려면 CallerReference 파라미터에 값을 제공해야 합니다. 호스팅 영역은 VPC에서 생성되므로 자동으로 비공개 상태이며, -HostedZoneConfig_PrivateZone 파라미터를 설정해서는 안 됩니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateHostedZone을 참조하세요.

New-R53QueryLoggingConfig

다음 코드 예시는 New-R53QueryLoggingConfig의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 호스팅 영역에 대한 새 Route53 DNS 쿼리 로깅 구성을 생성합니다. Amazon Route53은 지정된 Cloudwatch 로그 그룹에 DNS 쿼리 로그를 게시합니다.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

출력:

```
QueryLoggingConfig                      Location
------------------                      --------
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateQueryLoggingConfig를 참조하세요.

`New-R53ReusableDelegationSet`

다음 코드 예시는 `New-R53ReusableDelegationSet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 여러 호스팅 영역에서 재사용할 수 있는 4개의 이름 서버로 구성된 재사용 가능한 위임 세트를 생성합니다.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

출력:

```
DelegationSet                      Location
-------------                      --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateReusableDelegationSet를 참조하세요.

Register-R53VPCWithHostedZone

다음 코드 예시는 Register-R53VPCWithHostedZone의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC를 프라이빗 호스팅 영역과 연결합니다.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

출력:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateVPCWithHostedZone을 참조하세요.

Remove-R53HostedZone

다음 코드 예시는 Remove-R53HostedZone의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 ID로 호스팅 영역을 삭제합니다. -Force 스위치 파라미터를 추가하지 않으면 명령이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteHostedZone을 참조하세요.

Remove-R53QueryLoggingConfig

다음 코드 예시는 Remove-R53QueryLoggingConfig의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 DNS 쿼리 로깅에 지정된 구성을 제거합니다.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteQueryLoggingConfig를 참조하세요.

`Remove-R53ReusableDelegationSet`

다음 코드 예시는 `Remove-R53ReusableDelegationSet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 재사용 가능한 위임 세트를 삭제합니다.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteReusableDelegationSet를 참조하세요.

`Unregister-R53VPCFromHostedZone`

다음 코드 예시는 `Unregister-R53VPCFromHostedZone`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 VPC를 프라이빗 호스팅 영역에서 연결 해제합니다.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

출력:

```
Id                    Status  SubmittedAt         Comment
--                    ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisassociateVPCFromHostedZone을 참조하세요.

`Update-R53HostedZoneComment`

다음 코드 예시는 `Update-R53HostedZoneComment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 호스팅 영역에 대한 설명을 업데이트합니다.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

출력:

```
Id                     : /hostedzone/Z385PDS6AAAAAR
Name                   : example.com.
CallerReference        : C5B55555-7147-EF04-8341-69131E805C89
Config                 : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService          :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateHostedZoneComment를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon S3 예제

다음 코드 예제에서는 Amazon S3에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Copy-S3Object`

다음 코드 예시는 `Copy-S3Object`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 동일한 버킷으로 복사하지만 새로운 키인 “sample-copy.txt”를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

예시 2: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 "backup-files" 버킷으로 복사하며 “sample-copy.txt” 키를 사용합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
 sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

예시 3: 이 명령은 “test-files” 버킷에서 "local-sample.txt"라는 이름의 로컬 파일로 "sample.txt" 객체를 다운로드합니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

예시 4: 단일 객체를 지정된 파일로 다운로드합니다. 다운로드한 파일은 c:\downloads\data\archive.zip에서 찾을 수 있습니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

예시 5: 지정된 키 접두사와 일치하는 모든 객체를 로컬 폴더로 다운로드합니다. 상대 키 계층 구조는 전체 다운로드 위치에 하위 폴더로 보존됩니다.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CopyObject를 참조하세요.

Get-S3ACL

다음 코드 예시는 Get-S3ACL의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 S3 객체의 객체 소유자 세부 정보를 가져옵니다.

```
Get-S3ACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 AccessControlList.Owner
```

출력:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetACL을 참조하세요.

Get-S3Bucket

다음 코드 예시는 Get-S3Bucket의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 모든 S3 버킷을 반환합니다.

```
Get-S3Bucket
```

예시 2: 이 명령은 이름이 “test-files”인 버킷을 반환합니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListBuckets을 참조하세요.

Get-S3BucketAccelerateConfiguration

다음 코드 예시는 Get-S3BucketAccelerateConfiguration의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷에 대해 전송 가속화 설정이 활성화된 경우 Enabled 값을 반환합니다.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketAccelerateConfiguration을 참조하세요.

`Get-S3BucketAnalyticsConfiguration`

다음 코드 예시는 `Get-S3BucketAnalyticsConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터의 세부 정보를 반환합니다.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketAnalyticsConfiguration을 참조하세요.

`Get-S3BucketAnalyticsConfigurationList`

다음 코드 예시는 `Get-S3BucketAnalyticsConfigurationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 분석 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListBucketAnalyticsConfigurations을 참조하세요.

`Get-S3BucketEncryption`

다음 코드 예시는 `Get-S3BucketEncryption`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 연결된 모든 서버 측 암호화 규칙을 반환합니다.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketEncryption을 참조하세요.

`Get-S3BucketInventoryConfiguration`

다음 코드 예시는 `Get-S3BucketInventoryConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 대해 이름이 'testinventory'인 인벤토리의 세부 정보를 반환합니다.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketInventoryConfiguration을 참조하세요.

`Get-S3BucketInventoryConfigurationList`

다음 코드 예시는 `Get-S3BucketInventoryConfigurationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 인벤토리 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListBucketInventoryConfigurations을 참조하세요.

`Get-S3BucketLocation`

다음 코드 예시는 `Get-S3BucketLocation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 제약 조건이 있는 경우 'amzn-s3-demo-bucket' 버킷에 대한 위치 제약 조건을 반환합니다.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Value
-----
ap-south-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketLocation을 참조하세요.

`Get-S3BucketLogging`

다음 코드 예시는 `Get-S3BucketLogging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷의 로깅 상태를 반환합니다.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketLogging을 참조하세요.

`Get-S3BucketMetricsConfiguration`

다음 코드 예시는 `Get-S3BucketMetricsConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 'testfilter'라는 지표 필터에 대한 세부 정보를 반환합니다.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketMetricsConfiguration을 참조하세요.

`Get-S3BucketNotification`

다음 코드 예시는 `Get-S3BucketNotification`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 예시는 지정된 버킷의 알림 구성을 검색합니다.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

출력:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketNotification을 참조하세요.

`Get-S3BucketPolicy`

다음 코드 예시는 `Get-S3BucketPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 출력합니다.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketPolicy를 참조하세요.

`Get-S3BucketPolicyStatus`

다음 코드 예시는 `Get-S3BucketPolicyStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 버킷이 퍼블릭 버킷인지 여부를 나타내는 지정된 S3 버킷에 대한 정책 상태를 반환합니다.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketPolicyStatus를 참조하세요.

`Get-S3BucketReplication`

다음 코드 예시는 `Get-S3BucketReplication`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 설정된 복제 구성 정보를 반환합니다.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketReplication을 참조하세요.

`Get-S3BucketRequestPayment`

다음 코드 예시는 `Get-S3BucketRequestPayment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷에 대한 요청 결제 구성을 반환합니다. 기본적으로 버킷에서 다운로드하는 비용은 버킷 소유자가 지불합니다.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketRequestPayment를 참조하세요.

`Get-S3BucketTagging`

다음 코드 예시는 `Get-S3BucketTagging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 관련된 모든 태그를 반환합니다.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketTagging을 참조하세요.

`Get-S3BucketVersioning`

다음 코드 예시는 `Get-S3BucketVersioning`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷과 관련된 버전 관리 상태를 반환합니다.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketVersioning을 참조하세요.

`Get-S3BucketWebsite`

다음 코드 예시는 `Get-S3BucketWebsite`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 구성 세부 정보를 반환합니다.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetBucketWebsite를 참조하세요.

`Get-S3CORSConfiguration`

다음 코드 예시는 `Get-S3CORSConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 S3 버킷에 해당하는 모든 CORS 구성 규칙을 포함하는 객체를 반환합니다.

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 Configuration.Rules
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCORSConfiguration을 참조하세요.

`Get-S3LifecycleConfiguration`

다음 코드 예시는 `Get-S3LifecycleConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 버킷의 수명 주기 구성을 검색합니다.

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

출력:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetLifecycleConfiguration을 참조하세요.

`Get-S3Object`

다음 코드 예시는 `Get-S3Object`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListObjects를 참조하세요.

`Get-S3ObjectLockConfiguration`

다음 코드 예시는 `Get-S3ObjectLockConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 대해 객체 잠금 구성이 활성화된 경우 'Enabled' 값을 반환합니다.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

출력:

```
Value
-----
Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetObjectLockConfiguration을 참조하세요.

`Get-S3ObjectMetadata`

다음 코드 예시는 `Get-S3ObjectMetadata`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 S3 버킷에 'ListTrusts.txt' 키가 있는 객체의 메타데이터를 반환합니다.

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetObjectMetadata를 참조하세요.

Get-S3ObjectRetention

다음 코드 예시는 Get-S3ObjectRetention의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 객체 보존의 모드와 종료 날짜를 반환합니다.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetObjectRetention을 참조하세요.

Get-S3ObjectTagSet

다음 코드 예시는 Get-S3ObjectTagSet의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 샘플은 지정된 S3 버킷에 있는 객체와 연결된 태그를 반환합니다.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Key  Value
---  -----
test value
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetObjectTagging을 참조하세요.

`Get-S3PreSignedURL`

다음 코드 예시는 `Get-S3PreSignedURL`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 키와 만료 날짜에 대해 미리 서명된 URL을 반환합니다.

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires
 '2023-11-16'
```

예제 2: 이 명령에서는 지정된 키와 만료 날짜가 있는 디렉터리 버킷에 대해 미리 서명된 URL을 반환합니다.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPreSignedURL을 참조하세요.

`Get-S3PublicAccessBlock`

다음 코드 예시는 `Get-S3PublicAccessBlock`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 퍼블릭 액세스 차단 구성을 반환합니다.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPublicAccessBlock을 참조하세요.

`Get-S3Version`

다음 코드 예시는 `Get-S3Version`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 S3 버킷의 모든 객체 버전에 대한 메타데이터를 반환합니다.

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : amzn-s3-demo-bucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListVersions을 참조하세요.

`New-S3Bucket`

다음 코드 예시는 `New-S3Bucket`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 'sample-bucket'이라는 새 프라이빗 버킷을 생성합니다.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

예제 2: 이 명령에서는 읽기-쓰기 권한을 가진 'sample-bucket'이라는 새 버킷을 생성합니다.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

예제 3: 이 명령에서는 읽기 전용 권한을 가진 'sample-bucket'이라는 새 버킷을 생성합니다.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

예제 4: 이 명령에서는 PutBucketConfiguration을 사용하여 'amzn-s3-demo-bucket--use1-az5--x-s3'이라는 새 디렉터리 버킷을 생성합니다.

```
$bucketConfiguration = @{
      BucketInfo = @{
          DataRedundancy = 'SingleAvailabilityZone'
          Type = 'Directory'
      }
      Location = @{
          Name = 'usw2-az1'
          Type = 'AvailabilityZone'
      }
    }
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration
 $bucketConfiguration -Region us-west-2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucket을 참조하세요.

`Read-S3Object`

다음 코드 예시는 `Read-S3Object`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령은 버킷 "amzn-s3-demo-bucket"에서 "sample.txt" 항목을 검색하여 현재 위치의 "local-sample.txt"라는 파일에 저장합니다. 이 명령을 직접 호출하기 전에 "local-sample.txt" 파일이 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

예제 2:이 명령은 버킷 "amzn-s3-demo-bucket"에서 가상 디렉터리 "DIR"을 검색하여 현재 위치의 "Local-DIR" 폴더에 저장합니다. 이 명령을 직접 호출하기 전에 "Local-DIR" 폴더가 없어도 됩니다.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

예시 3: 버킷 이름에 'config'가 있는 버킷에서 키가 '.json'으로 끝나는 모든 객체를 지정된 폴더의 파일로 다운로드합니다. 객체 키는 파일 이름을 설정하는 데 사용됩니다.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetObject를 참조하세요.

`Remove-S3Bucket`

다음 코드 예시는 `Remove-S3Bucket`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 'test-files' 버킷에서 모든 객체 및 객체 버전을 제거한 다음 버킷을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가합니다. 비어 있지 않은 버킷은 삭제할 수 없다는 점에 유의하세요.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucket을 참조하세요.

`Remove-S3BucketAnalyticsConfiguration`

다음 코드 예시는 `Remove-S3BucketAnalyticsConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터를 제거합니다.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketAnalyticsConfiguration을 참조하세요.

`Remove-S3BucketEncryption`

다음 코드 예시는 `Remove-S3BucketEncryption`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이렇게 하면 제공된 S3 버킷에 대해 활성화된 암호화가 비활성화됩니다.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketEncryption을 참조하세요.

`Remove-S3BucketInventoryConfiguration`

다음 코드 예시는 `Remove-S3BucketInventoryConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에 해당하는 'testInventoryName'이라는 이름의 인벤토리를 제거합니다.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testInventoryName'
```

출력:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketInventoryConfiguration을 참조하세요.

`Remove-S3BucketMetricsConfiguration`

다음 코드 예시는 `Remove-S3BucketMetricsConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testmetrics'인 지표 필터를 제거합니다.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketMetricsConfiguration을 참조하세요.

`Remove-S3BucketPolicy`

다음 코드 예시는 `Remove-S3BucketPolicy`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 제거합니다.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketPolicy를 참조하세요.

`Remove-S3BucketReplication`

다음 코드 예시는 `Remove-S3BucketReplication`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷과 연결된 복제 구성을 삭제합니다. 이 작업에는 s3:DeleteReplicationConfiguration 작업에 대한 권한이 필요합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketReplication을 참조하세요.

`Remove-S3BucketTagging`

다음 코드 예시는 `Remove-S3BucketTagging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷과 연결된 모든 태그를 제거합니다.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketTagging을 참조하세요.

`Remove-S3BucketWebsite`

다음 코드 예시는 `Remove-S3BucketWebsite`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 호스팅 속성을 비활성화합니다.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteBucketWebsite를 참조하세요.

`Remove-S3CORSConfiguration`

다음 코드 예시는 `Remove-S3CORSConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 S3 버킷의 CORS 구성을 제거합니다.

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
 target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteCORSConfiguration을 참조하세요.

`Remove-S3LifecycleConfiguration`

다음 코드 예시는 `Remove-S3LifecycleConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 지정된 S3 버킷에 대한 모든 수명 주기 규칙을 제거합니다.

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteLifecycleConfiguration을 참조하세요.

`Remove-S3MultipartUpload`

다음 코드 예시는 `Remove-S3MultipartUpload`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 5일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

예시 2: 이 명령은 2014년 1월 2일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
 January 02, 2014"
```

예시 3: 이 명령은 2014년 1월 2일 10시 45분 37초 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02
 10:45:37"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AbortMultipartUpload를 참조하세요.

`Remove-S3Object`

다음 코드 예시는 `Remove-S3Object`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 "test-files" 버킷에서 "sample.txt" 객체를 제거합니다. 명령이 실행되기 전에 확인 프롬프트가 표시됩니다. 확인 프롬프트를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

예시 2: 이 명령은 버킷이 객체 버전을 활성화하도록 구성된 경우 “test-files” 버킷에서 지정된 버전의 “sample.txt” 객체를 제거합니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

예시 3: 이 명령은 단일 배치 작업으로 "test-files" 버킷에서 "sample1.txt", "sample2.txt" 및 "sample3.txt" 객체를 제거합니다. 서비스 응답에는 삭제의 성공 또는 오류 상태에 관계없이 처리된 모든 키가 나열됩니다. 서비스에서 처리하지 못한 키에 대한 오류만 가져오려면 -ReportErrorsOnly 파라미터를 추가합니다. 이 파라미터는 -Quiet라는 별칭으로 지정할 수도 있습니다.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

예시 4: 이 예시는 -keyCollection 파라미터와 함께 인라인 표현식을 사용하여 삭제할 객체의 키를 가져옵니다. Get-S3Object는 Amazon.S3.Model.S3Object 인스턴스의 컬렉션을 반환하며, 각 인스턴스에는 객체를 식별하는 유형 문자열의 키 멤버가 있습니다.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

예시 5: 이 예시는 버킷에서 키 접두사 “prefix/subprefix”가 있는 모든 객체를 가져와 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

예시 6: 이 예시는 삭제 마커를 나타내는 Amazon.S3.Model.S3ObjectVersion 인스턴스 컬렉션을 cmdlet으로 파이프하여 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우, 컬렉션을 cmdlet의 -InputObject(별칭 -S3ObjectCollection) 파라미터에 전달하여 서비스를 한 번 직접적으로 호출함으로써 일괄 삭제가 이루어지도록 하는 것이 좋습니다.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

예시 7: 이 스크립트는 -KeyAndVersionCollection 파라미터와 함께 사용할 객체의 배열을 구성하여 객체 집합(이 경우에는 삭제 마커)을 일괄 삭제하는 방법을 보여 줍니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteObjects를 참조하세요.

`Remove-S3ObjectTagSet`

다음 코드 예시는 `Remove-S3ObjectTagSet`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷에서 'testfile.txt' 키가 있는 객체와 연결된 모든 태그를 제거합니다.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
 '^Key'
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteObjectTagging을 참조하세요.

`Remove-S3PublicAccessBlock`

다음 코드 예시는 `Remove-S3PublicAccessBlock`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷의 퍼블릭 액세스 차단 설정을 끕니다.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

출력:

```
amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePublicAccessBlock을 참조하세요.

`Set-S3BucketEncryption`

다음 코드 예시는 `Set-S3BucketEncryption`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 버킷에서 Amazon S3 관리형 키(SSE-S3)를 사용하여 기본 AES256 서버 측 암호화를 활성화합니다.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketEncryption을 참조하세요.

`Test-S3Bucket`

다음 코드 예시는 `Test-S3Bucket`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령에서는 버킷이 있는 경우 True를 반환하고, 그렇지 않으면 False를 반환합니다. 버킷이 사용자에게 속하지 않더라도 명령은 True를 반환합니다.

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Test-S3Bucket을 참조하세요.

`Write-S3BucketAccelerateConfiguration`

다음 코드 예시는 `Write-S3BucketAccelerateConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 전송 가속화를 활성화합니다.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketAccelerateConfiguration을 참조하세요.

`Write-S3BucketNotification`

다음 코드 예시는 `Write-S3BucketNotification`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 예시는 S3 이벤트 ObjectRemovedDelete에 대한 SNS 주제를 구성하고 지정된 S3 버킷에 대한 알림을 활성화합니다.

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

예시 2: 이 예시는 지정된 버킷에 대해 ObjectCreatedAll 알림을 활성화하여 Lambda 함수로 전송합니다.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
```

```
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

예시 3: 이 예시는 서로 다른 키 접미사를 기반으로 2개의 서로 다른 Lambda 구성을 생성하고 단일 명령으로 둘 모두를 구성합니다.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
```

```
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketNotification을 참조하세요.

`Write-S3BucketReplication`

다음 코드 예시는 `Write-S3BucketReplication`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 'amzn-s3-demo-bucket' 버킷에서 키 이름 접두사 'TaxDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 단일 규칙을 사용하여 복제 구성을 설정합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

예제 2: 이 예제에서는 키 이름 접두사 'TaxDocs' 또는 'OtherDocs'로 생성된 모든 새 객체를 'amzn-s3-demo-bucket' 버킷에 복제할 수 있도록 하는 여러 규칙을 사용하여 복제 구성을 설정합니다. 키 접두사는 겹치지 않아야 합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

예제 3: 이 예제에서는 지정된 버킷의 복제 구성을 업데이트하여 키 이름 접두사가 'TaxDocs'인 객체를 'amzn-s3-demo-bucket' 버킷으로 복제하는 작업을 제어하는 규칙을 비활성화합니다.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketReplication을 참조하세요.

`Write-S3BucketRequestPayment`

다음 코드 예시는 `Write-S3BucketRequestPayment`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이름이 'amzn-s3-demo-bucket'인 버킷의 요청 결제 구성을 업데이트하여 버킷에서 다운로드를 요청하는 사람에게 다운로드 요금이 부과되도록 합니다. 기본적으로 다운로드 비용은 버킷 소유자가 지불합니다. 지불 요청을 기본값으로 다시 설정하려면 RequestPaymentConfiguration_Payer 파라미터에 'BucketOwner'를 사용하세요.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketRequestPayment를 참조하세요.

`Write-S3BucketTagging`

다음 코드 예시는 `Write-S3BucketTagging`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 이름이 **`cloudtrail-test-2018`**인 버킷에 두 개의 태그를 적용합니다. 하나는 키가 Stage이고 값이 Test인 태그이며 다른 하나는 키가 Environment고 값이 Alpha인 태그입니다. 버킷에 태그가 추가되었는지 확인하려면 **`Get-S3BucketTagging -BucketName bucket_name`**을 실행합니다. 결과에는 첫 번째 명령에서 버킷에 적용한 태그가 표시되어야 합니다. 단, **`Write-S3BucketTagging`**은 버킷에 설정된 기존 태그 전체를 덮어씁니다. 개별 태그를 추가하거나 삭제하려면 리소스 그룹 및 태그 지정 API cmdlet, **`Add-RGTResourceTag`** 및 **`Remove-RGTResourceTag`**를 실행합니다. 또는 AWS Management Console에서 Tag Editor를 사용하여 S3 버킷 태그를 관리합니다.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

예시 2: 이 명령은 이름이 **`cloudtrail-test-2018`**인 버킷을 **`Write-S3BucketTagging`** cmdlet으로 파이프합니다. 이렇게 하면 Stage:Production 및 Department:Finance 태그가 버킷에 적용됩니다. 단, **`Write-S3BucketTagging`**은 버킷에 설정된 기존 태그 전체를 덮어씁니다.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketTagging을 참조하세요.

`Write-S3BucketVersioning`

다음 코드 예시는 `Write-S3BucketVersioning`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 버전 관리를 활성화합니다.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status
 Enabled
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketVersioning을 참조하세요.

`Write-S3BucketWebsite`

다음 코드 예시는 `Write-S3BucketWebsite`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 인덱스 문서가 'index.html', 오류 문서가 'error.html'인 지정된 버킷에 대한 웹 사이트 호스팅을 활성화합니다.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutBucketWebsite를 참조하세요.

`Write-S3LifecycleConfiguration`

다음 코드 예시는 `Write-S3LifecycleConfiguration`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 $NewRule에 제공된 구성을 쓰거나 바꿉니다. 이 구성은 지정된 접두사 및 태그 값으로 범위 객체를 제한합니다.

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  50
  }
  Id =  "Test-From-Write-cmdlet-1"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "non-use"
       "Value" = "yes"
      }
     }
    )
   }
  }
  "Status"= 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 75
  }
 }

 Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
 $NewRule
```

예제 2: 이 예제에서는 필터링을 사용하여 여러 규칙을 설정합니다. $ArchiveRule은 객체가 30일 후에는 Glacier로, 120일 후에는 DeepArchive로 아카이브되도록 설정합니다. $ExpireRule은 'py' 접두사와 tag:key 'archieved'가 'yes'로 설정된 객체의 현재 버전과 이전 버전을 모두 150일 후에 만료시킵니다.

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  150
```

```
 }
 Id =  "Remove-in-150-days"
 Filter=  @{
  LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
   Operands=  @(
    [Amazon.S3.Model.LifecyclePrefixPredicate] @{
     "Prefix" =  "py"
    },
    [Amazon.S3.Model.LifecycleTagPredicate] @{
     "Tag"=  @{
      "Key" =  "archived"
      "Value" = "yes"
     }
    }
   )
  }
 }
 Status= 'Enabled'
 NoncurrentVersionExpiration = @{
  NoncurrentDays = 150
 }
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
 Expiration =  $null
 Id =  "Archive-to-Glacier-in-30-days"
 Filter=  @{
  LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
   Operands= @(
    [Amazon.S3.Model.LifecyclePrefixPredicate] @{
     "Prefix" =  "py"
    },
    [Amazon.S3.Model.LifecycleTagPredicate] @{
     "Tag"=  @{
      "Key" =  "reviewed"
      "Value" = "yes"
     }
    }
   )
  }
 }
 Status = 'Enabled'
 NoncurrentVersionExpiration = @{
  NoncurrentDays = 75
```

```
 }
 Transitions = @(
  @{
   Days = 30
   "StorageClass"= 'Glacier'
  },
  @{
   Days = 120
   "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
  }
 )
}

Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutLifecycleConfiguration을 참조하세요.

`Write-S3Object`

다음 코드 예시는 `Write-S3Object`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

예시 2: 이 명령은 단일 파일 "sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "sample.txt" 키가 있는 객체를 생성합니다. -Key 파라미터가 제공되지 않은 경우 파일 이름이 S3 객체 키로 사용됩니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

예시 3: 이 명령은 단일 파일 "local-sample.txt"를 Amazon S3에 업로드하고, "test-files" 버킷에 "prefix/to/sample.txt" 키가 있는 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

예시 4: 이 명령은 하위 디렉터리 "Scripts"의 모든 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

예시 5: 이 명령은 로컬 디렉터리 "Scripts"의 모든 *.ps1 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 "SampleScripts"를 각 객체에 적용합니다. 업로드된 각 파일에는 "SampleScripts/filename.ps1"이라는 키가 있으며, 여기서 'filename'은 상황에 따라 다릅니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

예시 6: 이 명령은 키가 'sample.txt'인 지정된 콘텐츠 문자열을 포함하는 새 S3 객체를 생성합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

예시 7: 이 명령은 지정된 파일(파일 이름이 키로 사용됨)을 업로드하고 지정된 태그를 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

예시 8: 이 명령은 지정된 폴더를 재귀적으로 업로드하고 지정된 태그를 모든 새 객체에 적용합니다.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutObject를 참조하세요.

`Write-S3ObjectRetention`

다음 코드 예시는 `Write-S3ObjectRetention`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예시 1: 이 명령은 지정된 S3 버킷의 'testfile.txt' 객체에 대해 '2019년 12월 31일 00:00:00' 날짜까지 거버넌스 보존 모드를 활성화합니다.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutObjectRetention을 참조하세요.

Tools for PowerShell V4를 사용한 Security Hub CSPM 예제

다음 코드 예제에서는 Security Hub CSPM과 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-SHUBFinding`

다음 코드 예시는 `Get-SHUBFinding`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 Amazon EC2 서비스에서 Security Hub 조사 결과를 검색합니다.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.ResourceType = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
```

```
    Comparison = 'PREFIX'
    Value = 'AwsEc2'
}
Get-SHUBFinding -Filter $filter
```

예제 2:이 명령은 AWS 계정 ID 123456789012에서 Security Hub 조사 결과를 검색합니다.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.AwsAccountId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = '123456789012'
}
Get-SHUBFinding -Filter $filter
```

예제 3: 이 명령에서는 표준 'pci-dss'에 대해 생성된 Security Hub 조사 결과를 검색합니다.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.GeneratorId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'pci-dss'
}
Get-SHUBFinding -Filter $filter
```

예제 4: 이 명령에서는 워크플로 상태가 NOTIFIED인 Security Hub 중요 심각도 조사 결과를 검색합니다.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.SeverityLabel = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = 'CRITICAL'
}
$filter.WorkflowStatus = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter
 -Property @{
    Comparison = 'EQUALS'
    Value = 'NOTIFIED'
}
Get-SHUBFinding -Filter $filter
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetFindings을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon SES 예제

다음 코드 예제에서는 Amazon SES에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Get-SESIdentity`

다음 코드 예시는 `Get-SESIdentity`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 명령은 확인 상태에 관계없이 특정 AWS 계정의 모든 자격 증명(이메일 주소 및 도메인)이 포함된 목록을 반환합니다.

```
Get-SESIdentity
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListIdentities를 참조하세요.

`Get-SESSendQuota`

다음 코드 예시는 `Get-SESSendQuota`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 사용자의 현재 전송 한도를 반환합니다.

```
Get-SESSendQuota
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetSendQuota를 참조하세요.

`Get-SESSendStatistic`

다음 코드 예시는 `Get-SESSendStatistic`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령은 사용자의 전송 통계를 반환합니다. 결과는 지난 2주 동안의 전송 활동을 나타내는 데이터 포인트 목록입니다. 목록의 각 데이터 포인트에는 15분 간격의 통계가 포함됩니다.

```
Get-SESSendStatistic
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetSendStatistics를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon SES API v2 예제

다음 코드 예제에서는 Amazon SES API v2와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Send-SES2Email`

다음 코드 예시는 `Send-SES2Email`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 표준 이메일 메시지를 보내는 방법을 보여줍니다.

```
Send-SES2Email -FromEmailAddress "sender@example.com" -Destination_ToAddress
 "recipient@example.com" -Subject_Data "Email Subject" -Text_Data "Email Body"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SendEmail을 참조하세요.

Tools for PowerShell V4를 사용한 Amazon SNS 예제

다음 코드 예제에서는 Amazon SNS에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Publish-SNSMessage`

다음 코드 예시는 `Publish-SNSMessage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 단일 MessageAttribute가 인라인으로 선언된 메시지를 게시하는 것을 보여줍니다.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

예제 2: 이 예제에서는 여러 MessageAttributes가 미리 선언된 메시지를 게시하는 것을 보여줍니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Publish를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon SQS 예제

다음 코드 예제에서는 Amazon SQS에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-SQSPermission`

다음 코드 예시는 `Add-SQSPermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1:이 예제에서는 지정된가 지정된 대기열에서 메시지를 보내 AWS 계정 도록 허용합니다.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddPermission을 참조하세요.

`Clear-SQSQueue`

다음 코드 예시는 `Clear-SQSQueue`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 모든 메시지를 삭제합니다.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PurgeQueue를 참조하세요.

`Edit-SQSMessageVisibility`

다음 코드 예시는 `Edit-SQSMessageVisibility`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 지정된 수신 핸들이 있는 메시지의 표시 제한 시간을 10시간(10시간 * 60분 * 60초 = 36,000초)으로 변경합니다.

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ChangeMessageVisibility를 참조하세요.

`Edit-SQSMessageVisibilityBatch`

다음 코드 예시는 `Edit-SQSMessageVisibilityBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 지정된 수신 핸들이 있는 2개의 메시지의 표시 제한 시간을 변경합니다. 첫 번째 메시지의 표시 제한 시간이 10시간(10시간 * 60분 * 60초 = 36,000초)으로 변경됩니다. 두 번째 메시지의 표시 제한 시간이 5시간(5시간 * 60분 * 60초 = 18,000초)으로 변경됩니다.

```
$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
```

출력:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ChangeMessageVisibilityBatch를 참조하세요.

Get-SQSDeadLetterSourceQueue

다음 코드 예시는 Get-SQSDeadLetterSourceQueue의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대기열을 배달 못한 편지 대기열로 사용하는 대기열의 URL을 나열합니다.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

출력:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDeadLetterSourceQueues를 참조하세요.

Get-SQSQueue

다음 코드 예시는 Get-SQSQueue의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 대기열을 나열합니다.

```
Get-SQSQueue
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 이름으로 시작하는 대기열을 나열합니다.

```
Get-SQSQueue -QueueNamePrefix My
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListQueues를 참조하세요.

`Get-SQSQueueAttribute`

다음 코드 예시는 `Get-SQSQueueAttribute`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대기열의 모든 속성을 나열합니다.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

출력:

```
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":"2012-10-17",
 "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]...}
```

예제 2: 이 예제에서는 지정된 대기열에 대해 지정된 속성만 별도로 나열합니다.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

출력:

```
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":"2012-10-17",
 "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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetQueueAttributes를 참조하세요.

`Get-SQSQueueUrl`

다음 코드 예시는 `Get-SQSQueueUrl`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름이 있는 대기열의 URL을 나열합니다.

```
Get-SQSQueueUrl -QueueName MyQueue
```

출력:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetQueueUrl을 참조하세요.

`New-SQSQueue`

다음 코드 예시는 `New-SQSQueue`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 이름으로 대기열을 생성합니다.

```
New-SQSQueue -QueueName MyQueue
```

출력:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateQueue를 참조하세요.

`Receive-SQSMessage`

다음 코드 예시는 `Receive-SQSMessage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대기열에 대해 수신할 다음 메시지 최대 10개에 대한 정보를 나열합니다. 지정된 메시지 속성이 있는 경우 해당 속성에 대한 값이 정보에 포함됩니다.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ReceiveMessage를 참조하세요.

`Remove-SQSMessage`

다음 코드 예시는 `Remove-SQSMessage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 지정된 수신 핸들이 있는 메시지를 삭제합니다.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteMessage를 참조하세요.

`Remove-SQSMessageBatch`

다음 코드 예시는 `Remove-SQSMessageBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 지정된 수신 핸들이 있는 메시지 2개를 삭제합니다.

```
$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
```

출력:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteMessageBatch를 참조하세요.

`Remove-SQSPermission`

다음 코드 예시는 `Remove-SQSPermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 지정된 대기열에서 지정된 레이블이 있는 권한 설정을 제거합니다.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemovePermission을 참조하세요.

`Remove-SQSQueue`

다음 코드 예시는 `Remove-SQSQueue`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 대기열을 삭제합니다.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteQueue를 참조하세요.

Send-SQSMessage

다음 코드 예시는 Send-SQSMessage의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 속성과 메시지 본문이 포함된 메시지를 메시지 전송이 10초 동안 지연된 상태로 지정된 대기열로 보냅니다.

```
$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
```

출력:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SendMessage를 참조하세요.

Send-SQSMessageBatch

다음 코드 예시는 Send-SQSMessageBatch의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 속성 및 메시지 본문이 포함된 메시지 2개를 지정된 대기열로 보냅니다. 첫 번째 메시지의 경우 15초, 두 번째 메시지의 경우 10초 동안 전송이 지연됩니다.

```
$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
```

출력:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SendMessageBatch를 참조하세요.

Set-SQSQueueAttribute

다음 코드 예시는 Set-SQSQueueAttribute의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 SNS 주제에 대기열을 구독하는 정책을 설정하는 방법을 보여줍니다. 메시지를 주제에 게시하면 구독 중인 대기열에 메시지가 전송됩니다.

```
# 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
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$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 }
```

예제 2: 이 예제는 지정된 대기열에 대해 지정된 속성을 설정합니다.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SetQueueAttributes를 참조하세요.

AWS STS Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS STS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Convert-STSAuthorizationMessage`

다음 코드 예시는 `Convert-STSAuthorizationMessage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 요청에 대한 응답으로 반환된 제공된 인코딩된 메시지 내용에 포함된 추가 정보를 디코딩합니다. 권한 부여 상태의 세부 정보가 작업을 요청한 사용자가 볼 수 없는 권한 있는 정보로 구성될 수 있기 때문에 추가 정보가 인코딩됩니다.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DecodeAuthorizationMessage를 참조하세요.

`Get-STSFederationToken`

다음 코드 예시는 `Get-STSFederationToken`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 페더레이션 사용자의 이름으로 'Bob'을 사용하여 1시간 동안 유효한 페더레이션 토큰을 요청합니다. 이 이름은 리소스 기반 정책(예: Amazon S3 버킷 정책)에서 페더레이션 사용자 이름을 참조하는 데 사용할 수 있습니다. JSON 형식으로 제공된 IAM 정책은 IAM 사용자가 사용할 수 있는 권한의 범위를 좁히는 데 사용됩니다. 제공된 정책은 요청하는 사용자에게 부여된 것보다 더 많은 권한을 부여할 수 없으며, 페더레이션 사용자에 대한 최종 권한은 전달된 정책과 IAM 사용자 정책의 교차점을 기준으로 가장 제한적인 세트입니다.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetFederationToken을 참조하세요.

`Get-STSSessionToken`

다음 코드 예시는 `Get-STSSessionToken`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 설정된 기간 동안 유효한 임시 자격 증명이 포함된 **`Amazon.RuntimeAWSCredentials`** 인스턴스를 반환합니다. 임시 자격 증명을 요청하는 데 사용되는 자격 증명은 현재 쉘 기본값에서 유추됩니다. 다른 자격 증명을 지정하려면 -ProfileName 또는 -AccessKey/-SecretKey 파라미터를 사용합니다.

```
Get-STSSessionToken
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 2: 1시간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 요청에 사용되는 자격 증명은 지정된 프로파일에서 가져옵니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 3: 프로파일 'myprofilename'에 자격 증명이 지정된 계정과 연결된 MFA 디바이스의 식별 번호와 디바이스에서 제공한 값을 사용하여 1시간 동안 유효한 임시 자격 증명이 들어 있는 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetSessionToken을 참조하세요.

`Use-STSRole`

다음 코드 예시는 `Use-STSRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 요청 사용자가 일반적으로 액세스할 수 없는 AWS 리소스에 액세스하는 데 1시간 동안 사용할 수 있는 임시 자격 증명(액세스 키, 보안 키 및 세션 토큰) 세트를 반환합니다. 반환된 자격 증명에는 수임 중인 역할의 액세스 정책과 제공된 정책에 의해 허용되는 권한이 있습니다. 제공된 정책을 사용하여 수임 중인 역할의 액세스 정책에 의해 정의된 권한을 초과하는 권한을 부여할 수 없습니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

예제 2: 수임된 역할의 액세스 정책에 정의된 것과 동일한 권한을 갖고 1시간 동안 유효한 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

예제 3: cmdlet을 실행하는 데 사용되는 사용자 자격 증명과 연결된 MFA에서 생성된 토큰과 일련 번호를 제공하는 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

예제 4: 고객 계정에 정의된 역할을 수임한 임시 자격 증명 세트를 반환합니다. 타사에서 수임할 수 있는 각 역할에 대해 고객 계정은 역할이 수임될 때마다 -ExternalID 파라미터로 전달되는 식별자를 사용하여 역할을 생성해야 합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssumeRole을 참조하세요.

`Use-STSWebIdentityRole`

다음 코드 예시는 `Use-STSWebIdentityRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Login with Amazon ID 제공업체를 통해 인증된 사용자에 대해 1시간 동안 유효한 임시 자격 증명 세트를 반환합니다. 자격 증명은 역할 ARN으로 식별된 역할과 연결된 액세스 정책을 수임합니다. 필요에 따라 액세스 권한을 더욱 세분화하는 -Policy 파라미터에 JSON 정책을 전달할 수 있습니다. 역할과 연결된 권한에서 사용 가능한 것보다 더 많은 권한을 부여할 수는 없습니다. -WebIdentityToken에 제공되는 값은 ID 제공업체가 반환한 고유한 사용자 식별자입니다.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssumeRoleWithWebIdentity를 참조하세요.

지원 Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 지원.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`Add-ASACommunicationToCase`

다음 코드 예시는 `Add-ASACommunicationToCase`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이메일 커뮤니케이션의 본문을 지정된 사례에 추가합니다.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

예제 2: 지정된 사례에 이메일 커뮤니케이션 본문을 추가하고 이메일의 CC 행에 포함된 하나 이상의 이메일 주소를 추가합니다.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddCommunicationToCase를 참조하세요.

`Get-ASACase`

다음 코드 예시는 `Get-ASACase`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 모든 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase
```

예제 2: 지정된 날짜 및 시간 이후의 모든 지원 사례에 대한 세부 정보를 반환합니다.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

예제 3: 해결된 지원 사례를 포함하여 처음 10개의 지원 사례에 대한 세부 정보를 반환합니다.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

예제 4: 지정된 지원 사례 하나의 세부 정보를 반환합니다.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

예제 5: 지정된 여러 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCases를 참조하세요.

`Get-ASACommunication`

다음 코드 예시는 `Get-ASACommunication`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 사례에 대한 모든 커뮤니케이션을 반환합니다.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

예제 2: 지정된 사례에 대해 2012년 1월 1일 자정(UTC) 이후의 모든 커뮤니케이션을 반환합니다.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeCommunications을 참조하세요.

`Get-ASAService`

다음 코드 예시는 `Get-ASAService`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 사용 가능한 모든 서비스 코드, 이름, 범주를 반환합니다.

```
Get-ASAService
```

예제 2: 지정된 코드가 있는 서비스의 이름과 범주를 반환합니다.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

예제 3: 지정된 서비스 코드의 이름과 범주를 반환합니다.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

예제 4: 지정된 서비스 코드의 이름과 범주(일본어)를 반환합니다. 현재 영어('en') 및 일본어('ja') 언어 코드가 지원됩니다.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeServices를 참조하세요.

`Get-ASASeverityLevel`

다음 코드 예시는 `Get-ASASeverityLevel`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: AWS 지원 사례에 할당할 수 있는 심각도 수준 목록을 반환합니다.

```
Get-ASASeverityLevel
```

예제 2: AWS 지원 사례에 할당할 수 있는 심각도 수준 목록을 반환합니다. 수준 이름은 일본어로 반환됩니다.

```
Get-ASASeverityLevel -Language "ja"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeSeverityLevels을 참조하세요.

`Get-ASATrustedAdvisorCheck`

다음 코드 예시는 `Get-ASATrustedAdvisorCheck`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Trusted Advisor 검사 컬렉션을 반환합니다. Language 파라미터를 반드시 지정해야 하며 영어로 출력하려면 "en", 일본어로 출력하려면 "ja" 값을 사용합니다.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrustedAdvisorChecks를 참조하세요.

`Get-ASATrustedAdvisorCheckRefreshStatus`

다음 코드 예시는 `Get-ASATrustedAdvisorCheckRefreshStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 검사에 대한 새로 고침 요청의 현재 상태를 반환합니다. Request-ASATrustedAdvisorCheckRefresh를 사용하여 검사의 상태 정보를 새로 고치도록 요청할 수 있습니다.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrustedAdvisorCheckRefreshStatuses를 참조하세요.

`Get-ASATrustedAdvisorCheckResult`

다음 코드 예시는 `Get-ASATrustedAdvisorCheckResult`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: Trusted Advisor 검사 결과를 반환합니다. 사용 가능한 Trusted Advisor 검사 목록은 Get-ASATrustedAdvisorChecks를 사용하여 얻을 수 있습니다. 출력은 검사의 전체 상태, 검사가 마지막으로 실행된 타임스탬프, 특정 검사에 대한 고유한 checkid입니다. 결과를 일본어로 출력하려면 -Language 'ja' 파라미터를 추가합니다.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrustedAdvisorCheckResult를 참조하세요.

`Get-ASATrustedAdvisorCheckSummary`

다음 코드 예시는 `Get-ASATrustedAdvisorCheckSummary`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 하나의 Trusted Advisor 검사에 대한 최신 요약을 반환합니다.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

예제 2: 지정된 여러 Trusted Advisor 검사에 대한 최신 요약을 반환합니다.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTrustedAdvisorCheckSummaries를 참조하세요.

`New-ASACase`

다음 코드 예시는 `New-ASACase`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: AWS 지원 센터에서 새 사례를 생성합니다. Get-ASAService cmdlet을 사용하여 -ServiceCode 및 -CategoryCode 파라미터 값을 얻을 수 있습니다. Get-ASASeverityLevel cmdlet을 사용하여 -SeverityCode 파라미터 값을 얻을 수 있습니다. -IssueType 파라미터 값은 'customer-service' 또는 'technical' 중 하나입니다. 성공하면 AWS 지원 사례 번호가 출력됩니다. 기본적으로 사례는 영어로 처리되며 일본어를 사용하려면 -Language 'ja' 파라미터를 추가해야 합니다. -ServiceCode, -CategoryCode, -Subject, -CommunicationBody 파라미터는 필수입니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateCase를 참조하세요.

`Request-ASATrustedAdvisorCheckRefresh`

다음 코드 예시는 `Request-ASATrustedAdvisorCheckRefresh`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 Trusted Advisor 검사에 대한 새로 고침을 요청합니다.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RefreshTrustedAdvisorCheck를 참조하세요.

`Resolve-ASACase`

다음 코드 예시는 `Resolve-ASACase`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 사례의 초기 상태와 이를 해결하기 위한 직접 호출이 완료된 후의 현재 상태를 반환합니다.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ResolveCase를 참조하세요.

Tools for PowerShell V4를 사용한 Systems Manager 예제

다음 코드 예제에서는 Systems Manager와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Add-SSMResourceTag`

다음 코드 예시는 `Add-SSMResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간을 새 태그로 업데이트합니다. 명령이 성공해도 출력은 없습니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

예제 2: PowerShell 버전 2에서 각 태그를 생성하려면 New-Object를 사용해야 합니다. 명령이 성공해도 출력은 없습니다.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AddTagsToResource를 참조하세요.

`Edit-SSMDocumentPermission`

다음 코드 예시는 `Edit-SSMDocumentPermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서의 모든 계정에 '공유' 권한을 추가합니다. 명령이 성공해도 출력은 없습니다.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

예제 2: 이 예제에서는 문서의 특정 계정에 '공유' 권한을 추가합니다. 명령이 성공해도 출력은 없습니다.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyDocumentPermission을 참조하세요.

Get-SSMActivation

다음 코드 예시는 Get-SSMActivation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 계정 활성화에 대한 세부 정보를 제공합니다.

```
Get-SSMActivation
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeActivations를 참조하세요.

Get-SSMAssociation

다음 코드 예시는 Get-SSMAssociation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스와 문서 간 연결을 설명합니다.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAssociation을 참조하세요.

`Get-SSMAssociationExecution`

다음 코드 예시는 `Get-SSMAssociationExecution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 제공된 연결 ID에 대한 실행을 반환합니다.

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAssociationExecutions를 참조하세요.

`Get-SSMAssociationExecutionTarget`

다음 코드 예시는 `Get-SSMAssociationExecutionTarget`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 연결 실행 대상의 일부인 리소스 ID와 해당 실행 상태를 표시합니다.

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

출력:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

예제 2: 이 명령은 어제 이후 명령 문서가 연결된 명령 자동화의 특정 실행을 확인합니다. 연결 실행이 실패했는지 추가로 확인하고, 실패한 경우 인스턴스 ID와 함께 실행에 대한 명령 간접 호출 세부 정보를 표시합니다.

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.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
    }
}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAssociationExecutionTargets를 참조하세요.

`Get-SSMAssociationList`

다음 코드 예시는 `Get-SSMAssociationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스의 모든 연결을 나열합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

출력:

```
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}
```

예제 2: 이 예제에서는 구성 문서의 모든 연결을 나열합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

출력:

```
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}
```

예제 3: PowerShell 버전 2에서 각 필터를 생성하려면 New-Object를 사용해야 합니다.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAssociations를 참조하세요.

Get-SSMAssociationVersionList

다음 코드 예시는 Get-SSMAssociationVersionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 제공된 연결의 모든 버전을 검색합니다.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListAssociationVersions를 참조하세요.

Get-SSMAutomationExecution

다음 코드 예시는 Get-SSMAutomationExecution의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 자동화 실행의 세부 정보를 표시합니다.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

출력:

```
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...}
```

예제 2: 이 예제에서는 지정된 자동화 실행 ID에 대한 단계 세부 정보를 나열합니다.

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetAutomationExecution을 참조하세요.

Get-SSMAutomationExecutionList

다음 코드 예시는 Get-SSMAutomationExecutionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 계정과 연결된 모든 활성 및 종료된 자동화 실행을 설명합니다.

```
Get-SSMAutomationExecutionList
```

출력:

```
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]]}
```

예제 2: 이 예제에서는 자동화 실행 상태가 '성공'이 아닌 실행의 ExecutionID, 문서, 실행 시작 및 종료 타임스탬프를 표시합니다.

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAutomationExecutions를 참조하세요.

`Get-SSMAutomationStepExecution`

다음 코드 예시는 `Get-SSMAutomationStepExecution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 자동화 워크플로에서 모든 활성 및 종료된 단계 실행에 대한 정보를 표시합니다.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAutomationStepExecutions를 참조하세요.

`Get-SSMAvailablePatch`

다음 코드 예시는 `Get-SSMAvailablePatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 MSRC 심각도가 위험인 Windows Server 2012에서 사용 가능한 모든 패치를 가져옵니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

출력:

```
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
...
```

예제 2: PowerShell 버전 2에서 각 필터를 생성하려면 New-Object를 사용해야 합니다.

```
$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
```

예제 3: 이 예제에서는 지난 20일 동안 릴리스되고 WindowsServer2019와 일치하는 제품에 적용할 수 있는 모든 업데이트를 가져옵니다.

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeAvailablePatches를 참조하세요.

`Get-SSMCommand`

다음 코드 예시는 `Get-SSMCommand`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 요청된 모든 명령을 나열합니다.

```
Get-SSMCommand
```

출력:

```
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            : {}
```

예제 2: 이 예제는 특정 명령의 상태를 가져옵니다.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

예제 3: 이 예제에서는 2019-04-01T00:00:00Z 이후에 간접 호출된 모든 SSM 명령을 검색합니다.

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListCommands를 참조하세요.

`Get-SSMCommandInvocation`

다음 코드 예시는 `Get-SSMCommandInvocation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 명령의 모든 간접 호출을 나열합니다.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

출력:

```
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        :
```

예제 2: 이 예제에서는 명령 ID e1eb2e3c-ed4c-5123-45c1-234f5612345f의 간접 호출에 대한 CommandPlugins를 나열합니다.

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListCommandInvocations를 참조하세요.

Get-SSMCommandInvocationDetail

다음 코드 예시는 Get-SSMCommandInvocationDetail의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스에서 실행된 명령의 세부 정보를 표시합니다.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetCommandInvocation을 참조하세요.

Get-SSMComplianceItemList

다음 코드 예시는 Get-SSMComplianceItemList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 리소스 ID 및 유형에 대한 규정 준수 항목 목록을 나열하며, 이때 필터링 규정 준수 유형은 '연결'로 필터링됩니다.

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListComplianceItems를 참조하세요.

`Get-SSMComplianceSummaryList`

다음 코드 예시는 `Get-SSMComplianceSummaryList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 규정 준수 유형에 대한 규정 준수 및 규정 미준수 리소스의 요약 개수를 반환합니다.

```
Get-SSMComplianceSummaryList
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListComplianceSummaries를 참조하세요.

`Get-SSMConnectionStatus`

다음 코드 예시는 `Get-SSMConnectionStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스의 세션 관리자 연결 상태를 검색하여 인스턴스가 연결되어 있고 세션 관리자 연결을 수신할 준비가 되었는지 확인합니다.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

출력:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetConnectionStatus를 참조하세요.

`Get-SSMDefaultPatchBaseline`

다음 코드 예시는 `Get-SSMDefaultPatchBaseline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기본 패치 기준을 표시합니다.

```
Get-SSMDefaultPatchBaseline
```

출력:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDefaultPatchBaseline을 참조하세요.

`Get-SSMDeployablePatchSnapshotForInstance`

다음 코드 예시는 `Get-SSMDeployablePatchSnapshotForInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스에서 사용하는 패치 기준의 현재 스냅샷을 표시합니다. 이 명령은 인스턴스 자격 증명을 사용하여 인스턴스에서 실행해야 합니다. 이 예제에서는 인스턴스 자격 증명을 사용하는지 확인하기 위해 자격 증명 파라미터에 **`Amazon.Runtime.InstanceProfileAWSCredentials`** 객체를 전달합니다.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

출력:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

예제 2: 이 예제에서는 전체 SnapshotDownloadUrl을 가져오는 방법을 보여줍니다. 이 명령은 인스턴스 자격 증명을 사용하여 인스턴스에서 실행해야 합니다. 인스턴스 자격 증명을 사용하는지 확인하기 위해 이 예제에서는 **`Amazon.Runtime.InstanceProfileAWSCredentials`** 객체를 사용하도록 PowerShell 세션을 구성합니다.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
```

```
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

출력:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDeployablePatchSnapshotForInstance를 참조하세요.

`Get-SSMDocument`

다음 코드 예시는 `Get-SSMDocument`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서의 콘텐츠를 반환합니다.

```
Get-SSMDocument -Name "RunShellScript"
```

출력:

```
Content
-------
{...
```

예제 2: 이 예제에서는 문서의 전체 콘텐츠를 표시합니다.

```
(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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetDocument를 참조하세요.

`Get-SSMDocumentDescription`

다음 코드 예시는 `Get-SSMDocumentDescription`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서에 대한 정보를 반환합니다.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDocument를 참조하세요.

`Get-SSMDocumentList`

다음 코드 예시는 `Get-SSMDocumentList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 계정의 모든 구성 문서를 나열합니다.

```
Get-SSMDocumentList
```

출력:

```
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
```

```
...
```

예제 2: 이 예제에서는 이름이 'Platform'과 일치하는 모든 자동화 문서를 검색합니다.

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

출력:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDocuments를 참조하세요.

`Get-SSMDocumentPermission`

다음 코드 예시는 `Get-SSMDocumentPermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서의 모든 버전을 나열합니다.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

출력:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeDocumentPermission을 참조하세요.

Get-SSMDocumentVersionList

다음 코드 예시는 Get-SSMDocumentVersionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서의 모든 버전을 나열합니다.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

출력:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListDocumentVersions를 참조하세요.

Get-SSMEffectiveInstanceAssociationList

다음 코드 예시는 Get-SSMEffectiveInstanceAssociationList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스에 대한 유효한 연결을 설명합니다.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

출력:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

예제 2: 이 예제에서는 인스턴스에 대한 유효한 연결의 콘텐츠를 설명합니다.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

출력:

```
{
    "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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEffectiveInstanceAssociations를 참조하세요.

Get-SSMEffectivePatchesForPatchBaseline

다음 코드 예시는 Get-SSMEffectivePatchesForPatchBaseline의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 최대 결과 목록이 1인 모든 패치 기준을 나열합니다.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

출력:

```
Patch                                          PatchStatus
-----                                          -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

예제 2: 이 예제에서는 최대 결과 목록이 1인 모든 패치 기준의 패치 상태를 표시합니다.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

출력:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeEffectivePatchesForPatchBaseline을 참조하세요.

Get-SSMInstanceAssociationsStatus

다음 코드 예시는 Get-SSMInstanceAssociationsStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 연결의 세부 정보를 보여줍니다.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 인스턴스 ID의 인스턴스 연결 상태를 확인하고 더 나아가 해당 연결의 실행 상태를 표시합니다.

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstanceAssociationsStatus를 참조하세요.

Get-SSMInstanceInformation

다음 코드 예시는 Get-SSMInstanceInformation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 각 인스턴스의 세부 정보를 보여줍니다.

```
Get-SSMInstanceInformation
```

출력:

```
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
```

예제 2:이 예제에서는 -Filter 파라미터를 사용하여 **AgentVersion**가 **us-east-1**인 리전의 AWS Systems Manager 인스턴스로만 결과를 필터링하는 방법을 보여줍니다**2.2.800.0**. InstanceInformation API 참조 주제(https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager-Type-InstanceInformation-ActivationId)에서 유효한 -Filter 키 값 목록을 찾을 수 있습니다.

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
```

```
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

출력:

```
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
```

예제 3:이 예제에서는 -InstanceInformationFilterList 파라미터를 사용하여 **Windows** 또는 **PlatformTypes**가 **us-east-1** 있는 리전의 AWS Systems Manager 인스턴스로만 결과를 필터링하는 방법을 보여줍니다**Linux**. InstanceInformationFilter API 참조 주제(https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html)에서 유효한 -InstanceInformationFilterList 키 값 목록을 찾을 수 있습니다.

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

출력:

```
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
```

예제 4: 이 예제에서는 ssm 관리형 인스턴스를 나열하고 InstanceId, PingStatus, LastPingDateTime 및 PlatformName을 csv 파일로 내보냅니다.

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstanceInformation을 참조하세요.

`Get-SSMInstancePatch`

다음 코드 예시는 `Get-SSMInstancePatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스에 대한 패치 규정 준수 세부 정보를 가져옵니다.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstancePatches를 참조하세요.

Get-SSMInstancePatchState

다음 코드 예시는 Get-SSMInstancePatchState의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스의 패치 요약 상태를 가져옵니다.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

예제 2: 이 예제에서는 두 인스턴스의 패치 요약 상태를 가져옵니다.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstancePatchStates를 참조하세요.

Get-SSMInstancePatchStatesForPatchGroup

다음 코드 예시는 Get-SSMInstancePatchStatesForPatchGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 그룹의 인스턴스당 패치 요약 상태를 가져옵니다.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeInstancePatchStatesForPatchGroup을 참조하세요.

Get-SSMInventory

다음 코드 예시는 Get-SSMInventory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인벤토리의 사용자 지정 메타데이터를 가져옵니다.

```
Get-SSMInventory
```

출력:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetInventory를 참조하세요.

`Get-SSMInventoryEntriesList`

다음 코드 예시는 `Get-SSMInventoryEntriesList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스의 모든 사용자 지정 인벤토리 항목을 나열합니다.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

출력:

```
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
```

예제 2: 이 예제에서는 세부 정보를 나열합니다.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

출력:

```
Key          Value
```

```
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListInventoryEntries를 참조하세요.

Get-SSMInventoryEntryList

다음 코드 예시는 Get-SSMInventoryEntryList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스의 **AWS:Network** 유형 인벤토리 항목을 검색합니다.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Get-SSMInventoryEntryList를 참조하세요.

Get-SSMInventorySchema

다음 코드 예시는 Get-SSMInventorySchema의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 계정의 인벤토리 유형 이름 목록을 반환합니다.

```
Get-SSMInventorySchema
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetInventorySchema를 참조하세요.

`Get-SSMLatestEC2Image`

다음 코드 예시는 `Get-SSMLatestEC2Image`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 최신 Windows AMI를 나열합니다.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

출력:

```
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
...
```

예제 2: 이 예제에서는 us-west-2 리전에 대한 특정 Amazon Linux 이미지의 AMI ID를 검색합니다.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

출력:

```
ami-09b92cd132204c704
```

예제 3: 이 예제에서는 지정된 와일드카드 표현식과 일치하는 모든 최신 Windows AMI를 나열합니다.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 Get-SSMLatestEC2Image를 참조하세요.

`Get-SSMMaintenanceWindow`

다음 코드 예시는 `Get-SSMMaintenanceWindow`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간에 대한 세부 정보를 가져옵니다.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetMaintenanceWindow를 참조하세요.

Get-SSMMaintenanceWindowExecution

다음 코드 예시는 Get-SSMMaintenanceWindowExecution의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간 실행의 일부로 실행된 작업에 대한 정보를 나열합니다.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetMaintenanceWindowExecution을 참조하세요.

Get-SSMMaintenanceWindowExecutionList

다음 코드 예시는 Get-SSMMaintenanceWindowExecutionList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간의 모든 실행을 나열합니다.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

출력:

```
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
```

예제 2: 이 예제에서는 지정된 날짜 이전에 유지 관리 기간의 모든 실행을 나열합니다.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

예제 3: 이 예제에서는 지정된 날짜 이후에 유지 관리 기간의 모든 실행을 나열합니다.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindowExecutions를 참조하세요.

`Get-SSMMaintenanceWindowExecutionTask`

다음 코드 예시는 `Get-SSMMaintenanceWindowExecutionTask`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간 실행의 일부였던 작업에 대한 정보를 나열합니다.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetMaintenanceWindowExecutionTask를 참조하세요.

`Get-SSMMaintenanceWindowExecutionTaskInvocationList`

다음 코드 예시는 `Get-SSMMaintenanceWindowExecutionTaskInvocationList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간 실행의 일부로 실행된 작업에 대한 간접 호출을 나열합니다.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindowExecutionTaskInvocations를 참조하세요.

Get-SSMMaintenanceWindowExecutionTaskList

다음 코드 예시는 Get-SSMMaintenanceWindowExecutionTaskList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간 실행과 연결된 작업을 나열합니다.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindowExecutionTasks를 참조하세요.

Get-SSMMaintenanceWindowList

다음 코드 예시는 Get-SSMMaintenanceWindowList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 계정의 모든 유지 관리 기간을 나열합니다.

```
Get-SSMMaintenanceWindowList
```

출력:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindows를 참조하세요.

Get-SSMMaintenanceWindowTarget

다음 코드 예시는 Get-SSMMaintenanceWindowTarget의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간의 모든 대상을 나열합니다.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindowTargets를 참조하세요.

Get-SSMMaintenanceWindowTaskList

다음 코드 예시는 Get-SSMMaintenanceWindowTaskList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간의 모든 작업을 나열합니다.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeMaintenanceWindowTasks를 참조하세요.

`Get-SSMParameterHistory`

다음 코드 예시는 `Get-SSMParameterHistory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 파라미터 값 기록을 나열합니다.

```
Get-SSMParameterHistory -Name "Welcome"
```

출력:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetParameterHistory를 참조하세요.

Get-SSMParameterList

다음 코드 예시는 Get-SSMParameterList의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 파라미터를 나열합니다.

```
Get-SSMParameterList
```

출력:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeParameters를 참조하세요.

Get-SSMParameterValue

다음 코드 예시는 Get-SSMParameterValue의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 파라미터 값을 나열합니다.

```
Get-SSMParameterValue -Name "Welcome"
```

출력:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

예제 2: 이 예제에서는 값의 세부 정보를 나열합니다.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

출력:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetParameters를 참조하세요.

Get-SSMPatchBaseline

다음 코드 예시는 Get-SSMPatchBaseline의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 모든 패치 기준을 나열합니다.

```
Get-SSMPatchBaseline
```

출력:

```
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...
```

예제 2:이 예제에서는에서 제공하는 모든 패치 기준을 나열합니다 AWS. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

출력:

```
Get-SSMPatchBaseline -Filter $filter1
```

예제 3: 이 예제에서는 사용자가 소유자인 모든 패치 기준을 나열합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

출력:

```
Get-SSMPatchBaseline -Filter $filter1
```

예제 4: PowerShell 버전 2에서 각 태그를 생성하려면 New-Object를 사용해야 합니다.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePatchBaselines를 참조하세요.

`Get-SSMPatchBaselineDetail`

다음 코드 예시는 `Get-SSMPatchBaselineDetail`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 기준의 세부 정보를 표시합니다.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPatchBaseline을 참조하세요.

`Get-SSMPatchBaselineForPatchGroup`

다음 코드 예시는 `Get-SSMPatchBaselineForPatchGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 그룹의 패치 기준을 표시합니다.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

출력:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 GetPatchBaselineForPatchGroup을 참조하세요.

`Get-SSMPatchGroup`

다음 코드 예시는 `Get-SSMPatchGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 그룹 등록을 나열합니다.

```
Get-SSMPatchGroup
```

출력:

```
BaselineIdentity                                                PatchGroup
----------------                                                ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePatchGroups를 참조하세요.

`Get-SSMPatchGroupState`

다음 코드 예시는 `Get-SSMPatchGroupState`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 그룹에 대한 개요 수준의 패치 규정 준수 요약을 가져옵니다.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

출력:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribePatchGroupState를 참조하세요.

`Get-SSMResourceComplianceSummaryList`

다음 코드 예시는 `Get-SSMResourceComplianceSummaryList`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 리소스 수준 요약 수를 가져옵니다. 요약에는 'Windows10'과 일치하는 제품의 규정 준수 및 비준수 상태에 대한 정보와 자세한 규정 준수 항목의 심각도 수가 포함됩니다. 파라미터가 지정되지 않은 경우 MaxResult의 기본값은 100이지만 이 값은 유효하지 않으므로 MaxResult 파라미터가 추가되고 값은 50으로 설정됩니다.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListResourceComplianceSummaries를 참조하세요.

`Get-SSMResourceTag`

다음 코드 예시는 `Get-SSMResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간의 태그를 나열합니다.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

출력:

```
Key   Value
---   -----
Stack Production
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ListTagsForResource를 참조하세요.

`New-SSMActivation`

다음 코드 예시는 `New-SSMActivation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 관리형 인스턴스를 생성합니다.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

출력:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateActivation을 참조하세요.

New-SSMAssociation

다음 코드 예시는 New-SSMAssociation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 ID를 사용하여 구성 문서를 인스턴스와 연결합니다.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

출력:

```
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 :
```

예제 2: 이 예제에서는 대상을 사용하여 구성 문서를 인스턴스와 연결합니다.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

출력:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

예제 3: 이 예제는 대상과 파라미터를 사용하여 구성 문서를 인스턴스와 연결합니다.

```
$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
```

출력:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

예제 4: 이 예제에서는 **AWS-GatherSoftwareInventory**를 사용하여 리전에 있는 모든 인스턴스와의 연결을 생성합니다. 또한 파라미터에서 수집할 사용자 지정 파일 및 레지스트리 위치도 제공합니다.

```
$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)"
```

출력:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAssociation을 참조하세요.

`New-SSMAssociationFromBatch`

다음 코드 예시는 `New-SSMAssociationFromBatch`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 구성 문서를 여러 인스턴스와 연결합니다. 출력은 해당하는 경우 성공한 작업과 실패한 작업의 목록을 반환합니다.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

출력:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

예제 2: 이 예제에서는 성공한 작업의 전체 세부 정보를 보여줍니다.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateAssociationBatch를 참조하세요.

`New-SSMDocument`

다음 코드 예시는 `New-SSMDocument`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제는 계정에서 문서를 생성합니다. 문서는 JSON 형식이어야 합니다. 구성 문서 작성에 대한 자세한 내용은 SSM API 참조의 구성 문서를 참조하세요.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateDocument를 참조하세요.

New-SSMMaintenanceWindow

다음 코드 예시는 New-SSMMaintenanceWindow의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 이름의 새 유지 관리 기간을 생성합니다. 이 유지 관리 기간은 매주 화요일 오후 4시에 4시간 동안 실행되며, 마감 시간은 1시간이고, 연결되지 않은 대상을 허용합니다.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

출력:

```
mw-03eb53e1ea7383998
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateMaintenanceWindow를 참조하세요.

New-SSMPatchBaseline

다음 코드 예시는 New-SSMPatchBaseline의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 Microsoft에서 릴리스하고 7일 후에 프로덕션 환경에서 Windows Server 2019를 실행하는 관리형 인스턴스에 대한 패치를 승인하는 패치 기준을 생성합니다.

```
$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
```

출력:

```
pb-0z4z6221c4296b23z
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreatePatchBaseline을 참조하세요.

`Register-SSMDefaultPatchBaseline`

다음 코드 예시는 `Register-SSMDefaultPatchBaseline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 기준을 기본 패치 기준으로 등록합니다.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

출력:

```
pb-03da896ca3b68b639
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterDefaultPatchBaseline을 참조하세요.

Register-SSMPatchBaselineForPatchGroup

다음 코드 예시는 Register-SSMPatchBaselineForPatchGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 그룹의 패치 기준을 등록합니다.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

출력:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterPatchBaselineForPatchGroup을 참조하세요.

Register-SSMTargetWithMaintenanceWindow

다음 코드 예시는 Register-SSMTargetWithMaintenanceWindow의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간에 한 인스턴스를 등록합니다.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

출력:

```
d8e47760-23ed-46a5-9f28-927337725398
```

예제 2: 이 예제에서는 유지 관리 기간에 여러 인스턴스를 등록합니다.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
```

```
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

출력:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

예제 3: 이 예제에서는 EC2 태그를 사용하여 유지 관리 기간에 인스턴스를 등록합니다.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

출력:

```
2994977e-aefb-4a71-beac-df620352f184
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterTargetWithMaintenanceWindow를 참조하세요.

`Register-SSMTaskWithMaintenanceWindow`

다음 코드 예시는 `Register-SSMTaskWithMaintenanceWindow`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 ID를 사용하여 유지 관리 기간에 작업을 등록합니다. 출력은 작업 ID입니다.

```
$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
```

출력:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

예제 2: 이 예제에서는 대상 ID를 사용하여 유지 관리 기간에 작업을 등록합니다. 출력은 작업 ID입니다.

```
$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
```

출력:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

예제 3: 이 예제에서는 Run Command 문서 **AWS-RunPowerShellScript**에 대한 파라미터 객체를 생성하고 대상 ID를 사용하여 지정된 유지 관리 기간을 포함하는 작업을 생성합니다. 반환 출력은 작업 ID입니다.

```
$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
```

출력:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

예제 4:이 예제에서는 라는 문서를 사용하여 AWS Systems Manager 자동화 작업을 등록합니다**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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterTaskWithMaintenanceWindow를 참조하세요.

`Remove-SSMActivation`

다음 코드 예시는 `Remove-SSMActivation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 활성화를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteActivation을 참조하세요.

Remove-SSMAssociation

다음 코드 예시는 Remove-SSMAssociation의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스와 문서 간의 연결을 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteAssociation을 참조하세요.

Remove-SSMDocument

다음 코드 예시는 Remove-SSMDocument의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 문서를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMDocument -Name "RunShellScript"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteDocument를 참조하세요.

Remove-SSMMaintenanceWindow

다음 코드 예시는 Remove-SSMMaintenanceWindow의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간을 제거합니다.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

출력:

```
mw-06d59c1a07c022145
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteMaintenanceWindow를 참조하세요.

`Remove-SSMParameter`

다음 코드 예시는 `Remove-SSMParameter`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 파라미터를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMParameter -Name "helloWorld"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteParameter를 참조하세요.

`Remove-SSMPatchBaseline`

다음 코드 예시는 `Remove-SSMPatchBaseline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 기준을 삭제합니다.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

출력:

```
pb-045f10b4f382baeda
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeletePatchBaseline을 참조하세요.

`Remove-SSMResourceTag`

다음 코드 예시는 `Remove-SSMResourceTag`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간에서 태그를 제거합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RemoveTagsFromResource를 참조하세요.

Send-SSMCommand

다음 코드 예시는 Send-SSMCommand의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 대상 인스턴스에서 echo 명령을 실행합니다.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

출력:

```
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}
```

예제 2: 이 예제에서는 중첩된 파라미터를 수락하는 명령을 실행하는 방법을 보여줍니다.

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 SendCommand를 참조하세요.

`Start-SSMAutomationExecution`

다음 코드 예시는 `Start-SSMAutomationExecution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 자동화 역할, AMI 소스 ID 및 Amazon EC2 인스턴스 역할을 지정하는 문서를 실행합니다.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

출력:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartAutomationExecution을 참조하세요.

`Start-SSMSession`

다음 코드 예시는 `Start-SSMSession`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예 1: 이 예제는 Session Manager 세션의 대상에 대한 연결을 시작하여 포트 전달을 활성화합니다.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

출력:

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StartSession을 참조하세요.

`Stop-SSMAutomationExecution`

다음 코드 예시는 `Stop-SSMAutomationExecution`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 자동화 실행을 중지합니다. 명령이 성공해도 출력은 없습니다.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopAutomationExecution을 참조하세요.

`Stop-SSMCommand`

다음 코드 예시는 `Stop-SSMCommand`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 명령 취소를 시도합니다. 작업이 성공해도 출력은 없습니다.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CancelCommand를 참조하세요.

`Unregister-SSMManagedInstance`

다음 코드 예시는 `Unregister-SSMManagedInstance`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 관리형 인스턴스를 등록 취소합니다. 명령이 성공해도 출력은 없습니다.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterManagedInstance를 참조하세요.

`Unregister-SSMPatchBaselineForPatchGroup`

다음 코드 예시는 `Unregister-SSMPatchBaselineForPatchGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 패치 기준에서 패치 그룹을 등록 취소합니다.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

출력:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterPatchBaselineForPatchGroup을 참조하세요.

`Unregister-SSMTargetFromMaintenanceWindow`

다음 코드 예시는 `Unregister-SSMTargetFromMaintenanceWindow`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간에서 대상을 제거합니다.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

출력:

```
WindowId             WindowTargetId
```

```
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterTargetFromMaintenanceWindow를 참조하세요.

`Unregister-SSMTaskFromMaintenanceWindow`

다음 코드 예시는 `Unregister-SSMTaskFromMaintenanceWindow`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간에서 작업을 제거합니다.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

출력:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeregisterTaskFromMaintenanceWindow를 참조하세요.

`Update-SSMAssociation`

다음 코드 예시는 `Update-SSMAssociation`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 연결을 새 문서 버전으로 업데이트합니다.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

출력:

```
Name                  : AWS-UpdateSSMAgent
```

```
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAssociation을 참조하세요.

Update-SSMAssociationStatus

다음 코드 예시는 Update-SSMAssociationStatus의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스 및 구성 문서 간 연결 상태를 업데이트합니다.

```
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"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateAssociationStatus를 참조하세요.

Update-SSMDocument

다음 코드 예시는 Update-SSMDocument의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정한 json 파일의 업데이트된 콘텐츠를 포함하는 문서의 새 버전을 생성합니다. 문서는 JSON 형식이어야 합니다. 'Get-SSMDocumentVersionList' cmdlet을 사용하여 문서 버전을 얻을 수 있습니다.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateDocument를 참조하세요.

`Update-SSMDocumentDefaultVersion`

다음 코드 예시는 `Update-SSMDocumentDefaultVersion`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 여기에서는 문서의 기본 버전을 업데이트합니다. 'Get-SSMDocumentVersionList' cmdlet을 사용하여 사용 가능한 문서 버전을 얻을 수 있습니다.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

출력:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateDocumentDefaultVersion을 참조하세요.

Update-SSMMaintenanceWindow

다음 코드 예시는 Update-SSMMaintenanceWindow의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 유지 관리 기간의 이름을 업데이트합니다.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

출력:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

예제 2: 이 예제에서는 유지 관리 기간을 활성화합니다.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

출력:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

예제 3: 이 예제에서는 유지 관리 기간을 비활성화합니다.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

출력:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateMaintenanceWindow를 참조하세요.

`Update-SSMManagedInstanceRole`

다음 코드 예시는 `Update-SSMManagedInstanceRole`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 관리형 인스턴스의 역할을 업데이트합니다. 명령이 성공해도 출력은 없습니다.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdateManagedInstanceRole을 참조하세요.

`Update-SSMPatchBaseline`

다음 코드 예시는 `Update-SSMPatchBaseline`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 기존 패치 기준에 두 개의 패치를 거부된 패치로 추가하고 한 개의 패치를 승인된 패치로 추가합니다.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 UpdatePatchBaseline을 참조하세요.

`Write-SSMComplianceItem`

다음 코드 예시는 `Write-SSMComplianceItem`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 지정된 관리형 인스턴스에 대한 사용자 지정 규정 준수 항목을 작성합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutComplianceItems를 참조하세요.

`Write-SSMInventory`

다음 코드 예시는 `Write-SSMInventory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 인스턴스에 랙 위치 정보를 할당합니다. 명령이 성공해도 출력은 없습니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutInventory를 참조하세요.

`Write-SSMParameter`

다음 코드 예시는 `Write-SSMParameter`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 파라미터를 생성합니다. 명령이 성공해도 출력은 없습니다.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

예제 2: 이 예제에서는 파라미터를 변경합니다. 명령이 성공해도 출력은 없습니다.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 PutParameter를 참조하세요.

Tools for PowerShell V4를 사용한 Amazon Translate 예제

다음 코드 예제에서는 Amazon Translate에서 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`ConvertTo-TRNTargetLanguage`

다음 코드 예시는 `ConvertTo-TRNTargetLanguage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 영어 텍스트를 프랑스어로 변환합니다. 변환할 텍스트를 -Text 파라미터로 전달할 수도 있습니다.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TranslateText를 참조하세요.

AWS WAFV2 Tools for PowerShell V4를 사용한 예제

다음 코드 예제에서는와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS WAFV2.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

New-WAF2WebACL

다음 코드 예시는 New-WAF2WebACL의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 명령에서는 'waf-test'라는 새 웹 ACL을 생성합니다. 서비스 API 설명서에 따라 'DefaultAction'은 필수 속성입니다. 따라서 '-DefaultAction_Allow' 및/또는 '-DefaultAction_Block' 값을 지정해야 합니다. '-DefaultAction_Allow' 및 '-DefaultAction_Block'은 필수 속성이 아니므로, 위 예제와 같이 '@{}' 값을 자리 표시자로 사용할 수 있습니다.

```
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 @{}
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateWebAcl을 참조하세요.

Tools for PowerShell V4를 사용한 WorkSpaces 예제

다음 코드 예제에서는 WorkSpaces와 함께 AWS Tools for PowerShell V4를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`Approve-WKSIpRule`

다음 코드 예시는 `Approve-WKSIpRule`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 기존 IP 그룹에 규칙을 추가합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AuthorizeIpRules을 참조하세요.

`Copy-WKSWorkspaceImage`

다음 코드 예시는 `Copy-WKSWorkspaceImage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 ID를 가진 워크스페이스 이미지를 us-west-2에서 이름이 'CopiedImageTest'인 현재 리전으로 복사합니다.

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

출력:

```
wsi-456abaqfe
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CopyWorkspaceImage를 참조하세요.

`Edit-WKSClientProperty`

다음 코드 예시는 `Edit-WKSClientProperty`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 Workspaces Client에 대한 재연결을 활성화합니다.

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyClientProperties를 참조하세요.

`Edit-WKSSelfServicePermission`

다음 코드 예시는 `Edit-WKSSelfServicePermission`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 디렉터리의 컴퓨팅 유형을 변경하고 볼륨 크기를 늘릴 수 있는 셀프 서비스 권한을 활성화합니다.

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifySelfservicePermissions을 참조하세요.

`Edit-WKSWorkspaceAccessProperty`

다음 코드 예시는 `Edit-WKSWorkspaceAccessProperty`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 디렉터리에 대해 Android 및 Chrome OS에서의 Workspace 액세스를 활성화합니다.

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyWorkspaceAccessProperties를 참조하세요.

`Edit-WKSWorkspaceCreationProperty`

다음 코드 예시는 `Edit-WKSWorkspaceCreationProperty`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 Workspace를 생성하는 동안 인터넷 액세스 및 유지 관리 모드의 기본값을 true로 설정합니다.

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyWorkspaceCreationProperties를 참조하세요.

`Edit-WKSWorkspaceProperty`

다음 코드 예시는 `Edit-WKSWorkspaceProperty`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 Workspace에 대해 Workspace 실행 모드 속성을 자동 중지로 변경합니다

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyWorkspaceProperties를 참조하세요.

`Edit-WKSWorkspaceState`

다음 코드 예시는 `Edit-WKSWorkspaceState`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 Workspace의 상태를 사용 가능 상태로 변경합니다.

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 ModifyWorkspaceState를 참조하세요.

`Get-WKSClientProperty`

다음 코드 예시는 `Get-WKSClientProperty`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 디렉터리에 대한 Workspace Client의 클라이언트 속성을 가져옵니다.

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeClientProperties를 참조하세요.

`Get-WKSIpGroup`

다음 코드 예시는 `Get-WKSIpGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 리전에서 지정된 IP 그룹의 세부 정보를 가져옵니다.

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

출력:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeIpGroups을 참조하세요.

Get-WKSTag

다음 코드 예시는 Get-WKSTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 Workspace에 대한 태그를 가져옵니다.

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

출력:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeTags를 참조하세요.

Get-WKSWorkspace

다음 코드 예시는 Get-WKSWorkspace의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 파이프라인에 대한 모든 WorkSpace의 세부 정보를 검색합니다.

```
Get-WKSWorkspace
```

출력:

```
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
```

예제 2: 이 명령에서는 **us-west-2** 리전의 워크스페이스에 대한 **WorkSpaceProperties**의 하위 속성 값을 보여줍니다. **WorkSpaceProperties**의 하위 속성에 대한 자세한 내용은 https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html 페이지를 참조하세요.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

출력:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

예제 3: 이 명령에서는 **us-west-2** 리전의 워크스페이스에 대한 **WorkSpaceProperties**의 하위 속성인 **RootVolumeSizeGib** 값을 보여줍니다. GiB 단위의 루트 볼륨 크기는 80입니다.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

출력:

```
        80
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspaces를 참조하세요.

`Get-WKSWorkspaceBundle`

다음 코드 예시는 `Get-WKSWorkspaceBundle`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 현재 리전에 있는 모든 Workspace 번들의 세부 정보를 가져옵니다.

```
Get-WKSWorkspaceBundle
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspaceBundles을 참조하세요.

`Get-WKSWorkspaceDirectory`

다음 코드 예시는 `Get-WKSWorkspaceDirectory`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 등록된 디렉터리의 디렉터리 세부 정보를 나열합니다.

```
Get-WKSWorkspaceDirectory
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspaceDirectories를 참조하세요.

`Get-WKSWorkspaceImage`

다음 코드 예시는 `Get-WKSWorkspaceImage`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 리전에 있는 모든 이미지의 모든 세부 정보를 가져옵니다.

```
Get-WKSWorkspaceImage
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspaceImages를 참조하세요.

`Get-WKSWorkspaceSnapshot`

다음 코드 예시는 `Get-WKSWorkspaceSnapshot`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 Workspace에 대해 생성된 가장 최근 스냅샷의 타임스탬프를 보여줍니다.

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

출력:

```
RebuildSnapshots                  RestoreSnapshots
----------------                  ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspaceSnapshots을 참조하세요.

`Get-WKSWorkspacesConnectionStatus`

다음 코드 예시는 `Get-WKSWorkspacesConnectionStatus`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 Workspace의 연결 상태를 가져옵니다.

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DescribeWorkspacesConnectionStatus를 참조하세요.

`New-WKSIpGroup`

다음 코드 예시는 `New-WKSIpGroup`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 FreshEmptyIpGroup이라는 빈 IP 그룹을 생성합니다.

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

출력:

```
wsipg-w45rty4ty
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateIpGroup을 참조하세요.

New-WKSTag

다음 코드 예시는 New-WKSTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 예제에서는 **ws-wsname**이라는 워크스페이스에 새 태그를 추가합니다. 태그의 키는 'Name'이고 키 값은 **AWS_Workspace**입니다.

```
$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
```

예제 2: 이 예제에서는 **ws-wsname**이라는 워크스페이스에 여러 태그를 추가합니다. 한 태그는 키가 'Name'이고 키 값이 **AWS_Workspace**이며, 다른 하나의 태그는 키가 'Stage'이고 키 값이 'Test'입니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateTags를 참조하세요.

New-WKSWorkspace

다음 코드 예시는 New-WKSWorkspace의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 제공된 번들, 디렉터리, 사용자에 대한 WorkSpace를 생성합니다.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

예제 2: 이 예제에서는 여러 WorkSpace를 생성합니다.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 CreateWorkspaces를 참조하세요.

Register-WKSIpGroup

다음 코드 예시는 Register-WKSIpGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 IP 그룹을 지정된 디렉터리에 등록합니다.

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 AssociateIpGroups을 참조하세요.

Register-WKSWorkspaceDirectory

다음 코드 예시는 Register-WKSWorkspaceDirectory의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 Workspaces Service에 지정된 디렉터리를 등록합니다.

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RegisterWorkspaceDirectory를 참조하세요.

Remove-WKSIpGroup

다음 코드 예시는 Remove-WKSIpGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 IP 그룹을 삭제합니다.

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteIpGroup을 참조하세요.

Remove-WKSTag

다음 코드 예시는 Remove-WKSTag의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 Workspace와 연결된 태그를 제거합니다.

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

출력:

```
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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DeleteTags를 참조하세요.

`Remove-WKSWorkspace`

다음 코드 예시는 `Remove-WKSWorkspace`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 여러 WorkSpace를 종료합니다. -Force 스위치를 사용하면 cmdlet이 확인 프롬프트를 표시하지 않습니다.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

예제 2: 모든 WorkSpace 컬렉션을 검색하고 Remove-WKSWorkspace의 -WorkSpaceId 파라미터에 ID를 파이프하여 모든 WorkSpace를 종료합니다. 각 WorkSpace가 종료되기 전에 cmdlet이 프롬프트를 표시합니다. 확인 프롬프트를 차단하려면 -Force 스위치를 추가합니다.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

예제 3: 이 예제에서는 종료할 WorkSpace를 정의하는 TerminateRequest 객체를 전달하는 방법을 보여줍니다. -Force 스위치 파라미터를 지정하지 않으면 cmdlet이 명령을 진행하기 전에 확인을 요청하는 프롬프트를 표시합니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 TerminateWorkspaces를 참조하세요.

`Reset-WKSWorkspace`

다음 코드 예시는 `Reset-WKSWorkspace`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 WorkSpace를 재구축합니다.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

예제 2: 모든 WorkSpace 컬렉션을 검색하고 Reset-WKSWorkspace의 -WorkSpaceId 파라미터에 ID를 파이프하여 WorkSpace를 재구축합니다.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RebuildWorkspaces를 참조하세요.

`Restart-WKSWorkspace`

다음 코드 예시는 `Restart-WKSWorkspace`의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 지정된 WorkSpace를 재부팅합니다.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

예제 2: 여러 WorkSpace를 재부팅합니다.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

예제 3: 모든 WorkSpace 컬렉션을 검색하고 Restart-WKSWorkspace의 -WorkSpaceId 파라미터에 ID를 파이프하여 WorkSpace를 재시작합니다.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 RebootWorkspaces를 참조하세요.

Stop-WKSWorkspace

다음 코드 예시는 Stop-WKSWorkspace의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 여러 WorkSpace를 중지합니다.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

예제 2: 모든 WorkSpace 컬렉션을 검색하고 Stop-WKSWorkspace의 -WorkSpaceId 파라미터에 ID를 파이프하여 WorkSpace를 중지합니다.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

예제 3: 이 예제에서는 중지할 WorkSpace를 정의하는 StopRequest 객체를 전달하는 방법을 보여줍니다.

```
$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 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 StopWorkspaces를 참조하세요.

Unregister-WKSIpGroup

다음 코드 예시는 Unregister-WKSIpGroup의 사용 방법을 보여줍니다.

Tools for PowerShell V4

예제 1: 이 샘플에서는 지정된 디렉터리에서 지정된 IP 그룹의 등록을 취소합니다.

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조(V4)의 DisassociateIpGroups을 참조하세요.

이 AWS 제품 또는 서비스에 대한 보안

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다. 보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 - AWS 는 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 AWS 제공할 책임이 있습니다. 당사의 보안 책임은에서 최우선 순위이며 AWS, 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안의 효과를 정기적으로 테스트하고 검증합니다.

클라우드의 보안 - 사용자의 책임은 사용 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항 및 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

주제

이 AWS 제품 또는 서비스의 데이터 보호

AWS 공동 책임 모델은 이 AWS 제품 또는 서비스의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별

사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요. AWS 활동 캡처에 CloudTrail 추적을 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-3 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 이 AWS 제품이나 서비스 또는 기타 AWS 서비스 서비스로 작업하는 경우도 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

데이터 암호화

보안 서비스의 주요 특징은 정보가 활발히 사용되지 않을 때 암호화된다는 것입니다.

유휴 데이터 암호화

AWS Tools for PowerShell는 사용자를 대신하여 AWS 서비스와 상호 작용하는 데 필요한 자격 증명 이외의 고객 데이터를 저장하지 않습니다.

AWS Tools for PowerShell를 사용하여 저장을 위해 로컬 컴퓨터로 고객 데이터를 전송하는 AWS 서비스를 호출하는 경우 해당 데이터가 저장, 보호 및 암호화되는 방법에 대한 자세한 내용은 해당 서비스 사용 설명서의 보안 및 규정 준수 장을 참조하세요.

전송 중 데이터 암호화

기본적으로 AWS Tools for PowerShell 및 AWS 서비스 끝점을 실행하는 클라이언트 컴퓨터에서 전송되는 모든 데이터는 HTTPS/TLS 연결을 통해 모든 데이터를 전송하여 암호화됩니다.

HTTPS/TLS 사용을 활성화하기 위해 어떤 조치도 필요하지 않습니다. 항상 활성화되어 있습니다.

자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- IAM AWS 서비스 작업 방법
- AWS 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은에서 수행하는 작업에 따라 다릅니다 AWS.

서비스 사용자 - AWS 서비스 를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. 에서 기능에 액세스할 수 없는 경우 사용 중인의 AWS 자격 증명 및 액세스 문제 해결 또는 사용 설명서를 AWS참조 AWS 서비스 하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 경우에 대한 전체 액세스 권한을 가지고 있을 것입니다 AWS. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 IAM을 사용하는 방법에 대해 자세히 알아보려면 사용 중인의 AWS 서비스 사용 설명서를 AWS참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 자격 AWS 증명 기반 정책 예제를 보려면 사용 중인의 사용 설명서를 참조 AWS 서비스 하세요.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증

명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

IAM AWS 서비스 작업 방법

가 대부분의 IAM 기능을 AWS 서비스 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

IAM AWS 서비스 에서 특정를 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 섹션을 참조하세요.

AWS 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단 AWS 하고 수정할 수 있습니다.

주제

- 에서 작업을 수행할 권한이 없음 AWS
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없음 AWS

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 awes:*GetWidget* 권한이 없을 때 발생합니다.

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

이 경우, awes:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

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

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 에서 이러한 기능을 AWS 지원하는지 여부를 알아보려면 섹션을 참조하세요IAM AWS 서비스 작업 방법.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 규정 준수 확인

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택하십시오. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. AWS 서비스 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 AWS 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 서비스를 참조하세요.

PowerShell용 도구에서 최소 TLS 버전 적용

AWS 서비스와 통신할 때 보안을 강화하려면 적절한 TLS 버전을 사용하도록 PowerShell용 도구를 구성해야 합니다. 이 방법에 대한 자세한 내용은 AWS SDK for .NET 개발자 가이드에서 최소 TLS 버전 적용을 참조하세요.

Tools for PowerShell에 대한 추가 보안 고려 사항

이 주제에는 이전 섹션에서 다룬 보안 주제 외에도 보안 고려 사항이 포함되어 있습니다.

민감한 정보의 로깅

이 도구의 일부 작업은 환경 변수의 정보를 포함하여 민감한 것으로 간주될 수 있는 정보를 반환할 수 있습니다. 이러한 정보의 노출은 특정 시나리오에서 보안 위험을 나타낼 수 있습니다. 예를 들어, 이러한 정보는 지속적 통합 및 지속적 배포(CI/CD) 로그에 포함될 수 있습니다. 따라서 이러한 출력을 언제 로그의 일부로 포함할지 검토하고 필요하지 않은 경우 출력을 억제하는 것이 중요합니다. 민감한 데이터 보호에 대한 자세한 내용은 이 AWS 제품 또는 서비스의 데이터 보호를 참조하세요.

다음 모범 사례를 고려하세요.

- 환경 변수를 사용하여 서버리스 리소스에 대한 민감한 값을 저장하지 마세요. 대신 서버리스 코드가 보안 암호 스토어(예:)에서 프로그래밍 방식으로 보안 암호를 검색하도록 합니다 AWS Secrets Manager.
- 빌드 로그의 내용을 검토하여 민감한 정보가 포함되어 있지 않은지 확인합니다. /dev/null로 파이핑하거나 출력을 bash 또는 PowerShell 변수로 캡처하여 명령 출력을 억제하는 등의 접근 방식을 고려합니다.
- 로그의 액세스 권한을 고려하여 사용 사례에 맞게 액세스 범위를 적절히 설정하세요.

Tools for PowerShell cmdlet 참조

Tools for PowerShell은 AWS 서비스에 액세스하는 데 사용할 수 있는 cmdlet을 제공합니다. 사용할 수 있는 cmdlet을 확인하려면 AWS Tools for PowerShell Cmdlet Reference를 참조하세요.

문서 기록

다음 주제에서는 AWS Tools for PowerShell 설명서에서 변경된 중요 사항에 대해 설명합니다.

또한 고객 피드백에 따라 문서를 정기적으로 업데이트합니다. 주제에 대한 의견을 보내시려면 각 페이지 하단의 "페이지 내용이 도움이 되었습니까?" 옆에 있는 피드백 버튼을 사용하세요.

AWS Tools for PowerShell 관련 변경 내용 및 업데이트에 대한 자세한 내용은 릴리스 정보를 참조하세요. 이 설명서에 대한 업데이트 알림을 받으려면 RSS feed를 구독하세요.

변경 사항	설명	날짜
새로운 기능	AWS Tools for PowerShell의 V4에 대한 지원 종료가 발표되었습니다.	2025년 9월 17일
새로운 기능	AWS Tools for PowerShell의 버전 5(V5)를 정식 버전으로 사용할 수 있습니다. 자세한 내용은 AWS Tools for PowerShell 사용 설명서(V5), 특히 V5로 마이그레이션 주제를 참조하세요.	2025년 6월 23일
새로운 기능	AWS Tools for PowerShell의 V5에 대한 미리 보기 콘텐츠를 릴리스했습니다.	2025년 6월 13일
새로운 기능	AWS Tools for PowerShell의 버전 5(미리 보기)에 대한 사용 설명서를 게시했습니다.	2025년 5월 28일
관찰성	관찰성을 위한 GA 릴리스 발표	2025년 2월 10일
새로운 기능	무결성 보호를 위한 새로운 기본 동작에 대한 정보가 추가되었습니다.	2025년 1월 15일

새로운 기능	AWS Tools for PowerShell 버전 5의 첫 번째 미리 보기 릴리스에 대한 정보가 추가되었습니다.	2024년 11월 18일
파이프라이닝, 출력 및 반복	더 이상 사용되지 않는 $AWSHistory 관련 내용을 대체했습니다.	2024년 10월 10일
관찰성	원격 측정 데이터를 수집할 수 있도록 AWS Tools for PowerShell에 관찰성에 대한 미리 보기 정보가 추가되었습니다.	2024년 9월 13일
Windows에 AWS Tools for PowerShell 설치	콘텐츠를 추출하기 전에 ZIP 파일 차단 해제에 대한 정보가 추가되었습니다.	2024년 8월 5일
EC2-Classic에 대한 정보	사용 중지된 EC2-Classic에 대한 정보를 제거했습니다.	2024년 8월 1일
코드 예시	cmdlet 예제가 있는 장을 포함했습니다.	2024년 4월 17일
추가 보안 고려 사항	민감한 데이터의 잠재적 로깅에 대한 정보가 포함되어 있습니다.	2024년 4월 16일
를 사용하여 도구 인증 구성 AWS	AWS Tools for PowerShell의 SSO 지원에 대한 정보를 추가했습니다.	2024년 3월 15일
Tools for PowerShell cmdlet 참조	Tools for PowerShell cmdlet 참조에 대한 링크가 포함된 섹션이 추가되었습니다.	2023년 11월 17일

IAM 모범 사례 업데이트 추가 포함	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오.	2023년 10월 12일
Windows에 설치	더 이상 사용되지 않는 MSI를 사용하여 Windows PowerShell용 도구를 설치하는 방법에 대한 정보를 제거했습니다.	2023년 9월 25일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오.	2023년 9월 8일
파이프라이닝 및 $AWSHistory	`Set-AWSHistoryConfiguration` cmdlet에 `IncludeSensitiveData` 파라미터를 추가했습니다.	2023년 3월 9일
cmdlet에서 ClientConfig 파라미터 사용	ClientConfig 파라미터 지원에 대한 정보가 추가되었습니다.	2022년 10월 28일
Windows PowerShell을 사용하여 Amazon EC2 인스턴스 시작	EC2-Classic에 대한 참고 사항이 추가되었습니다.	2022년 7월 26일
AWS Tools for PowerShell 버전 4	Windows 및 Linux/macOS에 대한 설치 지침을 포함한 버전 4에 대한 정보와 버전 3과의 차이점을 설명하고 새로운 기능을 소개하는 마이그레이션 주제가 추가되었습니다.	2019년 11월 21일

AWS Tools for PowerShell 3.3.563	`AWS.Tools.Common` 모듈의 평가판 버전을 설치하고 사용하는 방법에 대한 정보가 추가되었습니다. 이 새로운 모듈은 이전 모놀리식 패키지를 AWS 서비스당 하나의 공유 모듈과 하나의 모듈로 나눕니다.	2019년 10월 18일
AWS Tools for PowerShell 3.3.343.0	AWS Tools for PowerShell 사용 단원에 AWS Lambda 기능을 구축하려는 PowerShell Core 개발자를 위한 AWS Lambda Tools for PowerShell을 소개하는 정보를 추가했습니다.	2018년 9월 11일
AWS Tools for Windows PowerShell 3.1.31.0	시작하기 단원에 Security Assertion Markup Language(SAML)를 사용하여 사용자에 대한 연동 자격 증명 구성을 지원하는 새로운 cmdlet에 대한 정보를 추가했습니다.	2015년 12월 1일
AWS Tools for Windows PowerShell 2.3.19	Cmdlets 검색 및 별칭 단원에 사용자가 원하는 AWS cmdlet을 보다 쉽게 찾을 수 있도록 도와주는 새 `Get-AWSCmdletName` cmdlet에 대한 정보를 추가했습니다.	2015년 2월 5일

AWS Tools for Windows PowerShell 1.1.1.0	cmdlet에서 출력된 모음은 항상 PowerShell 파이프라인에 열거됩니다. 페이징 가능한 서비스 호출에 대한 자동 지원 새 $AWSHistory 셸 변수가 서비스 응답 및 서비스 요청(선택 사항)을 수집합니다. AWSRegion 인스턴스에서 SystemName 대신 Region 필드를 사용하여 파이프라이닝을 지원합니다. Remove-S3Bucket에서 -DeleteObjects 스위치 옵션을 지원합니다. Set-AWSCredentials에서 사용성 문제가 해결되었습니다. Initialize-AWSDefaults는 자격 증명 및 리전 데이터를 얻은 위치에서 보고합니다. Stop-EC2Instance는 입력으로 Amazon.EC2.Model.Reservation 인스턴스를 허용합니다. 일반 목록<T> 파라미터 유형이 어레이 유형(T[])으로 대체됩니다. 리소스를 삭제하거나 종료하는 cmdlet에서 삭제 전에 확인 메시지를 표시합니다. Write-S3Object는 Amazon S3에 업로드할 인라인 텍스트 콘텐츠를 지원합니다.	2013년 5월 15일

AWS Tools for Windows PowerShell 1.0.1.0	Tools for Windows PowerShell 모듈의 설치 위치가 변경되어 Windows PowerShell 버전 3을 사용하는 환경에서 자동 로드의 이점을 활용할 수 있습니다. 이제 모듈 및 지원 파일이 `AWS ToolsPowerShell` 아래의 `AWSPowerShell` 하위 폴더에 설치됩니다. `AWS ToolsPowerShell` 폴더에 있는 이전 버전의 파일이 설치 관리자에 의해 자동으로 제거됩니다. Windows PowerShell(모든 버전)의 `PSModulePath` 가 이 릴리스에서는 모듈의 상위 폴더(`AWS ToolsPowerShell`)를 포함하도록 업데이트되었습니다. Windows PowerShell 버전 2가 설치된 시스템의 경우, 시작 메뉴 바로 가기가 새 위치에서 모듈을 가져오고 나서 `Initialize-AWSDefaults` 를 실행하도록 업데이트되었습니다. Windows PowerShell 버전 3이 설치된 시스템의 경우, 시작 메뉴 바로 가기가 `Initialize-AWSDefaults` 만 남겨두고 `Import-Module` 명령을 제거하도록 업데이트되었습니다. `Import-Module` 파일의 `AWSPowerShell.psd1` 을 수행하도록 PowerShell 프로파일을 편집한 경우 파일의 새 위치를 가리키도록 업데이트해야 합니다(또는, PowerShell 버	2012년 12월 21일

	전 3을 사용 중인 경우 더 이상 필요하지 않으므로 `Import-Module` 문을 제거해야 함). 이러한 변경의 결과로서 Tools for Windows PowerShell 모듈이 이제는 `Get-Module -ListAvailable` 실행 시 사용 가능한 모듈로 나열됩니다. 또한, Windows PowerShell 버전 3 사용자의 경우, 모듈에서 내보낸 cmdlet을 실행하면 `Import-Module` 를 먼저 사용하지 않아도 현재의 PowerShell 셸에 모듈이 자동으로 로드됩니다. 따라서 실행 정책에 스크립트 실행이 허용되지 않는 시스템에서 cmdlet을 대화형으로 사용할 수 있습니다.	
AWS Tools for Windows PowerShell 1.0.0.0	초기 릴리스	2012년 12월 6일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.