

사용자 가이드

# AWS Tools for PowerShell



# AWS Tools for PowerShell: 사용자 가이드

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

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

# Table of Contents

AWS Tools for PowerShell이란 무엇입니까? .....	1
SDK 메이저 버전에 대한 유지 관리 및 지원 .....	2
AWS.Tools .....	2
AWSPowerShell.NetCore .....	3
AWSPowerShell .....	3
이 설명서의 사용법 .....	3
설치 .....	5
Windows에 설치 .....	5
필수 조건 .....	6
AWS.Tools 설치 .....	6
설치 AWSPowerShell. NetCore .....	8
설치 AWSPowerShell .....	9
스크립트 실행 활성화 .....	10
버전 관리 .....	12
업데이트 AWS Tools for PowerShell .....	13
Linux 또는 macOS에 설치 .....	15
설정 개요 .....	15
필수 조건 .....	6
AWS.Tools 설치 .....	16
설치 AWSPowerShell. NetCore .....	19
스크립트 실행 .....	10
PowerShell 콘솔 구성 .....	21
세션 초기화 PowerShell .....	21
버전 관리 .....	12
리눅스 또는 AWS Tools for PowerShell macOS에서 업데이트 .....	22
관련 정보 .....	23
AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션 .....	24
새롭게 완전 모듈화된 AWS.Tools 버전 .....	24
새로운 Get-AWSService cmdlet .....	25
Cmdlet에서 반환된 객체를 제어하는 새로운 -Select 파라미터 .....	25
출력 항목 수에 대한 보다 일관된 제한 .....	27
더욱 사용하기 쉬워진 Stream 파라미터 .....	27
속성 이름별로 파이프 확장 .....	28
정적 공통 파라미터 .....	28

AWS.Tools 필수 파라미터 선언 및 강제 시행 .....	28
모든 파라미터에 Null 값 사용 가능 .....	29
더 이상 사용되지 않는 기능 제거 .....	29
시작하기 .....	30
도구 인증 구성 .....	30
IAM Identity Center 사용 및 구성 .....	31
IAM ID 센터를 PowerShell 사용하기 위한 도구를 구성하십시오. ....	31
AWS 액세스 포털 세션 시작 .....	33
예 .....	34
추가 정보 .....	34
사용: AWS CLI .....	35
AWS 지역 지정하기 .....	38
사용자 지정 또는 비표준 엔드포인트 지정 .....	40
추가 정보 .....	40
연동 자격 증명 구성 .....	40
필수 조건 .....	41
자격 증명 연동 사용자가 AWS 서비스 API에 대한 연동된 액세스 권한을 얻는 방법 .....	41
AWS Tools for PowerShell에서 SAML 지원이 작동하는 방식 .....	43
PowerShell SAML 구성 Cmdlet 사용 방법 .....	44
추가 읽기 자료 .....	48
Cmdlet 검색 및 별칭 .....	49
Cmdlet 검색 .....	49
Cmdlet 이름 지정 및 별칭 .....	55
파이프라이닝 및 \$AWSHistory .....	59
\$AWSHistory .....	60
보안 인증 정보 및 프로파일 확인 .....	63
자격 증명 검색 순서 .....	63
사용자 및 역할 .....	64
사용자 및 권한 집합 .....	65
서비스 역할 .....	65
레거시 보안 인증 사용 .....	66
중요 경고 및 지침 .....	66
AWS 자격 증명 .....	67
공유 자격 증명 .....	76
AWS 서비스 작업 .....	82
PowerShell 파일 연결 인코딩 .....	82

PowerShell 도구에 대해 반환된 객체 .....	83
Amazon EC2 .....	83
Amazon S3 .....	83
AWS Lambda 및 AWS Tools for PowerShell .....	84
Amazon SNS 및 Amazon SQS .....	84
CloudWatch .....	84
참고 항목 .....	84
주제 .....	84
Amazon S3 및 Tools for Windows PowerShell .....	85
Amazon S3 버킷 생성, 리전 확인 및 제거(선택 사항) .....	85
Amazon S3 버킷을 웹 사이트로 구성하고 로깅 활성화 .....	86
Amazon S3 버킷에 객체 업로드 .....	87
Amazon S3 객체 및 버킷 삭제 .....	89
Amazon S3에 인라인 텍스트 콘텐츠 업로드 .....	90
Amazon EC2 및 Tools for Windows PowerShell .....	91
키 페어 생성 .....	91
보안 그룹 생성 .....	94
AMI 찾기 .....	98
인스턴스 시작 .....	101
AWS Lambda 및 AWS Tools for PowerShell .....	106
필수 조건 .....	6
AWSLambdaPSCore 모듈 설치 .....	107
참고 항목 .....	84
Amazon SQS, Amazon SNS 및 Tools for Windows PowerShell .....	107
Amazon SQS 대기열 생성 및 대기열 ARN 가져오기 .....	108
Amazon SNS 주제 생성 .....	108
SNS 주제에 권한 부여 .....	108
SNS 주제에 대한 대기열을 구독합니다. ....	109
권한 부여 .....	109
결과 확인 .....	110
AWS Tools for Windows PowerShell에서의 CloudWatch .....	111
CloudWatch 대시보드에 사용자 지정 지표 게시 .....	111
참고 항목 .....	84
ClientConfig 사용 .....	112
ClientConfig 파라미터 사용 .....	112
정의되지 않은 속성 사용 .....	113

AWS 리전 지정 .....	113
코드 예시 .....	114
작업 및 시나리오 .....	114
ACM .....	116
AppStream 2.0 .....	121
Aurora .....	147
Auto Scaling .....	148
AWS Budgets .....	183
AWS Cloud9 .....	185
AWS CloudFormation .....	191
CloudFront .....	204
CloudTrail .....	212
CloudWatch .....	217
CodeCommit .....	221
CodeDeploy .....	226
CodePipeline .....	245
Amazon Cognito 자격 증명 .....	263
AWS Config .....	267
Device Farm .....	286
AWS Directory Service .....	287
AWS DMS .....	311
DynamoDB .....	312
Amazon EC2 .....	327
Amazon ECR .....	454
Amazon ECS .....	455
Amazon EFS .....	461
Amazon EKS .....	468
Elastic Load Balancing - 버전 1 .....	480
Elastic Load Balancing - 버전 2 .....	500
Amazon FSx .....	523
AWS Glue .....	531
AWS Health .....	533
IAM .....	534
Kinesis .....	606
Lambda .....	610
Amazon ML .....	623

Macie .....	628
AWS OpsWorks .....	630
AWS 가격표 .....	631
Resource Groups .....	634
Resource Groups Tagging API .....	641
Route 53 .....	646
Amazon S3 .....	661
S3 Glacier .....	695
Amazon SES .....	699
Amazon SNS .....	700
Amazon SQS .....	702
AWS STS .....	713
AWS Support .....	718
Systems Manager .....	724
Amazon Translate .....	796
AWS WAFV2 .....	796
WorkSpaces .....	798
보안 .....	814
데이터 보호 .....	814
데이터 암호화 .....	815
ID 및 액세스 관리 .....	816
고객 .....	816
보안 인증을 통한 인증 .....	817
정책을 사용한 액세스 관리 .....	820
AWS 서비스에서 IAM을 사용하는 방식 .....	822
AWS 보안 인증 및 액세스 문제 해결 .....	822
규정 준수 검증 .....	824
최소 TLS 버전 적용 .....	825
추가 보안 고려 사항 .....	825
민감한 정보의 로깅 .....	826
cmdlet 참조 .....	827
문서 기록 .....	828
.....	dcccxxxiv

# AWS Tools for PowerShell이란 무엇입니까?

AWS Tools for PowerShell 는 에서 제공하는 기능을 기반으로 구축된 PowerShell 모듈 세트입니다 AWS SDK for .NET. 이를 AWS Tools for PowerShell 통해 PowerShell 명령줄에서 AWS 리소스에 대한 작업을 스크립팅할 수 있습니다.

cmdlet은 다양한 AWS 서비스 HTTP 쿼리 API를 사용하여 구현되더라도 매개 변수를 지정하고 결과를 처리할 수 있는 관용적 PowerShell 환경을 제공합니다. 예를 들어, cmdlet용 cmdlet은 PowerShell 파이프라이닝을 AWS Tools for PowerShell 지원합니다. 즉, cmdlet에서 개체를 파이프하고 들어오고 나갈 수 있습니다. PowerShell

IAM (인프라 지원) 을 비롯하여 자격 증명을 처리하는 AWS Tools for PowerShell 방법이 유연합니다. AWS Identity and Access Management IAM 사용자 자격 증명, 임시 보안 토큰 및 IAM 역할과 함께 도구를 사용할 수 있습니다.

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

## Note

AWS Tools for PowerShell 버전 4는 최신 메이저 릴리스이며 버전 3.3에 대한 이전 버전과 호환되는 업데이트입니다. AWS Tools for PowerShell 기존 cmdlet 동작을 유지하면서 기능이 상당히 향상되었습니다. 새 버전으로 업그레이드한 후에도 기존 스크립트가 계속 작동하지만 업그레이드하기 전에 철저히 테스트하는 것이 좋습니다. 버전 4의 변경 사항에 대한 자세한 내용은 [AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션](#) 단원을 참조하십시오.

AWS Tools for PowerShell 는 다음과 같은 세 가지 패키지로 제공됩니다.

- [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 안내서](#)에서 다음 내용을 참조하세요.

- [AWS SDK 및 도구 유지 관리 정책](#)
- [AWS SDK 및 도구 버전 지원 매트릭스](#)

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

PowerShell Gallery **AWS.Tools**

ZIP Archive **AWS.Tools**

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

이것은 최신 버전이며 Windows, Linux AWS Tools for PowerShell 및 macOS를 포함하여 지원되는 모든 운영 체제에서 실행됩니다. 이 패키지는 설치 모듈 1개(AWS.Tools.Installer, 공용 모듈 1개(AWS.Tools.Common, AWS 서비스별 모듈 1개 (예: AWS.Tools.EC2, AWS.Tools.IdentityManagement, AWS.Tools.S3, 등) 를 제공합니다.

이 AWS.Tools.Installer 모듈은 각 서비스의 모듈을 설치, 업데이트 및 제거할 수 있는 cmdlet을 제공합니다. AWS 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 실행하기만 하면 됩니다. PowerShell cmdlet을 실행하려는 AWS 서비스의 AWS.Tools.Common 모듈과 모듈을 자동으로 가져옵니다. AWS.Tools.Installer 모듈을 사용하여 서비스 모듈을 설치하는 경우 이 모듈은 자동으로 설치됩니다.

다음 버전을 실행 중인 AWS Tools for PowerShell 컴퓨터에 설치할 수 있습니다.

- PowerShell 윈도우, 리눅스 또는 macOS의 코어 6.0 이상
- .NET 프레임워크 4.7.2 이상이 설치된 윈도우에서 윈도우 PowerShell 5.1 이상

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

# AWSPowerShell. NetCore - 단일 모듈 버전의 AWS Tools for PowerShell

PowerShell Gallery **AWSPowerShell.NetCore**

ZIP Archive **AWSPowerShell.NetCore**

이 버전은 모든 AWS 서비스에 대한 지원이 포함된 하나의 대형 모듈로 구성되어 있습니다. 이 모듈을 사용하려면 먼저 수동으로 가져와야 합니다.

다음 버전을 실행 중인 AWS Tools for PowerShell 컴퓨터에 설치할 수 있습니다.

- PowerShell 윈도우, 리눅스 또는 macOS의 코어 6.0 이상
- .NET 프레임워크 4.7.2 이상이 설치된 윈도우에서 윈도우 PowerShell 3.0 이상

이 안내서에서는 이 버전만 지정해야 하는 경우 모듈 이름으로 참조합니다. AWSPowerShell NetCore.

## AWSPowerShell - 윈도우용 단일 모듈 버전 PowerShell

PowerShell Gallery **AWSPowerShell**

ZIP Archive **AWSPowerShell**

이 버전의 AWS Tools for PowerShell 는 Windows 버전 2.0~5.1을 실행하는 Windows PowerShell 컴퓨터에만 호환되며 설치할 수 있습니다. PowerShell 코어 6.0 이상 또는 다른 운영 체제 (리눅스 또는 macOS) 와는 호환되지 않습니다. 이 버전은 모든 AWS 서비스를 지원하는 하나의 대형 모듈로 구성되어 있습니다.

이 안내서에서는 이 버전만 지정해야 하는 경우 모듈 이름으로 참조합니다 AWSPowerShell.

## 이 설명서의 사용법

이 안내서는 다음과 같은 주요 단원으로 구성되어 있습니다.

### [AWS Tools for PowerShell 설치](#)

이 섹션에서는 설치 방법에 대해 설명합니다 AWS Tools for PowerShell. 여기에는 아직 계정이 없는 AWS 경우 가입하는 방법과 cmdlet을 실행하는 데 사용할 수 있는 IAM 사용자를 만드는 방법이 포함되어 있습니다.

## [AWS Tools for Windows PowerShell 시작](#)

이 섹션에서는 자격 증명 및 AWS 지역 지정 AWS Tools for PowerShell, 특정 서비스에 대한 cmdlet 찾기, cmdlet의 별칭 사용 등 를 사용하는 기본 사항에 대해 설명합니다.

## [AWS Tools for PowerShell에서 AWS 서비스 작업](#)

이 섹션에는 를 사용하여 가장 일반적인 몇 가지 작업을 수행하는 AWS Tools for PowerShell 방법에 대한 정보가 포함되어 있습니다. AWS

# AWS Tools for PowerShell 설치

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

주제

- [AWS Tools for PowerShell 윈도우에 설치하기](#)
- [리눅스 또는 AWS Tools for PowerShell macOS에 설치](#)
- [AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션](#)

## AWS Tools for PowerShell 윈도우에 설치하기

Windows 기반 컴퓨터는 다음 패키지 옵션 중 하나를 실행할 수 있습니다. AWS Tools for PowerShell

- [AWS.Tools](#)- 의 모듈화된 버전. AWS Tools for PowerShell각 AWS 서비스는 공유 지원 모듈과 함께 별도의 소형 모듈로 지원됩니다. `AWS.Tools.Common` `AWS.Tools.Installer`
- [AWSPowerShell.NetCore](#)- 의 단일 대형 모듈 버전. AWS Tools for PowerShell이 대형 단일 모듈에서 모든 AWS 서비스를 지원합니다.

### 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. 이 대형 단일 모듈에서 모든 AWS 서비스를 지원합니다.

실행 중인 Windows 릴리스 및 에디션에 따라 선택하는 패키지가 다릅니다.

### Note

윈도우용 도구 PowerShell (AWSPowerShell 모듈) 는 모든 윈도우 기반 아마존 머신 이미지 (AMI) 에 기본적으로 설치됩니다.

설정에는 이 AWS Tools for PowerShell 주제에 자세히 설명된 다음과 같은 상위 수준 작업이 포함됩니다.

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

## 필수 조건

PowerShell Core를 PowerShell 포함한 최신 버전은 Microsoft 웹 사이트의 [다양한 버전 설치에서](#) Microsoft에서 다운로드할 수 있습니다. PowerShell

## Windows에 **AWS.Tools** 설치

Windows PowerShell 5.1 또는 PowerShell Core 6.0 이상이 설치된 Windows를 실행하는 AWS Tools for PowerShell 컴퓨터에 모듈화된 버전을 설치할 수 있습니다. PowerShellCore를 설치하는 방법에 대한 자세한 내용은 Microsoft [웹 사이트의 다양한 버전 설치를](#) 참조하십시오. PowerShell

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 단순화합니다. AWS.Tools.Installer업데이트된 버전을 PowerShellGet 요구하고 자동으로 다운로드하고 설치합니다. AWS.Tools.Installer모듈 버전을 자동으로 동기화합니다. 한 모듈의 새 버전을 설치하거나 업데이트하면 cmdlet이 다른 모든 AWS.Tools 모듈을 동일한 버전으로 AWS.Tools.Installer 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.
- Install-Modulecmdlet을 사용하여 PowerShell 갤러리에서 각 서비스 모듈을 설치합니다.

모듈을 사용하여 **AWS.Tools** Windows에 설치하려면 **AWS.Tools.Installer**

1. PowerShell 세션을 시작합니다.

**Note**

당면한 작업에 필요한 경우를 제외하고는 높은 권한을 가진 관리자 PowerShell 권한으로 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. 모듈을 설치할 수 있게 **y** PowerShell 하려면 Enter를 누르십시오. 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 `Install-AWSToolsModule` cmdlet을 사용하여 사용하려는 각 AWS 서비스에 대한 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. 이 모듈은 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 PSRepository인 PSGallery(<https://www.powershellgallery.com/>)에서 요청된 모든 모듈을 다운로드하고 신뢰할 수 있는 소스로 간주합니다. Get-PSRepository -Name PSGallery 명령을 사용하여 이 PSRepository에 대해 자세히 알아볼 수 있습니다.

이전 명령을 실행하면 기본적으로 %USERPROFILE%\Documents\WindowsPowerShell\Modules 폴더에 모듈이 설치됩니다. 컴퓨터의 모든 AWS Tools for PowerShell 사용자를 설치하려면 관리자로 시작한 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 갤러리에](#) 있는 것처럼 적절한 모듈 이름을 사용하여 유사한 명령을 실행하십시오.

## 설치 AWSPowerShell. NetCore 윈도우에서

를 설치할 수 있습니다 AWSPowerShell. NetCore Windows PowerShell 버전 3~5.1 또는 PowerShell 코어 6.0 이상의 버전을 실행하는 컴퓨터에 설치하세요. PowerShell Core를 설치하는 방법에 대한 자세한 내용은 Microsoft PowerShell 웹 사이트의 [다양한 버전 설치를](#) 참조하십시오. PowerShell

설치할 수 있습니다 AWSPowerShell. NetCore 두 가지 방법 중 하나로

- 에서 모듈 다운로드 [AWSPowerShell.NetCore.zip](#)을 선택하고 모듈 디렉토리 중 하나에서 압축을 풉니다. PSModulePath 환경 변수의 값을 표시하여 모듈 디렉토리를 찾을 수 있습니다.
- 다음 절차에 설명된 대로 Install-Module cmdlet을 사용하여 PowerShell 갤러리에서 설치합니다.

설치하려면 AWSPowerShell NetCore 설치 모듈 PowerShell cmdlet을 사용하여 갤러리에서

설치하려면. AWSPowerShell NetCore PowerShell 갤러리에서 컴퓨터는 PowerShell 5.0 이상을 실행하거나 PowerShell 3 이상을 [PowerShellGet](#) 실행해야 합니다. 다음 명령을 실행합니다.

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

관리자 PowerShell 권한으로 실행하는 경우 이전 명령은 컴퓨터의 모든 사용자를 AWS Tools for PowerShell 위해 설치됩니다. 관리자 권한 없이 표준 PowerShell 사용자로 실행하는 경우 현재 AWS Tools for PowerShell 사용자에게만 동일한 명령이 설치됩니다.

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
```

PowerShell 3.0 이상 릴리스에서는 일반적으로 모듈에서 cmdlet을 처음 실행할 때 모듈을 PowerShell 세션에 로드합니다. AWSPowerShell NetCore 모듈이 너무 커서 이 기능을 지원할 수 없습니다. 대신 를 명시적으로 로드해야 합니다. AWSPowerShell NetCore 다음 명령을 실행하여 PowerShell 세션에 코어 모듈을 입력합니다.

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

를 로드하려면 AWSPowerShell.NetCore 모듈을 PowerShell 세션에 자동으로 추가하세요. PowerShell 프로필에 해당 명령을 추가하세요. PowerShell 프로필 편집에 대한 자세한 내용은 PowerShell 설명서의 [프로필](#) 정보를 참조하십시오.

## AWSPowerShell Windows에 설치 PowerShell

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

- [AWSPowerShell.zip에서](#) 모듈을 다운로드하고 모듈 디렉토리 중 하나에서 압축을 풉니다. PSModulePath 환경 변수의 값을 표시하여 모듈 디렉토리를 찾을 수 있습니다.

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

설치 모듈 AWSPowerShell cmdlet을 사용하여 PowerShell 갤러리에서 설치하려면

PowerShell 5.0 이상을 실행 중이거나 3 이상 버전에 설치한 경우 PowerShell 갤러리에서 을 설치할 [PowerShellGet](#) 수 있습니다. AWSPowerShell PowerShell 다음 명령을 실행하여 Microsoft [PowerShell 갤러리에서](#) 설치하고 업데이트할 AWSPowerShell 수 있습니다.

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

AWSPowerShell 모듈을 PowerShell 세션에 자동으로 로드하려면 이전 import-module cmdlet을 프로필에 추가하십시오. PowerShell [프로필 편집에 대한 자세한 내용은 설명서의 PowerShell 프로필 정보를 참조하십시오.](#) PowerShell

#### Note

윈도우용 도구는 모든 윈도우 기반 아마존 머신 이미지 (AMI) 에 기본적으로 PowerShell 설치됩니다.

## 스크립트 실행 활성화

AWS Tools for PowerShell 모듈을 로드하려면 스크립트 실행을 활성화해야 합니다. PowerShell 스크립트 실행을 활성화하려면 Set-ExecutionPolicy cmdlet을 실행하여 RemoteSigned 정책을 설정합니다. 자세한 내용은 Microsoft Technet 웹 사이트의 [실행 정책 소개](#)를 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 관리자로 PowerShell 세션을 여십시오. 시작을 선택한 다음 All Programs(모든 프로그램)를 선택합니다. [보조프로그램] 을 선택한 다음 [Windows] 를 선택합니다 PowerShell.

PowerShellWindows를 마우스 오른쪽 단추로 클릭하고 컨텍스트 메뉴에서 관리자 권한으로 실행을 선택합니다.

2. 명령 프롬프트에서 다음을 입력합니다.

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

### Note

64비트 시스템에서는 32비트 버전의 PowerShell Windows PowerShell (x86)에 대해 이 작업을 별도로 수행해야 합니다.

실행 정책을 올바르게 설정하지 않은 경우 프로필과 같은 스크립트를 실행하려고 할 때마다 다음 오류가 PowerShell 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
```

Windows용 도구 PowerShell 설치 프로그램은 AWSPowerShell 모듈이 포함된 디렉토리의 위치를 ModulePath 포함하도록 [PS](#)를 자동으로 업데이트합니다.

에는 AWS 모듈의 디렉터리 위치가 PSModulePath 포함되므로 Get-Module -ListAvailable cmdlet에는 모듈이 표시됩니다.

```
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

AWS 새 AWS 서비스 및 기능을 지원하기 위해 AWS Tools for PowerShell 정기적으로 새 버전을 릴리스합니다. 설치한 도구의 버전을 확인하려면 [Get-AWSPowerShellVersion](#) cmdlet을 실행하십시오.

```
PS > Get-AWSPowerShellVersion
```

```
Tools for PowerShell
Version 4.1.11.0
Copyright 2012-2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Amazon Web Services SDK for .NET
Core Runtime Version 3.7.0.12
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md
```

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

또한 [Get-AWSPowerShellVersion](#) 명령에 `-ListServiceVersionInfo` 매개 변수를 추가하여 현재 버전의 도구에서 지원되는 AWS 서비스 목록을 볼 수 있습니다. 모듈화된 `AWS.Tools.*` 옵션을 사용하는 경우 현재 가져온 모듈만 표시됩니다.

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

```
...
```

Service	Noun Prefix	Module Name	SDK
Assembly			
Version			
-----	-----	-----	
Alexa For Business 3.7.0.11	ALXB	AWS.Tools.AlexaForBusiness	
Amplify Backend 3.7.0.11	AMPB	AWS.Tools.AmplifyBackend	

Amazon API Gateway 3.7.0.11	AG	AWS.Tools.APIGateway
Amazon API Gateway Management API 3.7.0.11	AGM	AWS.Tools.ApiGatewayManagementApi
Amazon API Gateway V2 3.7.0.11	AG2	AWS.Tools.ApiGatewayV2
Amazon Appflow 3.7.1.4	AF	AWS.Tools.Appflow
Amazon Route 53 3.7.0.12	R53	AWS.Tools.Route53
Amazon Route 53 Domains 3.7.0.11	R53D	AWS.Tools.Route53Domains
Amazon Route 53 Resolver 3.7.1.5	R53R	AWS.Tools.Route53Resolver
Amazon Simple Storage Service (S3) 3.7.0.13	S3	AWS.Tools.S3
...		

실행 PowerShell 중인 버전을 확인하려면 \$PSVersionTable [자동 변수의](#) 내용을 \$PSVersionTable 보려면 를 입력하여 확인하십시오.

```
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
```

## AWS Tools for PowerShell 윈도우에서 업데이트

업데이트된 버전이 AWS Tools for PowerShell 출시되면 로컬에서 실행 중인 버전을 정기적으로 업데이트해야 합니다.

## AWS.Tools 모듈화된 모듈을 업데이트하십시오.

AWS.Tools 모듈을 최신 버전으로 업데이트하려면 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.

## PowerShell Core용 도구 업데이트

Get-AWSPowerShellVersioncmdlet을 실행하여 실행 중인 버전을 확인한 다음 [PowerShell 갤러리](#) 웹 사이트에서 PowerShell 제공되는 Windows용 도구 버전과 비교하십시오. 2~3주마다 확인하는 것이 좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 포함된 버전으로 업데이트한 후에만 사용할 수 있습니다.

의 AWSPowerShell 최신 릴리스를 설치하기 전. NetCore기존 모듈을 제거합니다. 기존 패키지를 제거하기 전에 열려 있는 PowerShell 세션을 모두 닫으십시오. 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 명령을 Import-Module AWSPowerShell.NetCore 실행하여 업데이트된 cmdlet을 세션에 로드합니다. PowerShell

## Windows용 도구를 업데이트하세요. PowerShell

Get-AWSPowerShellVersioncmdlet을 실행하여 실행 중인 버전을 확인한 다음 [PowerShell 갤러리](#) 웹 사이트에서 PowerShell 제공되는 Windows용 도구 버전과 비교하십시오. 2~3주마다 확인하는 것이

좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 포함된 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 PowerShell](#) 웹 사이트에서 최신 버전을 다운로드하십시오. 다운로드된 파일 이름의 패키지 버전 번호를 `Get-AWSPowerShellVersion` cmdlet을 실행할 때 얻을 수 있는 버전 번호와 비교합니다.
  2. 다운로드 버전이 설치한 버전보다 많으면 Windows용 도구 PowerShell 콘솔을 모두 닫으십시오.
  3. 새 버전의 Windows용 도구를 설치하십시오. PowerShell

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

## 리눅스 또는 AWS Tools for PowerShell macOS에 설치

이 항목에서는 Linux 또는 AWS Tools for PowerShell macOS에 를 설치하는 방법에 대한 지침을 제공합니다.

### 설정 개요

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

- [AWS.Tools](#)— 의 모듈화된 버전. AWS Tools for PowerShell각 AWS 서비스는 공유 지원 모듈을 사용하는 작은 개별 모듈로 지원됩니다. `AWS.Tools.Common`
- [AWSPowerShell.NetCore](#)— 의 단일 대형 모듈 버전. AWS Tools for PowerShell이 대형 단일 모듈에서 모든 AWS 서비스를 지원합니다.

#### 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 pwsh 셸에서 PowerShell 실행하여 시작하십시오.
3. `AWS.Tools` 또는 둘 중 하나를 설치하십시오 `AWSPowerShell`. `NetCore`.
4. 적절한 `Import-Module cmdlet`을 실행하여 모듈을 세션으로 PowerShell 가져옵니다.
5. [Initialize-AWSDefaultConfiguration](#) cmdlet을 실행하여 자격 증명을 제공하십시오. AWS

## 필수 조건

를 AWS Tools for PowerShell Core 실행하려면 컴퓨터에서 PowerShell Core 6.0 이상이 실행되고 있어야 합니다.

- 지원되는 Linux 플랫폼 릴리스 목록과 Linux 기반 컴퓨터에 최신 버전을 설치하는 방법에 대한 자세한 내용은 Microsoft 웹 [PowerShell 사이트에서 Linux에 설치를 참조하십시오](#). PowerShell Arch, Kali, Raspbian 등과 같은 Linux 기반의 일부 운영 체제는 공식적으로 지원되지 않지만 다양한 수준의 커뮤니티 지원이 이루어지고 있습니다.
- 지원되는 macOS 버전 및 macOS에 최신 버전을 설치하는 방법에 대한 자세한 내용은 Microsoft 웹 사이트에서 [macOS에 PowerShell 설치하기를 참조하십시오](#). PowerShell

## Linux 또는 macOS에 **AWS.Tools** 설치

Core 6.0 이상을 실행하는 AWS Tools for PowerShell PowerShell 컴퓨터에 모듈화된 버전을 설치할 수 있습니다. PowerShell Core를 설치하는 방법에 대한 자세한 내용은 Microsoft PowerShell 웹 사이트의 [다양한 버전 설치를 참조하십시오](#). PowerShell

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 단순화합니다. `AWS.Tools.Installer` 업데이트된 버전을 `PowerShellGet` 요구하고 자동으로 다운로드하고 설치합니다. `AWS.Tools.Installer` 모듈 버전을 자동으로 동기화합니다. 한 모듈의 새 버전을 설치하거나 업데이트하면 cmdlet이 다른 모든 `AWS.Tools` 모듈을 동일한 버전으로 `AWS.Tools.Installer` 자동으로 업데이트합니다.

이 방법은 다음 절차에 설명되어 있습니다.

- [AWS.Tools.zip](#)에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `$Env:PSModulePath` 변수의 값을 인쇄하여 모듈 디렉터리를 찾을 수 있습니다.
- `Install-Modulecmdlet`을 사용하여 PowerShell 갤러리에서 각 서비스 모듈을 설치합니다.

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

1. 다음 명령을 실행하여 PowerShell Core 세션을 시작합니다.

```
$ pwsh
```

#### Note

당면한 작업에 필요한 경우를 제외하고는 상승된 권한을 가진 관리자 PowerShell 권한으로 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. 모듈을 설치할 수 있게 **y** PowerShell 하려면 Enter를 누르십시오. 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. 이 모듈은 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 PSRepository인 PSGallery(<https://www.powershellgallery.com/>)에서 요청된 모든 모듈을 다운로드하고 해당 리포지토리를 신뢰할 수 있는 소스로 간주합니다. Get-PSRepository -Name PSGallery 명령을 사용하여 이 PSRepository에 대해 자세히 알아볼 수 있습니다.

이전 명령은 모듈을 시스템의 기본 디렉터리에 설치합니다. 실제 디렉토리는 운영 체제 배포판과 버전, PowerShell 설치한 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 에 있을 가능성이 큼니다. ~/.local/share/powershell/Modules 자세한 내용은 Microsoft PowerShell 웹 [PowerShell 사이트의 Linux에 설치](#)를 참조하십시오. 모듈이 설치된 위치를 확인하려면 다음 명령을 실행합니다.

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

다른 모듈을 설치하려면 [PowerShell 갤러리에](#) 있는 것과 같이 적절한 모듈 이름을 사용하여 유사한 명령을 실행하십시오.

## 설치 AWSPowerShell. NetCore 리눅스 또는 macOS에서

의 AWSPowerShell 최신 릴리스로 업그레이드하려면 NetCore의 [리눅스 또는 AWS Tools for PowerShell macOS에서 업데이트](#) 지침을 따르십시오. 의 AWSPowerShell 이전 버전을 제거합니다. NetCore 먼저.

설치할 수 있습니다 AWSPowerShell. NetCore 다음 두 가지 방법 중 하나로:

- [AWSPowerShell.NetCore.zip](#)에서 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 갤러리에서 설치합니다.

설치하기. AWSPowerShell NetCore 리눅스 또는 macOS에서 설치 모듈 cmdlet을 사용하는 경우 다음 명령을 실행하여 PowerShell Core 세션을 시작합니다.

```
$ pwsh
```

### Note

관리자 승격된 PowerShell PowerShell 권한으로 `sudo pwsh` 실행하기 위해 실행하는 것으로 시작하지 않는 것이 좋습니다. 이는 잠재적 보안 위험 때문이며 최소 권한의 원칙과 일치하지 않습니다.

설치하려면. AWSPowerShell NetCore PowerShell 갤러리의 단일 모듈 패키지에서 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. 모듈을 설치할 수 있도록 **y** PowerShell 하려면 **l**을 입력합니다. 저장소를 신뢰하지 않은 상태에서 메시지가 표시되지 않도록 하려면 다음 명령을 실행할 수 있습니다.

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

컴퓨터의 모든 AWS Tools for PowerShell 사용자용으로 설치하려는 경우가 아니라면 이 명령을 루트로 실행할 필요가 없습니다. 이 작업을 수행하려면 시작한 PowerShell 세션에서 다음 명령을 실행하십시오 `sudo pwsh`.

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

## 스크립트 실행

이 `Set-ExecutionPolicy` 명령은 Windows 이외의 시스템에서는 사용할 수 없습니다. 실행할 `Get-ExecutionPolicy` 수 있습니다. 그러면 Windows가 아닌 시스템에서 실행되는 PowerShell Core의 기본 실행 정책 설정이 다음과 `Unrestricted` 같음을 알 수 있습니다. 자세한 내용은 Microsoft Technet 웹 사이트의 [실행 정책 소개](#)를 참조하십시오.

에는 AWS 모듈의 디렉터리 위치가 `PSModulePath` 포함되므로 `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 콘솔을 구성합니다. (AWS Tools for PowerShell Core AWSPowerShell NetCore 전용)

PowerShell Core는 일반적으로 모듈에서 cmdlet을 실행할 때마다 모듈을 자동으로 로드합니다. 하지만 이 경우에는 작동하지 않습니다. AWSPowerShell NetCore 크기가 크기 때문이죠. 달리기 시작하기 AWSPowerShell.NetCore cmdlet의 경우 먼저 명령을 실행해야 합니다. Import-Module AWSPowerShell.NetCore 이는 AWS.Tools 모듈의 cmdlet에서는 필요하지 않습니다.

## 세션 초기화 PowerShell

를 설치한 후 Linux 기반 또는 macOS 기반 시스템에서 시작하는 PowerShell 경우 [Initialize-를](#) 실행하여 AWS 사용할 액세스 키를 AWSDefaultConfiguration 지정해야 합니다. AWS Tools for PowerShellInitialize-AWSDefaultConfiguration에 대한 자세한 정보는 [AWS 자격 증명 사용](#).

### Note

의 이전 (3.3.96.0 이전) 릴리스에서는 이 cmdlet의 이름이 지정되었습니다. AWS Tools for PowerShellInitialize-AWSDefaults

## 버전 관리

AWS 새 서비스 및 기능을 지원하기 위해 AWS Tools for PowerShell 정기적으로 새 버전을 릴리스합니다. AWS Tools for PowerShell 설치한 버전을 확인하려면 [Get-AWSPowerShellVersion](#) cmdlet을 실행합니다.

```
PS > Get-AWSPowerShellVersion
```

```
Tools for PowerShell
```

```
Version 4.0.123.0
Copyright 2012-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Amazon Web Services SDK for .NET
Core Runtime Version 3.3.103.22
Copyright 2009-2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Release notes: https://github.com/aws/aws-tools-for-powershell/blob/master/CHANGELOG.md

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

현재 버전의 도구에서 지원되는 AWS 서비스 목록을 보려면 [Get-AWSPowerShellVersion](#) cmdlet에 `-ListServiceVersionInfo` 매개 변수를 추가하십시오.

실행 PowerShell 중인 버전을 확인하려면 `$PSVersionTable` [자동](#) 변수의 내용을 `$PSVersionTable` 보려면 `l` 입력하여 확인하십시오.

```
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
```

## 리눅스 또는 AWS Tools for PowerShell macOS에서 업데이트

정기적으로 의 업데이트된 버전이 AWS Tools for PowerShell 출시되면 로컬에서 실행 중인 버전을 업데이트해야 합니다.

### 모듈화된 모듈 **AWS.Tools** 업데이트

**AWS.Tools** 모듈을 최신 버전으로 업데이트하려면 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.

## PowerShell Core용 도구 업데이트

Get-AWSPowerShellVersioncmdlet을 실행하여 실행 중인 버전을 확인한 다음 [PowerShell 갤러리](#) 웹 사이트에서 PowerShell 제공되는 Windows용 도구 버전과 비교하십시오. 2~3주마다 확인하는 것이 좋습니다. 새 명령 및 AWS 서비스에 대한 지원은 해당 지원이 포함된 버전으로 업데이트한 후에만 사용할 수 있습니다.

의 AWSPowerShell 최신 릴리스를 설치하기 전. NetCore기존 모듈을 제거합니다. 기존 패키지를 제거하기 전에 열려 있는 PowerShell 세션을 모두 닫으십시오. 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 명령을 Import-Module AWSPowerShell.NetCore 실행하여 업데이트된 cmdlet을 세션에 로드합니다. PowerShell

## 관련 정보

- [AWS Tools for Windows PowerShell 시작](#)
- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)

## AWS Tools for PowerShell 버전 3.3에서 버전 4로 마이그레이션

AWS Tools for PowerShell 버전 4는 AWS Tools for PowerShell 3.3의 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

그 AWSPowerShell, NetCore 그리고 AWSPowerShell 패키지는 “모놀리식”이었습니다. 즉, 모든 AWS 서비스가 동일한 모듈에서 지원되었으며, 이로 인해 규모가 매우 커지고 각각의 새로운 AWS 서비스 및 기능이 추가될 때마다 규모가 점점 커졌습니다. 새 AWS.Tools 패키지는 소형 모듈로 분할되므로 사용 중인 AWS 서비스에 필요한 모듈만 다운로드하고 설치할 수 있는 유연성을 제공합니다. 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-module을 우선 호출하지 않고도 첫 번째 호출 시 cmdlet을 자동으로 가져올 수 있습니다. 하지만 cmdlet을 호출하기 전에 연관된 .NET 개체와 상호 작용하려면 여전히 관련 .NET 유형을 PowerShell 알리기 Import-Module 위해 호출해야 합니다.

예를 들어 다음 명령에는 Amazon.EC2.Model.Filter에 대한 참조가 있습니다. 이 유형의 참조는 자동 가져오기를 트리거할 수 없으므로 우선 Import-Module을 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

모듈의 `AWS.Tools` 컬렉션에서 각 AWS 서비스에 대한 모듈 이름을 보다 수월하게 찾으려면 `Get-AWSService` cmdlet을 사용합니다.

```
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. 그러면 AWS Tools for PowerShell 클라이언트가 응답을 PowerShell 스크립트에서 사용할 수 있는 개체로 변환하고 다른 명령으로 파이프합니다. 최종 PowerShell 개체의 원래 응답에 필요한 것보다 많은 필드나 속성이 있는 경우도 있고, 기본적으로 없는 응답의 필드나 속성을 개체에 포함해야 하는 경우도 있습니다. `-Select` 파라미터를 사용하면 cmdlet에서 반환된 .NET 객체에 포함시킬 사항을 지정할 수 있습니다.

예를 들어 [Get-S3Object cmdlet](#)은 [Amazon S3 SDK](#) 작업을 호출합니다. [ListObjectsListObjectsResponse](#)이 작업은 객체를 반환합니다. 하지만 기본적으로 `Get-S3Object` cmdlet은 SDK 응답의 `S3Objects` 요소만 사용자에게 반환합니다. PowerShell 다음 예제에서 이 객체는 2개 원소가 있는 배열입니다.

```
PS > Get-S3Object -BucketName mybucket

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

```
StorageClass : STANDARD

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

AWS Tools for PowerShell 버전 4에서는 SDK API 호출에 의해 반환된 완전한 .NET 응답 객체를 반환하도록 `-Select *`를 지정할 수 있습니다.

```
PS > Get-S3Object -BucketName mybucket -Select *
IsTruncated      : False
NextMarker       :
S3Objects        : {file1.txt, file2.txt}
Name             : mybucket
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 mybucket -Select S3Objects.Key
file1.txt
file2.txt
```

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

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

각 cmdlet에 대한 [참조 항목](#)에서 `-Select` 파라미터를 지원하는지 여부를 식별합니다.

## 출력 항목 수에 대한 보다 일관된 제한

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

이 동작은 `AWS.Tools`에서 제거됩니다.

이 동작은 에서 더 이상 사용되지 않습니다. `AWSPowerShell NetCore` 및 `AWSPowerShell` 향후 릴리스에서 해당 버전에서 제거될 예정입니다.

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 파이프하고 `-First n` 파라미터를 지정합니다. 여기서 `n`은 최종 출력에 포함할 최대 항목 수입니다.

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

모든 AWS 서비스가 동일한 방식으로 `-MaxItems`를 지원하지는 않으므로, 가끔씩 발생하는 불일치와 예기치 않은 결과가 제거됩니다. 또한 `-MaxItems`가 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 PowerShell은 UTF-8 인코딩을 사용하여 모든 문자열을 byte[]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='myBucket'; Key='file1.txt'; PartNumber=1 } | Get-S3ObjectMetadata
```

### Note

일부 속성은 AWS Tools for PowerShell의 이전 버전에서 이 기능을 지원했었습니다. 버전 4에서는 모든 파라미터에 대해 이를 활성화함으로써 일관성을 개선했습니다.

## 정적 공통 파라미터

AWS Tools for PowerShell의 버전 4.0에서는 일관성을 개선하기 위해 모든 파라미터가 정적 파라미터입니다.

AWS Tools for PowerShell의 이전 버전에서는 AccessKey, SecretKey, ProfileName 또는 Region과 같은 일부 공통 파라미터는 **동적** 파라미터였고, 기타 모든 파라미터는 정적 파라미터였습니다. 이렇게 하면 정적 매개변수가 동적 매개변수보다 먼저 PowerShell 바인딩되므로 문제가 발생할 수 있습니다. 예를 들어 다음 명령을 실행했다고 가정해보겠습니다.

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

이전 버전의 예에서는 -Region 동적 파라미터 대신 -RegionName 정적 us-west-2 파라미터에 값을 PowerShell 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. AWS서비스에서 API의 매개 변수가 필요하다고 선언하면 해당 cmdlet 매개 변수를 지정하지 않은 경우 해당 cmdlet

매개 변수를 PowerShell 입력하라는 메시지가 표시됩니다. 이것은 AWS.Tools에만 적용됩니다. 이전 버전과의 호환성을 보장하기 위해 이 내용은 적용되지 않습니다. AWSPowerShell NetCore 또는 AWSPowerShell.

## 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 검증은 실질적으로 우회할 수 있습니다. 하지만 AWS 서비스에서 여전히 해당 정보가 필요하기 때문에 Amazon EC2 API 서비스 호출이 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 기능은 AWS Tools for PowerShell의 이전 릴리스에서 더 이상 사용되지 않으며 버전 4에서 제거되었습니다.

- Stop-EC2Instance cmdlet에서 -Terminate 파라미터가 제거되었습니다. 대신 Remove-EC2Instance을 사용하세요.
- Clear-AWSCredential cmdlet에서 -ProfileName 매개 변수를 제거했습니다. 대신 Remove-AWSCredentialProfile을 사용하세요.
- cmdlet Import-EC2Instance 및 Import-EC2Volume이 제거되었습니다.

# AWS Tools for Windows PowerShell 시작

이 섹션의 일부 주제에서는 [도구를 설치](#)한 후 Tools for Windows PowerShell 사용의 기본 사항에 대해 설명합니다. 예를 들면 Tools for Windows PowerShell에서 AWS와 상호 작용할 때 사용해야 할 [보안 인증 정보](#)와 [AWS 리전](#)을 지정하는 방법을 설명합니다.

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

## 주제

- [다음을 사용하여 도구 인증을 구성합니다. AWS](#)
- [AWS 지역 지정하기](#)
- [AWS Tools for PowerShell를 사용하여 연동 자격 증명 구성](#)
- [Cmdlet 검색 및 별칭](#)
- [파이프라이닝 및 \\$AWSHistory](#)
- [보안 인증 정보 및 프로파일 확인](#)
- [사용자 및 역할에 대한 추가 정보](#)
- [레거시 보안 인증 사용](#)

## 다음을 사용하여 도구 인증을 구성합니다. AWS

를 사용하여 개발할 AWS 때 코드가 인증되는 방식을 설정해야 합니다. AWS 서비스 AWS 리소스에 대한 프로그래밍 방식 액세스를 구성할 수 있는 방법은 환경 및 사용 가능한 AWS 액세스에 따라 다릅니다.

도구의 다양한 인증 방법을 보려면 AWS SDK 및 도구 참조 안내서의 인증 및 [액세스](#)를 참조하십시오. PowerShell

이 항목에서는 신규 사용자가 현지에서 개발 중이고 고용주로부터 인증 방법을 제공받지 않았으며 임시 자격 증명을 얻기 AWS IAM Identity Center 위해 사용할 것으로 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. [IAM ID 센터를 PowerShell 사용하기 위한 도구를 구성하십시오.](#)
3. [AWS 액세스 포털 세션 시작](#)

## IAM Identity Center 사용 및 구성

사용하려면 AWS IAM Identity Center 먼저 활성화하고 구성해야 합니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [IAM Identity Center 인증](#) 항목의 1단계를 참조하십시오. PowerShell 특히 IAM Identity Center를 통한 액세스가 설정되어 있지 않습니다 아래에 있는 필요한 지침을 따르세요.

### IAM ID 센터를 PowerShell 사용하기 위한 도구를 구성하십시오.

#### Note

도구 버전 4.1.538부터 SSO 자격 증명을 구성하고 AWS 액세스 포털 세션을 시작하는 데 권장되는 방법은 이 항목에 설명된 대로 [Initialize-AWSSSOConfiguration](#) 및 [Invoke-AWSSSOLogin](#) cmdlet을 사용하는 것입니다. PowerShell 해당 버전의 도구 PowerShell (또는 이후 버전)에 액세스할 수 없거나 해당 cmdlet을 사용할 수 없는 경우에도 이를 사용하여 이러한 작업을 수행할 수 있습니다. AWS CLI 방법을 알아보려면 [포털 AWS CLI 로그인에 사용](#)을 참조하십시오.

다음 절차는 도구에서 임시 자격 증명을 얻는 데 PowerShell 사용하는 SSO 정보로 공유 AWS config 파일을 업데이트합니다. 이 절차를 수행하면 AWS 액세스 포털 세션도 시작됩니다. 공유 config 파일에 이미 SSO 정보가 있고 도구를 사용하여 액세스 포털 세션을 시작하는 방법을 알고 싶다면 이 항목의 다음 섹션을 참조하십시오. PowerShell [AWS 액세스 포털 세션 시작](#)

1. 아직 설치하지 않았다면 일반 cmdlet을 AWS Tools for PowerShell 포함하여 운영 체제와 환경에 맞게 파일을 열고 PowerShell 설치하십시오. 이를 위한 자세한 방법은 [AWS Tools for PowerShell 설치](#) 섹션을 참조하세요.

예를 들어 Windows용 PowerShell 도구의 모듈화된 버전을 설치하는 경우 다음과 비슷한 명령을 실행할 가능성이 큽니다.

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

2. 다음 명령을 실행합니다. 예제 속성 값을 IAM Identity Center 구성의 값으로 바꾸십시오. 이러한 속성과 검색 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [IAM Identity Center 자격 증명 공급자 설정](#)을 참조하십시오.

```
$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 cmdlet을 단독으로 사용해도 됩니다. 그러면 도구에 Initialize-AWSSSOConfiguration 속성 값을 PowerShell 입력하라는 메시지가 표시됩니다.

특정 속성값에 대한 고려 사항:

- 지침에 따라 [IAM Identity Center를 활성화하고 구성한](#) 경우의 값은 다음과 같을 -RoleName 수 PowerUserAccess 있습니다. 하지만 PowerShell 업무용으로 특별히 IAM Identity Center 권한 세트를 생성한 경우에는 그 권한 세트를 대신 사용하십시오.
  - IAM ID 센터를 구성한 AWS 리전 곳에서 사용해야 합니다.
3. 이때 공유 AWS config 파일에는 도구 항목에서 참조할 수 있는 구성 값 집합이 포함된 프로파일 이 들어 있습니다. my-sso-profile PowerShell 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 [공유 파일의 위치](#)를 참조하세요.

Tools for PowerShell에서는 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 획득합니다. AWSsso\_role\_name값은 IAM Identity Center 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용자에게 대한 액세스를 허용해야 합니다. AWS 서비스

다음 샘플은 위에 표시된 명령을 사용하여 만든 프로필을 보여줍니다. 일부 속성 값과 순서는 실제 프로필과 다를 수 있습니다. 프로필 sso-session 속성은 AWS 액세스 포털 세션을 시작하기 위한 설정이 포함된 이름이 지정된 my-sso-session 섹션을 나타냅니다.

```
[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 도구에서 이미 로그인했음을 PowerShell 알려줍니다.

그렇지 않은 경우 Tools for Tools를 사용하여 기본 웹 브라우저에서 SSO 인증 페이지를 자동으로 PowerShell 열려고 시도할 수 있습니다. 브라우저의 프롬프트를 따르십시오. 여기에는 SSO 인증 코드, 사용자 이름 및 암호, AWS IAM Identity Center 계정 및 권한 집합에 액세스할 수 있는 권한이 포함될 수 있습니다.

의 도구는 SSO PowerShell 로그인에 성공했음을 알려줍니다.

## AWS 액세스 포털 세션 시작

액세스 AWS 서비스명령을 실행하기 전에 도구에서 IAM Identity Center 인증을 사용하여 자격 증명을 PowerShell 확인할 수 있도록 활성 AWS 액세스 포털 세션이 필요합니다. AWS 액세스 포털에 로그인하려면 에서 PowerShell 다음 명령을 실행합니다. 여기서 `-ProfileName my-sso-profile` 는 이 항목의 이전 섹션에서 설명한 절차를 따랐을 때 공유 config 파일에 생성된 프로필의 이름입니다.

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

이미 활성 AWS 액세스 포털 세션이 있는 경우 도구에서 이미 로그인했음을 PowerShell 알려줍니다.

그렇지 않은 경우 Tools for Tools를 사용하여 기본 웹 브라우저에서 SSO 인증 페이지를 자동으로 PowerShell 열려고 시도할 수 있습니다. 브라우저의 프롬프트를 따르십시오. 여기에는 SSO 인증 코드, 사용자 이름 및 암호, AWS IAM Identity Center 계정 및 권한 집합에 액세스할 수 있는 권한이 포함될 수 있습니다.

의 도구는 SSO PowerShell 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-STSCallerIdentitycmdlet에 대한 응답은 공유 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고합니다. config

## 예

다음은 도구에서 IAM ID 센터를 사용하는 방법의 예입니다. PowerShell 이 예제에서는 다음을 가정합니다.

- 이 주제의 앞부분에서 설명한 대로 IAM Identity Center를 사용하도록 설정하고 구성했습니다. SSO 속성은 이 항목의 앞부분에서 구성한 my-sso-profile 프로필에 있습니다.
- Initialize-AWSSSOConfiguration 또는 Invoke-AWSSSOLogin cmdlet을 통해 로그인할 때 사용자는 Amazon S3에 대해 최소한 읽기 전용 권한을 가집니다.
- 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 도구 인증에 대한 PowerShell 추가 옵션은 AWS SDK 및 도구 참조 안내서의 [구성](#) 장을 참조하십시오.
- 일부 명령에는 AWS 지역을 지정해야 합니다. -Regioncmdlet 옵션, [default] 프로필, AWS\_REGION 환경 변수 등 여러 가지 방법으로 이 작업을 수행할 수 있습니다. 자세한 내용은 이 안내서의 [AWS 지역 지정하기](#) 내용과 AWS SDK 및 도구 참조 안내서의 [AWS 지역을](#) 참조하십시오.
- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 [IAM의 보안 모범 사례](#)를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하십시오.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 [표준화된 보안 인증 공급자](#)를 참조하세요.

## 주제

- [포털 AWS CLI 로그인에 사용](#)

## 포털 AWS CLI 로그인에 사용

도구 버전 4.1.538부터 SSO 자격 증명을 구성하고 AWS 액세스 포털 세션을 시작하는 데 권장되는 방법은 에 설명된 대로 [Initialize-AWSSSOConfiguration](#) 및 [Invoke-AWSSSOLogin](#) cmdlet 을 사용하는 것입니다. PowerShell [다음을 사용하여 도구 인증을 구성합니다. AWS](#) 해당 버전의 도구 PowerShell (또는 이후 버전) 에 액세스할 수 없거나 해당 cmdlet을 사용할 수 없는 경우에도 를 사용하여 이러한 작업을 수행할 수 있습니다. AWS CLI

를 통해 IAM Identity PowerShell Center를 사용하도록 도구를 구성하십시오. AWS CLI

아직 설정하지 않았다면 계속 진행하기 전에 [IAM ID 센터를 활성화하고 구성해야](#) 합니다.

를 통해 IAM Identity PowerShell Center를 사용하도록 도구를 구성하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [IAM Identity Center 인증](#) 항목의 2단계에 나와 있습니다. AWS CLI 이 구성을 완료하면 시스템에 다음 요소가 포함되어야 합니다.

- 는 AWS CLI 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 사용합니다.
- 도구에서 참조할 수 있는 구성 값 집합이 포함된 [\[default\] 프로필이](#) 포함된 공유 AWS config 파일입니다. PowerShell 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 [공유 파일의 위치](#)를 참조하세요. Tools for PowerShell 에서는 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 획득합니다. AWSsso\_role\_name 값은 IAM Identity Center 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용자에게 대한 액세스를 허용해야 합니다. AWS 서비스

다음 샘플 config 파일은 SSO 토큰 공급자로 설정한 [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 해상도가 작동하려면 PowerShell 세션에 다음 모듈을 설치하고 가져와야 합니다.

- `AWS.Tools.SSO`
- `AWS.Tools.SSOIDC`

이전 버전의 Tools를 사용하고 있는데 이러한 모듈이 없으면 다음과 비슷한 오류가 발생합니다. “AWSSDKAssembly.SSOIDC를 찾을 수 없습니다...”. PowerShell

## 액세스 포털 세션 시작 AWS

액세스하는 AWS 서비스명령을 실행하기 전에 Windows용 도구에서 IAM Identity Center 인증을 사용하여 자격 증명을 PowerShell 확인할 수 있도록 활성 AWS 액세스 포털 세션이 필요합니다. 구성된 세션 길이에 따라 액세스는 결국 PowerShell 만료되며 Windows용 도구에서 인증 오류가 발생합니다. AWS 액세스 포털에 로그인하려면 에서 다음 명령을 실행합니다. AWS CLI

```
aws sso login
```

[default]프로필을 사용하고 있으므로 `--profile` 옵션을 사용하여 명령을 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 명령이 `aws sso login --profile named-profile` 대신 사용됩니다. 명명된 프로필에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [프로필](#) 섹션을 참조하십시오.

이미 활성 세션이 있는지 테스트하려면 명명된 프로필도 동일하게 고려하여 다음 AWS CLI 명령어를 실행하세요.

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

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

**ℹ Note**

이미 활성 AWS 액세스 포털 세션이 있고 실행 `aws sso login` 중인 경우에는 자격 증명을 제공할 필요가 없습니다.

로그인 과정에서 데이터에 AWS CLI 대한 접근을 허용하라는 메시지가 표시될 수 있습니다. AWS CLI 는 Python용 SDK를 기반으로 구축되었으므로 권한 메시지에 다양한 `botocore` 이름이 포함될 수 있습니다.

## 예

다음은 도구에서 IAM Identity Center를 사용하는 방법의 예입니다. PowerShell 이 예제에서는 다음을 가정합니다.

- 이 주제의 앞부분에서 설명한 대로 IAM Identity Center를 사용하도록 설정하고 구성했습니다. SSO 속성은 [default] 프로필에 있습니다.
- 를 사용하여 `aws sso login` 로그인하면 해당 사용자는 Amazon AWS CLI S3에 대해 최소한 읽기 전용 권한을 가집니다.
- 해당 사용자는 일부 S3 버킷을 볼 수 있습니다.

다음 PowerShell 명령을 사용하여 S3 버킷 목록을 표시할 수 있습니다.

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule S3
# And if using an older version of the AWS Tools for PowerShell:
Install-AWSToolsModule SS0, SS00IDC

# In older versions of the AWS Tools for PowerShell, we're not invoking a cmdlet from
these modules directly,
# so we must import them explicitly:
Import-Module AWS.Tools.SS0
Import-Module AWS.Tools.SS00IDC

# Older versions of the AWS Tools for PowerShell don't support the SS0 login flow, so
login with the CLI
aws sso login

# Now we can invoke cmdlets using the SS0 profile
Get-S3Bucket
```

위에서 설명한 것처럼 [default] 프로필을 사용하고 있으므로 옵션을 사용하여 `Get-S3Bucket` cmdlet을 호출할 필요가 없습니다. `-ProfileName SS0` 토큰 공급자 구성에서 명명된 프로필을 사용

하는 경우 `Get-S3Bucket -ProfileName named-profile` 명령을 사용합니다. 명명된 프로필에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [프로필](#) 섹션을 참조하십시오.

## 추가 정보

- 프로필 및 환경 변수 사용과 같은 도구 인증에 대한 PowerShell 자세한 옵션은 AWS SDK 및 도구 참조 안내서의 [구성](#) 장을 참조하십시오.
- 일부 명령에는 AWS 지역을 지정해야 합니다. `-Regioncmdlet` 옵션, [default] 프로필, `AWS_REGION` 환경 변수 등 여러 가지 방법으로 이 작업을 수행할 수 있습니다. 자세한 내용은 이 안내서의 [AWS 지역 지정하기](#) 내용과 AWS SDK 및 도구 참조 안내서의 [AWS 지역을](#) 참조하십시오.
- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 [IAM의 보안 모범 사례](#)를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 [임시 보안 자격 증명](#)을 참조하십시오.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 [표준화된 보안 인증 공급자](#)를 참조하세요.

## AWS 지역 지정하기

AWS Tools for PowerShell 명령을 실행할 때 사용할 AWS 지역을 지정하는 방법은 두 가지가 있습니다.

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

Windows용 도구에서 사용할 지역을 파악할 PowerShell 수 없는 경우 대부분의 AWS cmdlet이 실패합니다. 예외로는 [Amazon S3](#), [Amazon SES](#) 및 `aws iam` cmdlet이 포함되며 AWS Identity and Access Management, 이 cmdlet은 자동으로 글로벌 엔드포인트로 기본 설정됩니다.

단일 명령에 사용할 지역을 지정하려면 AWS

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. 모든 PowerShell 세션에 설정을 적용하려면 명령에서 했던 것처럼 이 명령을 PowerShell 프로필에 Import-Module 추가합니다.

모든 AWS CLI 명령의 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 명령의 현재 기본 리전을 지우려면

PowerShell 명령 프롬프트에서 다음 명령을 입력합니다.

```
PS > Clear-DefaultAWSRegion
```

사용 가능한 모든 AWS 지역 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

다음 샘플 형식으로 Windows용 도구 PowerShell 명령에 -EndpointUrl 공통 매개 변수를 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-AWSRegion을 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 지역에 대한 추가 정보는 AWS SDK 및 도구 참조 안내서의 [AWS 지역](#)을 참조하십시오.

## AWS Tools for PowerShell를 사용하여 연동 자격 증명 구성

조직의 사용자가 AWS 리소스에 액세스하도록 하려면 보안, 감사, 규정 준수, 그리고 역할 및 계정 분리 지원을 위해 반복 가능한 표준 인증 방법을 구성해야 합니다. 사용자에게 AWS API에 액세스하는 기능을 제공하는 것이 일반적이지만, 연동 API 액세스 권한이 없는 경우 AWS Identity and Access Management(IAM) 사용자를 생성해야 하는데, 이로 인해 연동 사용의 목적이 무효화됩니다. 이 주제에서는 연동된 액세스 솔루션을 용이하게 하는 AWS Tools for PowerShell의 SAML(Security Assertion Markup Language) 지원에 대해 설명합니다.

AWS Tools for PowerShell의 SAML 지원 덕분에 사용자에게 AWS 서비스에 대한 연동 액세스를 제공할 수 있습니다. SAML은 서비스 간에, 특히 자격 증명 공급자([Active Directory Federation Services](#) 등)와 서비스 공급자(AWS) 간에, 사용자 인증 및 권한 부여 데이터를 전송하기 위한 XML 기반의 개방형 표준 형식입니다. SAML과 그 작동 방식에 대한 자세한 내용은 Wikipedia의 [SAML](#) 또는 Organization for the Advancement of Structured Information Standards(OASIS) 웹 사이트의 [SAML Technical Specifications](#)를 참조하십시오. AWS Tools for PowerShell의 SAML 지원은 SAML 2.0과 호환됩니다.

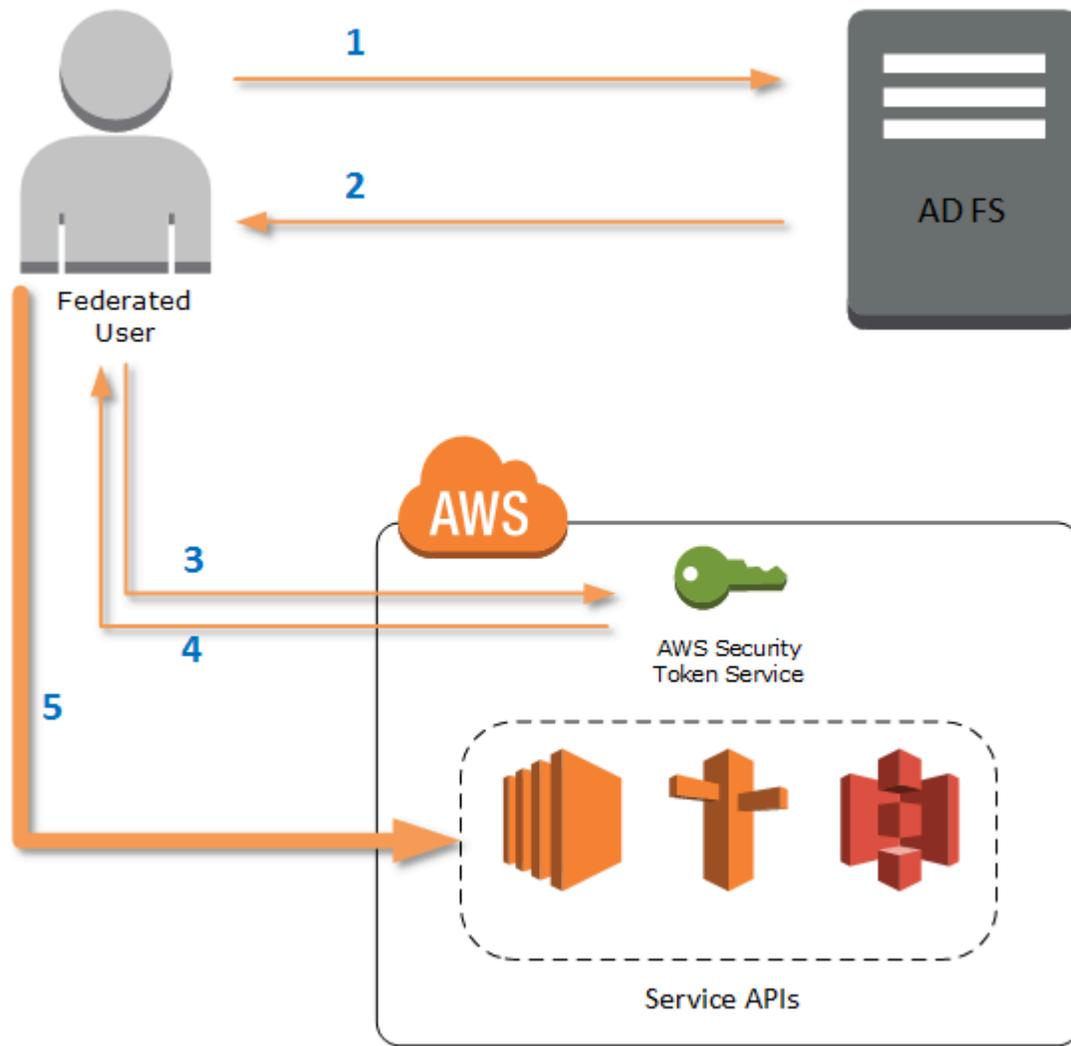
## 필수 조건

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. Active Directory Federation Services에 대해 특별히 이 작업을 수행하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [SAML 2.0 연동 정보](#) 및 블로그 게시물인 [Windows Active Directory, AD FS 및 SAML 2.0을 사용해 AWS에 대한 연동 활성화](#)를 참조하세요. 블로그 게시물에서는 AD FS 2.0을 설명하지만, AD FS 3.0을 실행 중인 경우라도 단계는 비슷합니다.
- 버전 3.1.31.0 이상의 AWS Tools for PowerShell이 로컬 워크스테이션에 설치되어 있어야 합니다.

## 자격 증명 연동 사용자가 AWS 서비스 API에 대한 연동된 액세스 권한을 얻는 방법

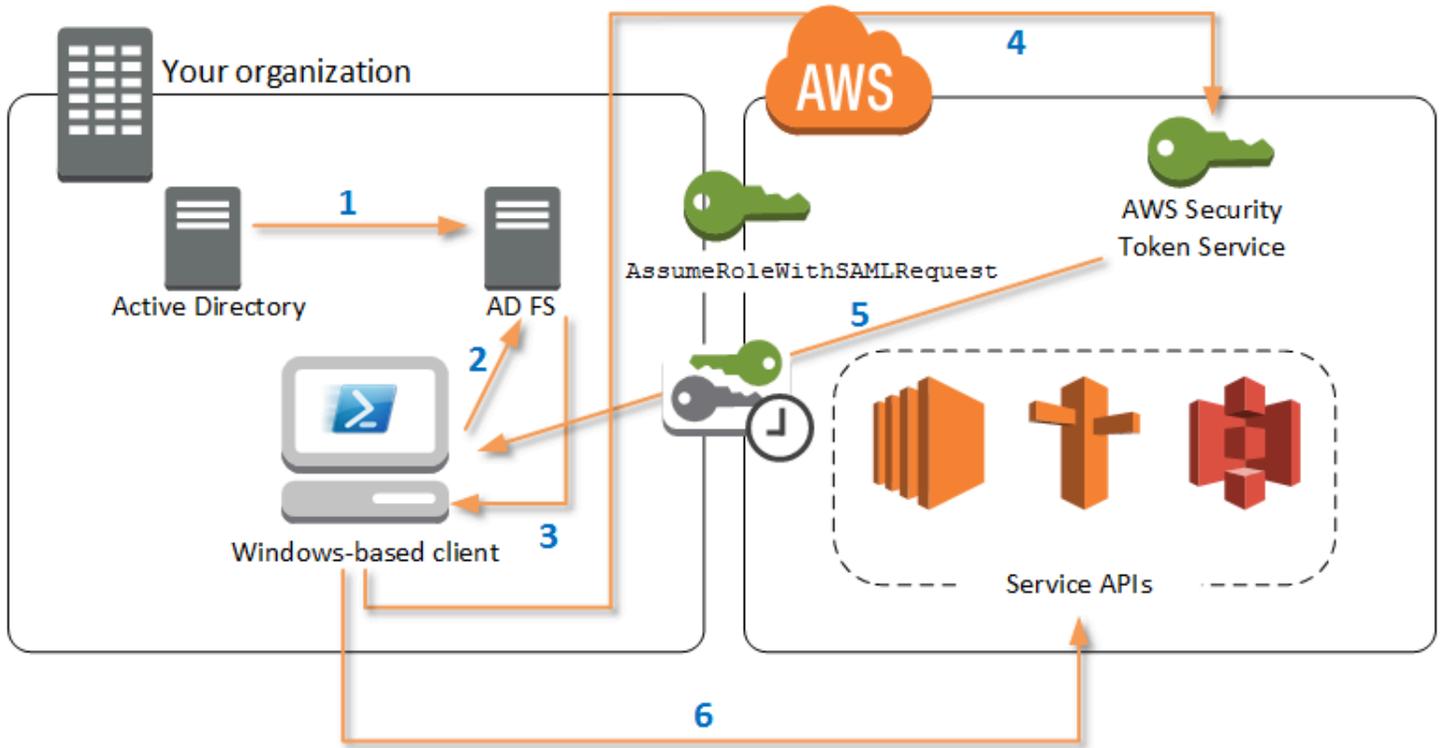
다음 프로세스에서는 Active Directory(AD) 사용자가 AD FS에 의해 연동되어 AWS 리소스에 대한 액세스 권한을 얻는 방법을 개략적으로 설명합니다.



1. 연동 사용자 컴퓨터의 클라이언트는 AD FS에 대해 인증합니다.
2. 인증에 성공하면 AD FS는 사용자에게 SAML 어설션을 보냅니다.
3. 사용자의 클라이언트는 SAML 연동 요청의 일환으로 AWS Security Token Service(STS)에 SAML 어설션을 보냅니다.
4. STS는 사용자가 수입할 수 있는 역할에 대한 AWS 임시 자격 증명이 포함된 SAML 응답을 반환합니다.
5. 사용자는 AWS Tools for PowerShell에서 이루어진 요청에 이러한 임시 자격 증명을 포함시켜 AWS 서비스 API에 액세스합니다.

## AWS Tools for PowerShell에서 SAML 지원이 작동하는 방식

이 단원에서는 AWS Tools for PowerShell cmdlet에서 사용자에게 대한 SAML 기반 자격 증명 연동의 구성을 활성화하는 방법을 설명합니다.



1. AWS Tools for PowerShell은 Windows 사용자의 현재 자격 증명을 사용하여, 또는 (사용자가 AWS 로의 호출을 위해 자격 증명을 필요로 하는 cmdlet을 실행할 때는) 대화식으로 AD FS에 대해 인증합니다.
2. AD FS에서 사용자를 인증합니다.
3. AD FS는 어설션을 포함하는 SAML 2.0 인증 응답을 생성합니다. 어설션의 목적은 사용자에게 대한 정보를 식별하고 제공하는 것입니다. AWS Tools for PowerShell은 SAML 어설션으로부터 사용자의 인증된 역할 목록을 추출합니다.
4. AWS Tools for PowerShell은 AssumeRoleWithSAMLRequest API 호출을 통해 요청된 역할의 Amazon 리소스 이름(ARN)을 포함하여 SAML 요청을 STS로 전달합니다.
5. SAML 요청이 유효하면 STS는 AWS, AccessKeyId, SecretAccessKey 및 SessionToken을 포함하는 응답을 반환합니다. 이러한 자격 증명은 3,600초(1시간) 동안 유효합니다.
6. 이제 사용자는 사용자의 역할에 액세스 권한이 부여된 모든 AWS 서비스 API에서 사용할 수 있는 유효한 자격 증명을 가집니다. AWS Tools for PowerShell은 모든 후속 AWS API 호출에 대해 이러한 자격 증명을 자동으로 적용하고, 자격 증명이 만료되면 자동으로 갱신합니다.

**Note**

자격 증명이 만료되어 새 자격 증명에 필요한 경우 AWS Tools for PowerShell은 AD FS에서 자동으로 재인증을 수행하여 이후 1시간 동안 유효한 새 자격 증명을 획득합니다. 도메인에 로그인된 계정의 사용자의 경우, 이 프로세스가 자동으로 수행됩니다. 도메인에 로그인되지 않은 계정의 경우, AWS Tools for PowerShell에서는 재인증 전에 자격 증명을 입력하라는 메시지가 사용자에게 표시됩니다.

## PowerShell SAML 구성 Cmdlet 사용 방법

AWS Tools for PowerShell에는 SAML 지원을 제공하는 새로운 cmdlet 두 개가 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 프로파일과 마찬가지로, 표시 이름을 역할 프로파일에 할당할 수 있습니다. 동일한 표시 이름을 `Set-AWSCredential` cmdlet과 함께 사용하거나, AWS 서비스 API를 호출하는 cmdlet에 대해 `-ProfileName` 파라미터의 값으로 사용할 수 있습니다.

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"
```

- 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-AWSSamlRoleProfile`을 실행할 때 이 이름을 사용할 수 있습니다.

- 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 도메인에 조인되지 않은 계정의 경우, 다음 행과 같이, Active Directory 자격 증명을 제공하고 나서 사용자에게 액세스 권한이 부여된 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 실행하기 위해 AWS 공유 자격 증명 파일에 정의된 역할 프로파일을 사용할 수 있습니다. 역할 프로파일의 이름을 `Set-AWSCredential`에 입력하여(또는 AWS Tools for PowerShell에 `ProfileName` 파라미터 값 형태로 입력하여) 프로파일에 설명된 역할에 대해 임시 AWS 자격 증명을 자동으로 가져옵니다.

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` 프로파일을 통해 획득한 임시 AWS 자격 증명을 사용하여 AWS 서비스 API에서 작업을 수행할 수 있습니다. 다음 단원에서는 역할 프로파일 사용법 예를 보여 줍니다.

## 예제 1: Set-AWSCredential을 사용하여 기본 역할 지정

이 예제는 Set-AWSCredential을 사용하여 AWS Tools for PowerShell 세션의 기본 역할을 설정합니다. 그리고 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 예제는 SAMLDemoProfile 프로파일과 연결된 역할의 AWS 계정에서 사용 가능한 모든 Amazon S3 버킷을 나열합니다. 이 예제는 AWS Tools for PowerShell 세션 초기에 다른 프로파일을 사용했을 수도 있지만 이를 지원하는 cmdlet을 사용하여 -ProfileName 파라미터에 다른 값을 지정함으로써 프로파일을 변경할 수 있다는 것을 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	mybucket1
4/15/2015 12:46:50 AM	mybucket2
4/15/2015 6:15:53 AM	mybucket3
1/12/2015 11:20:16 PM	mybucket4

Get-S3Bucket cmdlet은 Set-AWSSamlRoleProfile cmdlet을 실행하여 생성된 프로파일의 이름을 지정합니다. 이 명령은 (예를 들면 Set-AWSCredential cmdlet을 실행하여) 세션의 초기에 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 PowerShell은 도구가 현재 자격 증명만료되었음을 감지할 때 새 SAML 어설션을 요청하여 자격 증명을 자동으로 새로 고칩니다.

도메인에 조인된 사용자의 경우, 인증 시 현재 사용자의 Windows 자격 증명 사용되므로 이 프로세스가 중단 없이 발생합니다. 도메인에 조인되지 않은 사용자 계정의 경우, AWS Tools for PowerShell에서 사용자 암호를 요청하는 PowerShell 자격 증명 프롬프트가 표시됩니다. 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 2.0](#)을 참조하십시오.

질문이나 의견이 있는 경우에는 [PowerShell 스크립팅](#) 또는 [.NET 개발](#)을 위한 AWS 개발자 포럼을 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 참조](#)를 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 접두어나 AWS 서비스 이름의 일부를 포함시킬 수 있습니다. 예를 들면 Amazon EC2의 cmdlet 명사 접두사는 EC2입니다. 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
Get-GMLInstance	DescribeInstances	Amazon GameLift Service	GML

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	ServiceName	ServiceOperation	CmdletNounPrefix
-----	-----	-----	-----
Approve-ECCacheSecurityGroupIngress	Amazon ElastiCache	AuthorizeCacheSecurityGroupIngress	EC
Get-ECCacheSecurityGroup	Amazon ElastiCache	DescribeCacheSecurityGroups	EC
New-ECCacheSecurityGroup	Amazon ElastiCache	CreateCacheSecurityGroup	EC
Remove-ECCacheSecurityGroup	Amazon ElastiCache	DeleteCacheSecurityGroup	EC
Revoke-ECCacheSecurityGroupIngress	Amazon ElastiCache	RevokeCacheSecurityGroupIngress	EC
Add-EC2SecurityGroupToClientVpnTargetNetwork	Amazon Elastic Compute Cloud	ApplySecurityGroupsToClientVpnTargetNetwork	EC2
Get-EC2SecurityGroup	Amazon Elastic Compute Cloud	DescribeSecurityGroups	EC2
Get-EC2SecurityGroupReference	Amazon Elastic Compute Cloud	DescribeSecurityGroupReferences	EC2
Get-EC2StaleSecurityGroup	Amazon Elastic Compute Cloud	DescribeStaleSecurityGroups	EC2
Grant-EC2SecurityGroupEgress	Amazon Elastic Compute Cloud	AuthorizeSecurityGroupEgress	EC2
Grant-EC2SecurityGroupIngress	Amazon Elastic Compute Cloud	AuthorizeSecurityGroupIngress	EC2
New-EC2SecurityGroup	Amazon Elastic Compute Cloud	CreateSecurityGroup	EC2
Remove-EC2SecurityGroup	Amazon Elastic Compute Cloud	DeleteSecurityGroup	EC2
Revoke-EC2SecurityGroupEgress	Amazon Elastic Compute Cloud	RevokeSecurityGroupEgress	EC2

Revoke-EC2SecurityGroupIngress	Amazon Elastic Compute Cloud	EC2	RevokeSecurityGroupIngress
Update-EC2SecurityGroupRuleEgressDescription	Amazon Elastic Compute Cloud	EC2	UpdateSecurityGroupRuleDescriptionsEgress
Update-EC2SecurityGroupRuleIngressDescription	UpdateSecurityGroupRuleDescriptionsIngress	Amazon Elastic Compute Cloud	EC2
Edit-EFSMountTargetSecurityGroup	Amazon Elastic File System	EFS	ModifyMountTargetSecurityGroups
Get-EFSMountTargetSecurityGroup	Amazon Elastic File System	EFS	DescribeMountTargetSecurityGroups
Join-ELBSecurityGroupToLoadBalancer	Elastic Load Balancing	ELB	ApplySecurityGroupsToLoadBalancer
Set-ELB2SecurityGroup	Elastic Load Balancing V2	ELB2	SetSecurityGroups
Enable-RDSDBSecurityGroupIngress	Amazon Relational Database Service	RDS	AuthorizeDBSecurityGroupIngress
Get-RDSDBSecurityGroup	Amazon Relational Database Service	RDS	DescribeDBSecurityGroups
New-RDSDBSecurityGroup	Amazon Relational Database Service	RDS	CreateDBSecurityGroup
Remove-RDSDBSecurityGroup	Amazon Relational Database Service	RDS	DeleteDBSecurityGroup
Revoke-RDSDBSecurityGroupIngress	Amazon Relational Database Service	RDS	RevokeDBSecurityGroupIngress
Approve-RSClusterSecurityGroupIngress	Amazon Redshift	RS	AuthorizeClusterSecurityGroupIngress
Get-RSClusterSecurityGroup	Amazon Redshift	RS	DescribeClusterSecurityGroups
New-RSClusterSecurityGroup	Amazon Redshift	RS	CreateClusterSecurityGroup
Remove-RSClusterSecurityGroup	Amazon Redshift	RS	DeleteClusterSecurityGroup
Revoke-RSClusterSecurityGroupIngress	Amazon Redshift	RS	RevokeClusterSecurityGroupIngress

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 SecurityGroup을 포함하는 모든 cmdlet 이름을 모두 가져오려면 다음 명령을 실행합니다.

```
PS > Get-AWSCmdletName -ApiOperation SecurityGroup -MatchWithRegex -Service EC2
```

CmdletName	ServiceOperation
ServiceName	CmdletNounPrefix
-----	-----
-----	-----
Add-EC2SecurityGroupToClientVpnTargetNetwrk	Amazon Elastic Compute Cloud EC2
ApplySecurityGroupsToClientVpnTargetNetwork	DescribeSecurityGroups
Get-EC2SecurityGroup	DescribeSecurityGroupReferences
Amazon Elastic Compute Cloud EC2	DescribeStaleSecurityGroups
Get-EC2SecurityGroupReference	AuthorizeSecurityGroupEgress
Amazon Elastic Compute Cloud EC2	AuthorizeSecurityGroupIngress
Get-EC2StaleSecurityGroup	CreateSecurityGroup
Amazon Elastic Compute Cloud EC2	DeleteSecurityGroup
Grant-EC2SecurityGroupEgress	RevokeSecurityGroupEgress
Amazon Elastic Compute Cloud EC2	RevokeSecurityGroupIngress
Grant-EC2SecurityGroupIngress	UpdateSecurityGroupRuleDescriptionsEgress
Amazon Elastic Compute Cloud EC2	
New-EC2SecurityGroup	
Amazon Elastic Compute Cloud EC2	
Remove-EC2SecurityGroup	
Amazon Elastic Compute Cloud EC2	
Revoke-EC2SecurityGroupEgress	
Amazon Elastic Compute Cloud EC2	
Revoke-EC2SecurityGroupIngress	
Amazon Elastic Compute Cloud EC2	
Update-EC2SecurityGroupRuleEgressDescription	
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-CWResourceTag	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-CWResourceTag	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 S3GetObject 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.txt
```

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

### 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 동사](#)를 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.

## 별칭

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 Where-Object cmdlet의 ? 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 AWSPowerShell	Edit-RSClusterIamRoles	3.3.343.0
Alias AWSPowerShell	Enable-ORGAllFeatures	3.3.343.0
Alias AWSPowerShell	Find-CTEvents	3.3.343.0
Alias AWSPowerShell	Get-ASACases	3.3.343.0
Alias AWSPowerShell	Get-ASAccountLimits	3.3.343.0
Alias AWSPowerShell	Get-ASACommunications	3.3.343.0
Alias AWSPowerShell	Get-ASAServices	3.3.343.0
Alias AWSPowerShell	Get-ASASeverityLevels	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckRefreshStatuses	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorChecks	3.3.343.0
Alias AWSPowerShell	Get-ASATrustedAdvisorCheckSummaries	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHooks	3.3.343.0
Alias AWSPowerShell	Get-ASLifecycleHookTypes	3.3.343.0
Alias AWSPowerShell	Get-AWSCredentials	3.3.343.0
Alias AWSPowerShell	Get-CDApplications	3.3.343.0
Alias AWSPowerShell	Get-CDDeployments	3.3.343.0
Alias AWSPowerShell	Get-CFCloudFrontOriginAccessIdentities	3.3.343.0
Alias AWSPowerShell	Get-CFDistributions	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigRules	3.3.343.0
Alias AWSPowerShell	Get-CFGConfigurationRecorders	3.3.343.0
Alias AWSPowerShell	Get-CFGDeliveryChannels	3.3.343.0
Alias AWSPowerShell	Get-CFInvalidations	3.3.343.0

```
Alias          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 변수](#)를 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
```

## 파이프라이닝 및 \$AWSHistory

컬렉션을 반환하는 AWS 서비스 호출의 경우, 컬렉션 내의 개체가 파이프라인에 열거됩니다. 모음 이외에 페이징 제어 필드가 아닌 추가 필드를 포함하는 결과 객체에는 이러한 필드가 호출에 대한 Note 속성으로 추가됩니다. 이러한 Note 속성은 새 `$AWSHistory` 세션 변수에 기록되며, 이 데이터에 액세스해야 합니다. `$AWSHistory` 변수는 다음 단원에서 설명합니다.

### Note

v1.1 이전 버전의 Tools for Windows PowerShell에서는 모음 객체가 자체적으로 내보내지므로 파이프라이닝을 계속하려면 `foreach {$_ .getenumerator() }`를 사용해야 했습니다.

예

다음 예제에서는 각 리전에서 AWS 리전 및 Amazon EC2 머신 이미지(AMI) 목록을 반환합니다.

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

다음 예제에서는 현재 기본 리전의 모든 Amazon EC2 인스턴스를 중지합니다.

PS > **Get-EC2Instance | Stop-EC2Instance**

컬렉션은 파이프라인에 열거되므로 해당 cmdlet의 출력은 단일 객체 또는 컬렉션인 \$null가 될 수 있습니다. 모음인 경우 .Count 속성을 사용하여 모음이 크기를 결정할 수 있습니다. 하지만 단일 객체만 내보낼 때는 .Count 속성이 없습니다. 스크립트에서 방출된 객체 수를 일관된 방식으로 확인해야 하는 경우, \$AWSHistory에서 마지막 명령 값의 EmittedObjectsCount 속성을 확인할 수 있습니다.

## \$AWSHistory

파이프라이닝을 더 잘 지원하기 위해 AWS 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 AWS SDK 응답과 결과 데이터가 파이프에 존재할 수 없다는 뜻입니다.

대부분의 사용자에게는 이 데이터가 필요하지 않겠지만, cmdlet의 기본 AWS 서비스 호출로/에서 보내고 받은 내용을 정확히 알 수 있으므로 진단 목적으로 유용할 수 있습니다.

버전 1.1부터는 이 데이터와 그 밖의 데이터를 \$AWSHistory라는 새 셸 변수에서 사용할 수 있습니다. 이 변수는 AWS cmdlet 호출과 각 호출에 대해 수신된 서비스 응답에 관한 기록을 유지합니다. 선택적으로 이 기록은 각 cmdlet의 서비스 요청을 기록하도록 구성할 수 있습니다. cmdlet의 전체 실행 시간 같은 유용한 추가 데이터도 각 항목에서 얻을 수 있습니다. 보안상의 이유로 민감한 데이터가 포함된 요청 및 응답은 기본적으로 기록되지 않습니다. 그러나 필요한 경우 이 동작을 재정의하도록 기록을 구성할 수 있습니다. 자세한 내용은 아래의 Set-AWSHistoryConfiguration cmdlet 섹션을 참조하세요.

\$AWSHistory.Commands 목록의 각 항목은 AWSCmdletHistory 유형입니다. 이 유형에는 다음과 같은 유용한 멤버가 있습니다.

### CmdletName

cmdlet의 이름입니다.

### CmdletStart

cmdlet이 실행된 날짜/시간입니다.

### CmdletEnd

cmdlet이 모든 처리를 완료한 날짜/시간입니다.

## 요청

요청 기록이 활성화된 경우 마지막 서비스 요청의 목록입니다.

## 응답

수신된 마지막 서비스 응답의 목록입니다.

## LastServiceResponse

최근 서비스 응답을 반환하는 헬퍼입니다.

## LastServiceRequest

가장 최근의 서비스 응답을 반환하는 헬퍼입니다(사용 가능한 경우).

서비스를 호출하는 AWS cmdlet이 사용될 때까지는 \$AWSHistory 변수가 생성되지 않습니다. 해당 시점까지는 \$null로 평가됩니다.

### Note

Tools for Windows PowerShell의 이전 버전에서는 서비스 응답과 관련된 데이터를 반환된 객체의 Note 속성으로 내보냈습니다. 이제는 이러한 데이터가 목록의 각 호출에 대해 기록되는 응답 항목에 있습니다.

## Set-AWSHistoryConfiguration

cmdlet 호출은 0개 이상의 서비스 요청 및 응답 항목을 유지할 수 있습니다. 메모리 영향을 제한하기 위해 \$AWSHistory 목록은 기본적으로 마지막 다섯 개 cmdlet 실행과 각 실행마다 마지막 다섯 개 서비스 응답(그리고 활성화된 경우 마지막 서비스 요청)에 대한 기록만 보관합니다. Set-AWSHistoryConfiguration cmdlet을 실행하여 이러한 기본 제한을 변경할 수 있습니다. 항상 목록의 크기와 서비스 요청의 기록 여부를 둘 다 제어할 수 있습니다.

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

모든 파라미터는 선택 사항입니다.

MaxCmdletHistory 파라미터는 언제든지 추적할 수 있는 최대 cmdlet 수를 설정합니다. 값을 0으로 설정하면 AWS cmdlet 활동 기록이 해제됩니다. MaxServiceCallHistory 파라미터는 각 cmdlet에

대해 추적되는 최대 서비스 응답 수(및/또는 요청 수)를 설정합니다. `RecordServiceRequests` 파라미터(지정한 경우)는 각 cmdlet에 대해 서비스 요청 추적을 설정합니다. `IncludeSensitiveData` 파라미터는 지정된 경우 각 cmdlet의 중요한 데이터가 포함된 서비스 응답 및 요청에 대한 추적을 활성화합니다(추적되는 경우).

파라미터 없이 실행할 경우 `Set-AWSHistoryConfiguration`은 현재 목록 크기를 변경하지 않고, 이전 요청 기록만 모두 해제합니다.

현재 기록 목록의 모든 항목을 지우려면 `Clear-AWSHistory` cmdlet을 실행합니다.

## \$AWSHistory 예

목록에서 유지되고 있는 AWS cmdlet의 세부 정보를 파이프라인에 열거합니다.

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

실행된 마지막 AWS cmdlet의 세부 정보에 액세스합니다.

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

실행된 마지막 AWS cmdlet에서 수신된 마지막 서비스 응답의 세부 정보에 액세스합니다. AWS cmdlet이 출력을 페이징하고 있는 경우 서비스를 여러 번 호출하여 모든 데이터를 수집하거나 최대량의 데이터(cmdlet의 파라미터에 의해 결정됨)를 가져올 수 있습니다.

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

마지막 요청의 세부 정보에 액세스합니다. 위에서도 설명했듯이, cmdlet은 사용자를 대신하여 페이징 중인 경우 두 개 이상의 요청을 만들 수도 있습니다. 서비스 요청 추적이 활성화되지 않은 경우 `$null`을 생성합니다.

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

## 여러 페이지를 반환하는 작업에 대한 자동 Page-to-Completion

해당 호출에 대해 기본 최대 객체 반환 수를 적용하거나 페이징 가능 결과 세트를 지원하는 서비스 API의 경우 기본적으로 모든 cmdlet "page-to-completion"입니다. 각 cmdlet은 전체 데이터 세트를 파이프라인으로 반환하기 위해 사용자를 대신하여 필요한 만큼 호출합니다.

Get-S3Object를 사용하는 다음 예제에서는 \$c 변수가 test 버킷(잠재적으로 매우 큰 데이터 세트)의 모든 키에 대한 S3Object 인스턴스를 포함합니다.

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

반환되는 데이터의 양을 계속 제어하려는 경우, 개별 cmdlet에서 매개 변수를 계속 사용하거나(예: Get-S3Object에서 MaxKey를 사용) cmdlet의 페이징 매개 변수 조합과 서비스의 다음 토큰 데이터를 가져오기 위해 \$AWSHistory 변수에 배치된 데이터를 사용하여 자체적으로 페이징을 명시적으로 처리할 수 있습니다. 다음 예제에서는 MaxKeys 파라미터를 사용하여 반환되는 S3Object 인스턴스의 수를 버킷에서 처음 발견되는 500개 이하로 제한합니다.

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

추가 데이터가 있지만 반환되지 않았는지 여부를 알아보려면 cmdlet에 의한 서비스 호출을 기록한 \$AWSHistory 세션 변수 항목을 사용합니다.

다음 표현식이 \$true로 평가되는 경우, \$AWSHistory.LastServiceResponse.NextMarker를 사용하여 다음 결과 세트에 대한 next 마커를 찾을 수 있습니다.

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

Get-S3Object를 사용하여 페이징을 수동으로 제어하려면 cmdlet에 대한 MaxKey 및 Marker 파라미터의 조합과 마지막으로 기록된 응답에 대한 IsTruncated/NextMarker 노트를 사용합니다. 다음 예제에서는 \$c 변수가 지정된 키 접두사 마커 시작 후 버킷에서 발견되는 다음 500개 객체에 대해 S3Object 인스턴스를 최대 500개까지 포함합니다.

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

## 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 구성](#)에서 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 [시작](#)을 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(예: EC2)를 찾아 선택한 다음 EC2 사용 사례(예: EC2)를 선택합니다.
4. 다음을 선택하고 애플리케이션에서 사용할 AWS 서비스에 대한 [적절한 정책](#)을 선택합니다.

### Warning

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

5. 다음(Next)을 선택합니다. 역할 이름, 설명 및 원하는 태그를 입력합니다.

태그에 대한 정보는 [IAM 사용 설명서](#)의 [AWS 리소스 태그](#)를 사용하여 액세스 제어에서 찾을 수 있습니다.

## 6. 역할 생성을 선택합니다.

[IAM 사용 설명서](#)의 [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 Center](#)과 같은 보안 인증 공급자를 통한 페더레이션을 사용하세요.

### 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 방법](#)을 참조하세요.  
모범 보안 사례는 [도구 인증 구성](#)에 설명된 대로 AWS IAM Identity Center를 사용하세요.

## 보안 인증에 대한 중요 경고 및 지침

### 보안 인증에 대한 경고

- 금지 사항. AWS 리소스에 액세스할 때는 계정의 루트 자격 증명을 사용해서는 안 됩니다. 이 자격 증명은 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 보안 인증은 일반 텍스트로 저장된다는 점에 유의하세요.

## 보안 인증 정보를 안전하게 관리하기 위한 추가 지침

AWS 보안 인증 정보를 안전하게 관리하는 방법에 대한 일반적인 설명은 [AWS 일반 참조의 AWS 보안 인증 정보 및 IAM 사용 설명서](#)의 [보안 모범 사례 및 사용 사례](#)를 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 역할](#)을 사용하세요.
- Amazon EC2 인스턴스에서 실행 중인 애플리케이션에 [IAM 역할](#)을 사용하세요.

## 주제

- [AWS 자격 증명 사용](#)
- [AWS Tools for PowerShell의 공유 자격 증명](#)

## AWS 자격 증명 사용

각 AWS Tools for PowerShell 명령은 AWS 자격 증명 세트를 포함해야 합니다. 이러한 자격 증명 세트는 해당하는 웹 서비스 요청을 암호로 서명하는 데 사용됩니다. 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 Center](#)과 같은 보안 인증 공급자를 통한 페더레이션을 사용하세요.

**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 방법](#)을 참조하세요.

모범 보안 사례는 [도구 인증 구성](#)에 설명된 대로 AWS IAM Identity Center를 사용하세요.

자격 증명에 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 PowerShell에서 관련 자격 증명을 검색합니다. AWS 보안 인증 정보를 안전하게 관리하는 방법에 대한 일반적인 설명은 Amazon Web Services 일반 참조에서 [AWS 액세스 키 관리를 위한 모범 사례](#)를 참조하세요.

**Note**

자격 증명을 얻어서 AWS Tools for PowerShell를 사용하려면 AWS 계정이 필요합니다. AWS 계정을 만들려면 AWS Account Management 계정 관리 참조 가이드에서 [시작하기: AWS를 처음 사용하시나요?](#)를 참조하세요.

**주제**

- [자격 증명 저장소 위치](#)
- [프로파일 관리](#)
- [자격 증명 지정](#)
- [자격 증명 검색 순서](#)
- [AWS Tools for PowerShell Core의 자격 증명 처리](#)

**자격 증명 저장소 위치**

AWS Tools for PowerShell에서는 두 가지 자격 증명 저장소 중 하나를 사용할 수 있습니다.

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

[AWS SDK for .NET](#) 및 [Toolkit for Visual Studio](#)에서도 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 SDK 및 AWS Command Line Interface에서도 자격 증명 파일을 사용할 수 있습니다. AWS 사용자 컨텍스트 외부에서 스크립트를 실행 중인 경우, 자격 증명을 포함하는 파일이 모든 사용자 계정 (로컬 시스템 및 사용자)에서 해당 자격 증명에 액세스할 수 있는 위치로 복사되도록 해야 합니다.

## 프로파일 관리

프로파일은 AWS Tools for PowerShell을 사용하여 서로 다른 자격 증명 세트를 참조할 수 있도록 해줍니다. AWS Tools for PowerShell cmdlet을 사용하여 AWS SDK 저장소의 프로파일을 관리할 수 있습니다. 또한 [Toolkit for Visual Studio](#)를 사용하거나, [AWS SDK for .NET](#)을(를) 사용하여 프로그래밍 방식으로 AWS SDK 저장소를 관리할 수도 있습니다. 자격 증명 파일의 프로파일을 관리하는 방법에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

### 새 프로파일 추가

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

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

- -AccessKey - 액세스 키 ID입니다.
- -SecretKey - 보안 키입니다.
- -StoreAs - 프로파일 이름으로, 고유해야 합니다. 기본 프로파일을 지정하려면 이름 default를 사용합니다.

### 프로파일 업데이트

AWS SDK 저장소는 수동으로 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-AWSCredential](#)은 이전 버전과의 호환성을 위해 계속 제공되지만, `Remove-AWSCredentialProfile`가 선호됩니다.

### 자격 증명 지정

자격 증명을 지정하는 방법에는 몇 가지가 있습니다. 기본 방법은 리터럴 자격 증명을 명령줄에 통합하는 대신 프로파일을 식별하는 것입니다. AWS Tools for PowerShell은 [자격 증명 검색 순서](#)에 설명된 검색 순서사용해 프로파일을 찾습니다.

Windows에서는 AWS SDK 저장소에 저장된 AWS 자격 증명이 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 프로파

일(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 AWS라는 이름의 프로파일에 저장되어 있는 경우, 모든 default SDK 및 관리 도구는 로컬 컴퓨터에서 자격 증명을 자동으로 찾아줍니다. 예를 들어 로컬 컴퓨터에 default라는 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 default가 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 중이거나 자격 증명 프로파일을 수동으로 설정하고자 하는 경우가 아니면 `Initialize-AWSDefaultConfiguration`을 실행하지 않는 것이 좋습니다. 이 경우 자격 증명 프로파일에는 자격 증명이 포함되어 있지 않다는 점에 유의하십시오. EC2 인스턴스에서 `Initialize-AWSDefaultConfiguration`을 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-AWSDefaultConfiguration`를 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 저장소에 default 프로파일 이름으로 포함되어 있습니다. 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 지역 지정하기](#) 섹션을 참조하세요. 다음 예제에서는 기본 프로파일 및 리전을 지정합니다.

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

기본적으로 AWS 공유 자격 증명 파일이 사용자의 홈 폴더(Windows의 `C:\Users\username\.aws` 또는 Linux의 `~/.aws`)에 있다고 가정합니다. 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 증명을 사용하여 실행되도록 하려면 `-ProfileLocation` 계정을 사용하지 않는 컨텍스트나 프로세스에서 스크립트를 실행할 때마다 AWS 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 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).

- default SDK 저장소의 AWS 프로파일.
- default 공유 자격 증명 파일의 AWS 프로파일.
- 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.

## AWS Tools for PowerShell Core의 자격 증명 처리

AWS Tools for PowerShell Core의 cmdlet은 AWS와 유사하게 AWS Tools for Windows PowerShell 액세스 키 및 보안 키나 실행 시 자격 증명 프로파일의 이름을 수락합니다. 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. 이 파일은 AWS Tools for PowerShell에서 자격 증명 프로파일을 검색하는 첫 번째 파일이자, AWS Tools for PowerShell에서 자격 증명 프로파일을 저장하는 파일이기도 합니다. AWS SDK for .NET 자격 증

명 저장소 파일에 대한 자세한 내용은 [AWS 자격 증명 구성](#)을 참조하세요. 현재 Tools for Windows PowerShell 모듈에서는 다른 파일이나 위치에 자격 증명 쓰기를 지원하지 않습니다.

두 모듈 모두 다른 AWS SDK 및 AWS에서 사용되는 AWS CLI 공유 자격 증명 파일에서 프로파일을 읽어올 수 있습니다. Windows에서 이 파일의 기본 위치는 C:\Users\\.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 API는 다른 플랫폼에서 사용할 수 없으므로 AWS Tools for PowerShell Core 모듈에서는 AWS 공유 자격 증명 파일만을 전적으로 사용하고 공유 자격 증명 파일에 새 자격 증명 프로파일을 쓸 수 있도록 지원합니다.

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

```
# 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

```

## AWS Tools for PowerShell의 공유 자격 증명

Tools for Windows PowerShell은 AWS CLI 및 다른 AWS SDK와 마찬가지로 AWS 공유 자격 증명 파일의 사용을 지원합니다. Tools for Windows PowerShell은 이제 .NET 자격 증명 파일과 AWS 공유 자격 증명 파일 모두에 대해 `basic`, `session` 및 `assume role` 자격 증명 프로파일의 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 Center](#)과 같은 보안 인증 공급자를 통한 페더레이션을 사용하세요.

### 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 방법](#)을 참조하세요.

모범 보안 사례는 [도구 인증 구성](#)에 설명된 대로 AWS IAM Identity Center를 사용하세요.

새 프로파일 유형과 AWS 공유 자격 증명 파일 액세스는 자격 증명 관련 cmdlet인 [Initialize-AWSDefaultConfiguration](#), [New-AWSCredential](#) 및 [Set-AWSCredential](#)에 추가된 다음 파라미터에서 지원됩니다. 서비스 cmdlet에서는 공통 파라미터인 `-ProfileName`을 추가하여 새 프로파일을 참조할 수 있습니다.

## AWS Tools for PowerShell에서 IAM 역할 사용

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

파라미터 이름	설명
<code>ExternalId</code>	역할에 필요할 경우 역할 수입 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 <a href="#">AWS 리소스에 대한 액세스 권한을 서드 파티에 부여할 때 외부 ID를 사용하는 방법</a> 을 참조하세요.
<code>MfaSerial</code>	역할에 필요할 경우 역할 수입 시 사용될 MFA 일련 번호입니다. 자세한 내용은 IAM 사용 설명서의 <a href="#">AWS에서 멀티 팩터 인증(MFA) 사용</a> 을 참조하세요.
<code>RoleArn</code>	역할 자격 증명 수입을 위해 수입할 역할의 ARN입니다. IAM 역할 생성 및 사용에 대한 자세한 내용은 IAM 사용 설명서에서 <a href="#">IAM 역할</a> 을 참조하세요.
<code>SourceProfile</code>	역할 자격 증명 수입에 사용될 소스 프로파일의 이름입니다. 이 프로파일에서 확인된 자격 증명

파라미터 이름	설명
	은 RoleArn 파라미터가 지정한 역할을 수입하는 데 사용됩니다.

### 역할 수입을 위한 프로파일 설정

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

첫 번째 명령은 역할 프로파일에서 참조하는 원본 프로파일을 생성합니다.. 두 번째 명령은 어떤 역할을 수입할 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 예제에 정의된 역할 프로파일을 사용하여 [Get-S3Bucket](#) cmdlet에 액세스합니다. AWS Tools for PowerShell 는 my\_source\_profile에서 자격 증명을 조회하고 이러한 자격 증명을 사용하여 사용자 대신 AssumeRole을 호출한 다음, 해당되는 임시 역할 자격 증명을 사용하여 Get-S3Bucket를 호출합니다.

```

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	-AccessKey
이들은 IAM 사용자에게 대한 장기 자격 증명입니다.	-SecretKey
세션:  이들은 <a href="#">Use-STSRole</a> cmdlet의 직접 호출 등을 통해 수동으로 검색하는 IAM 역할에 대한 단기 자격 증명입니다..	-AccessKey  -SecretKey  -SessionToken
역할:  이들은 사용자를 위해 AWS Tools for PowerShell이 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 자격 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. 프로파일을 찾을 수 없는 경우 AWS에서 ( <i>user's home</i>

프로파일 위치 값	프로파일 해결 동작 <i>directory</i> ) \.aws\credentials 공유 자격 증명 파일을 검색합니다.
AWS 공유 자격 증명 파일 형식으로 된 파일의 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\user\.aws\credentials -ProfileName
    basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\user\.aws\credentials

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

## 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\user\.aws\credentials
```

## 자격 증명 프로파일 제거

자격 증명 프로파일을 제거하려면 새 [Remove-AWSCredentialProfile](#) cmdlet을 실행합니다. [Clear-AWSCredential](#)은 더 이상 사용되지 않지만 이전 버전과의 호환성을 위해 여전히 제공되고 있습니다.

## 중요 정보

[Initialize-AWSDefaultConfiguration](#), [New-AWSCredential](#) 및 [Set-AWSCredential](#)만 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 PowerShell에서 AWS 서비스 작업

이 단원에서는 AWS 서비스에 액세스하기 위해 AWS Tools for PowerShell을 사용한 예를 제공합니다. 이러한 예제는 cmdlet을 사용하여 실제 AWS 작업을 수행하는 방법을 보여 주기 위한 것입니다. 이러한 예제에서는 Tools for PowerShell에서 제공하는 cmdlet을 사용합니다. 사용할 수 있는 cmdlet을 확인하려면 [AWS Tools for PowerShell Cmdlet Reference](#)를 참조하세요.

## PowerShell 파일 연결 인코딩

AWS Tools for PowerShell의 일부 cmdlet은 AWS에 있는 기존 파일이나 레코드를 편집합니다. Amazon Route 53를 위한 [ChangeResourceRecordSets](#) API를 호출하는 Edit-R53ResourceRecordSet이 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 Out-File로 파이프할 수 있습니다. Export-Csv나 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

기본 PowerShell 환경에서 AWS Tools for PowerShell의 유용성을 높이기 위해 AWS Tools for PowerShell cmdlet에서 반환되는 객체는 AWS SDK의 해당 API에서 일반적으로 반환되는 JSON 텍스트 객체가 아니라 .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. 이 동작은 설계에 따른 것이며, 관용적인 PowerShell과의 일관성을 높이기 위해 AWS Tools for PowerShell 경험을 지원합니다.

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 단원에서는 AWS Lambda Tools for PowerShell 모듈에 대한 간단한 개요를 제공하고 모듈을 설정하기 위한 절차를 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 대기열 생성
- 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 항목

- [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](#)
- [AWS Tools for Windows PowerShell에서의 CloudWatch](#)
- [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 콘솔](#)을 사용한 유사한 프로세스를 설명하는 [정적 웹 사이트 호스팅](#)에 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.

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)
- [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-example이라는 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 항목

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)
- [Put 버킷\(Amazon S3 서비스 참조\)](#)
- [Amazon S3용 AWS 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 객체로 업로드해야 합니다](#).

```
PS > Write-S3BucketWebsite -BucketName website-example -
WebsiteConfiguration_IndexDocumentSuffix index.html -WebsiteConfiguration_ErrorDocument
error.html
RequestId      : A1813E27995FFDDD
```

```

AmazonId2      : T7h1D0eLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xA1I
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}
Metadata       : {}
ResponseXml     :

```

## 참고 항목

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)
- [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. -File에 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 개요](#)를 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 항목

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)
- [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 cmdlet`을 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 cmdlet`을 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 항목

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)
- [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 mybucket -key myobject.txt -content "file content"

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

```

```

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

```

## Amazon EC2 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 섹션을 참조하십시오. [AWS Tools for Windows PowerShell 시작](#)

### 주제

- [키 페어 만들기](#)
- [Windows를 사용하여 보안 그룹 생성 PowerShell](#)
- [Windows PowerShell을 사용하여 Amazon Machine Images 찾기](#)
- [윈도우를 사용하여 Amazon EC2 인스턴스 시작 PowerShell](#)

### 키 페어 만들기

다음 New-EC2KeyPair 예제에서는 키 페어를 만들고 PowerShell 변수 \$myPSKeyPair에 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----
                   MIIIEogIBAAKCAQEAKK+ANYUS9c7niNjYfaCn6KYj/D0I6djnFoQE...
                   Mz6bt0xPcE7EMeH1wySUP8nouAS9xb1917+VkD74bN9KmNcPa/Mu...
                   Zyn4vVe0Q5il/MpkrRogHq0B0rigeTeV5Yc31v00RFFPu0Kz4kcm...
                   w3Jg8dKsWn0p10pX7V3sRC02KgJIbejQUvBFGi50QK9bm4tXBIeC...
                   daxKIAQMtDUdmBDrhR1/YMv8itFe5DiLLbq7Ga+FDcS85NstBa3h...
                   iuskGkcvGwkcFQkLmRHRoDpPb+OdFsZtjHZDpMVfMA9tT8EdbkEF...
                   3SrNeqZPsxJJIX0odb3CxLJpg75JU5kyWnb0+sDNVHoJiZCULCr0...
                   GG1LfEgB95KjGIk7zEv2Q7K6s+DHclrDeMZwa7KFNRZuCuX7jssC...
                   x098abxMr3o3TNU6p1ZYRJEQ0oJr0W+kC+/8SWb8NIwflTwhmJEy...
                   1BX9X8WFX/A8VLHrT1elrKmlkNECgYEAwltkV1p0JAFhz9p7ZFEv...
                   vvVsPaF0Ev9bk9pqhx269PB50x2KokwCagDMMaYvasWobuLmNu/1...
                   1mwRx7KTeQ7W1J30LgxHA1QNMkip9c4Tb3q9vVc3t/fPf8vwfJ8C...
                   63g6N6rk2FkHZX1E62BgbewUd3eZ0S05Ip4VUdvtGcuc8/qa+e5C...
                   KXgyt9n164pMv+VaXfXkZhdLAdY0Khc9TGB9++VMSG5TrD15YJId...
                   gYALEI7m1jJKpHWAES0hiemw5VmKyIZpzGstSJsFStERlAjiETDH...
                   YAtnI4J8dRyP9I7B0VOn3wNfIjk85gi1/00c+j8S65giLAFndWGR...
                   9R9wIkM5BMUCsRRcDy0yuwKBgEbk0nGGSD0ah4HkvrUkepIbUDTD...
                   AnEBM1cXI5UT7BfKInpUihZi59QhgdK/hk0SmWhlZGwikJ5VizBf...
                   drkBr/vTKVRMTi31VFB7KkIV1xJxC5E/BZ+YdZEpWoCZAoGAC/Cd...
                   TTld5N6opg0XAcQJwzqoGa9ZMwc5Q9f4bfRc67emkw0ZAAwSsvWR...
                   x302duuy7/smTwWwskEWRK5IrUxoMv/VVYaqdzc0ajwieNrbl1r7c...
```

```
-----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.

## Windows를 사용하여 보안 그룹 생성 PowerShell

를 AWS Tools for PowerShell 사용하여 보안 그룹을 만들고 구성할 수 있습니다. 보안 그룹을 생성할 때 해당 그룹이 EC2-Classic인지 EC2-VPC인지 여부를 지정합니다. 응답으로 보안 그룹의 ID가 반환됩니다.

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

주제

- [필수 조건](#)
- [EC2-Classic에 대한 보안 그룹 생성](#)
- [EC2-VPC에 대한 보안 그룹 생성](#)

### 필수 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

0.0.0.0/0을 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-Classic에 대한 보안 그룹 생성

### Warning

EC2-Classic은 2022년 8월 15일에 사용 중지될 예정입니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. [자세한 내용은 Amazon EC2 사용 설명서 또는 Amazon EC2 사용 설명서의 EC2-Classic에서 VPC로 마이그레이션을 참조하십시오.](#) 또는 블로그 게시물 [EC2-Classic 네트워킹은 사용 중지 중입니다 - 준비 방법은 다음과 같습니다](#)를 참조하세요.

다음 예제에서는 `New-EC2SecurityGroup` cmdlet을 사용하여 EC2-Classic에 대한 보안 그룹을 생성합니다.

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

```
sg-0a346530123456789
```

보안 그룹의 초기 구성을 보려면 `Get-EC2SecurityGroup` cmdlet을 사용합니다.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup
```

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

TCP 포트 22(SSH) 및 TCP 포트 3389에 대해 인바운드 트래픽을 허용하도록 보안 그룹을 구성하려면 `Grant-EC2SecurityGroupIngress` cmdlet을 사용합니다. 예를 들어 다음 스크립트는 단일 IP 주소인 203.0.113.25/32에서 들어오는 SSH 트래픽을 활성화하는 방법을 보여 줍니다.

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

```
ipPermissions.IpRanges = $cidrBlocks
Grant-EC2SecurityGroupIngress -GroupName myPSSecurityGroup -IpPermissions
$ipPermissions
```

보안 그룹이 업데이트되었는지 확인하려면 Get-EC2SecurityGroup cmdlet을 다시 실행합니다. EC2-Classic에 대한 아웃바운드 규칙은 지정할 수 없습니다.

```
PS > Get-EC2SecurityGroup -GroupNames myPSSecurityGroup
```

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

보안 그룹 규칙을 보려면 IpPermissions 속성을 사용합니다.

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

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

## EC2-VPC에 대한 보안 그룹 생성

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 203.0.113.25/32에서 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. 하지만 AWS에서는 새 AMI에 최신 업데이트 및 보안 개선을 제공하므로 AWS Windows AMI의 ID가 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, xvdc, xvdc...}
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

```

## 윈도우를 사용하여 Amazon EC2 인스턴스 시작 PowerShell

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](#)
- [Windows를 사용하여 보안 그룹 생성 PowerShell](#)
- [윈도우를 사용하여 아마존 머신 이미지 찾기 PowerShell](#)

### ⚠ Important

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

### 주제

- [EC2-Classic에서 인스턴스 시작](#)
- [VPC에서 인스턴스 시작](#)
- [VPC에서 스팟 인스턴스 시작](#)

## EC2-Classic에서 인스턴스 시작

### ⚠ Warning

EC2-Classic은 2022년 8월 15일에 사용 중지될 예정입니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. [자세한 내용은 Amazon EC2 사용 설명서 또는 Amazon EC2 사용 설명서의 EC2-Classic에서 VPC로 마이그레이션을 참조하십시오.](#) 또는 블로그 게시물 [EC2-Classic 네트워킹은 사용 중지 중입니다 - 준비 방법은 다음과 같습니다](#)를 참조하세요.

다음 명령은 단일 t1.micro 인스턴스를 생성 및 시작합니다.

```
PS > New-EC2Instance -ImageId ami-c49c0dac `
  -MinCount 1 `
  -MaxCount 1 `
  -KeyName myPSKeyPair `
  -SecurityGroups myPSSecurityGroup `
  -InstanceType t1.micro

ReservationId : r-b70a0ef1
OwnerId       : 123456789012
RequesterId   :
```

```

Groups           : {myPSSecurityGroup}
GroupName        : {myPSSecurityGroup}
Instances        : {}

```

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 Instances 속성을 표시하는 Get-EC2Instance을 호출합니다.

```

PS > $reservation = New-Object 'collections.generic.list[string]'
PS > $reservation.add("r-5caa4371")
PS > $filter_reservation = New-Object Amazon.EC2.Model.Filter -Property @{Name =
    "reservation-id"; Values = $reservation}
PS > (Get-EC2Instance -Filter $filter_reservation).Instances

```

```

AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken         :
EbsOptimized        : False
Hypervisor          : xen
IamInstanceProfile  :
ImageId             : ami-c49c0dac
InstanceId          : i-5203422c
InstanceLifecycle   :
InstanceType        : t1.micro
KernelId           :
KeyName             : myPSKeyPair
LaunchTime          : 12/2/2018 3:38:52 PM
Monitoring          : Amazon.EC2.Model.Monitoring
NetworkInterfaces   : {}
Placement           : Amazon.EC2.Model.Placement
Platform           : Windows
PrivateDnsName      :
PrivateIpAddress    : 10.25.1.11
ProductCodes        : {}
PublicDnsName       :
PublicIpAddress     : 198.51.100.245
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SecurityGroups      : {myPSSecurityGroup}

```

```

SourceDestCheck      : True
SpotInstanceRequestId :
SriovNetSupport      :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              :
Tags                  : {}
VirtualizationType    : hvm
VpcId                 :

```

## VPC에서 인스턴스 시작

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 `Instances` 속성을 표시하는 `Get-EC2Instance`을 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 사용해 PowerShell Core 6.0에서 AWS Lambda 함수를 개발할 수 있습니다. PowerShell 개발자는 Lambda를 사용하여 PowerShell 환경에서 AWS 리소스를 관리하고 자동 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 별도의 PowerShell용 AWS 모듈이며, AWS Tools for PowerShell에 포함되지 않으므로 AWSLambdaPSCore 모듈을 설치해도 AWS Tools for PowerShell은 설치되지 않습니다.

AWSLambdaPSCore 모듈을 설치한 후 사용 가능한 PowerShell cmdlet을 사용하거나 직접 개발하여 서버리스 함수를 작성할 수 있습니다. AWS Lambda Tools for PowerShell 모듈에는 PowerShell 기반 서버리스 애플리케이션용 프로젝트 템플릿과 프로젝트를 AWS에 게시하는 도구가 포함되어 있습니다.

AWSLambdaPSCore 모듈 지원은 Lambda를 지원하는 모든 리전에서 사용할 수 있습니다. 지원되는 리전에 대한 자세한 내용은 [AWS 리전 표](#)를 참조하세요.

### 필수 조건

AWSLambdaPSCore 모듈을 설치하고 사용하려면 먼저 다음 절차를 수행해야 합니다. 이러한 단계에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 [PowerShell 개발 환경 설정](#)을 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 웹사이트](#)의 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 센터](#)에서 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 모델](#)을 참조하세요.

## 참고 항목

- [AWS 개발자 블로그에 PowerShell Core를 위한 Lambda 지원 발표](#)
- [PowerShell 갤러리의 AWSLambdaPSCore 모듈](#)
- [PowerShell 개발 환경 설정](#)
- [GitHub의 AWS Lambda Tools for PowerShell](#)
- [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 주제 생성
- 대기열에 메시지를 보낼 수 있도록 SNS 주제에 권한을 부여합니다.
- SNS 주제에 대한 대기열을 구독합니다.

- SNS 주제에 게시하고 SQS 대기열에서 메시지를 읽을 수 있도록 IAM 사용자 또는 AWS 계정에 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 myQueue에 대한 대기열 myTopic를 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 sns:Publish 주제에 대한 myTopic 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 sqs:ReceiveMessage에 대한 sqs:DeleteMessage 및 myQueue 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" :
    "11E17A2+X0uJZnw3TlgcXz4C4KPLXZxbxoEMIirelh13u/oxkWmz5+9tJKFMns1Z0qQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1b03hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
vyuGkJReV710JWPHnCN
+qTE21Id2RPkF0eGtLGawTsSPTWEvJdDbL1f7E0zZ0q1niXTUtpsZ8Swx01X3Q06u9i9qBFt0ekJFZNJp6Avu05hIk1b4yoc
y0a8Y191Wp7a7EoWaBn0zhCESe7o
kZC6ncBJWphX7KCGVYD0qhVf/5VDgBuv9w8T+higJyvri3WbaSvg==",
  "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
```

```

MD50fMessageAttributes :
MessageAttributes      : {}
MessageId              : 728180b6-f62b-49d5-b4d3-3824bb2e77f4
ReceiptHandle         :
  AQEB2vvk1e5c0KFjeIWJticabkc664yuDEjhucnIOqdVUmie7bX7GiJb17F0enABUgaI2XjEcNPxixhVc/
wfsAJZLNHn18S1bQa0R/kD+Saq40Ivfj8x3M40h1yM1cVKpYmhAzsYrAwAD5g5FvxNBD6zs
  +HmXdkax2Wd+9AxrH1QZV5ur1MoByKWwBDbSqoYJTJquCc10gWIak/sBx/
daBRMTiVQ4GHsrQWMVHtNC14q7Jy/0L2dkmb4dzJfJq0VbFSX1G+u/lrSLpgae+Dfux646y8yFiPFzY4ua4mCF/
SVUn63Spy
  sHN12776axknhg3j9K/Xwj54DixdsegrKoLx+ctI
+0jzAetBR66Q1VhIoJAq7s0a2Msey0eM/Jjucg6Sr9VUnTWVhV8ErXmotoiEg==

```

## AWS Tools for Windows PowerShell에서의 CloudWatch

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 콘솔](#)로 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

```

다음을 참조하세요.

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

## 참고 항목

- [AWS Tools for PowerShell에서 AWS 서비스 작업](#)

- [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 AWSconfig 파일
5. 환경 변수
6. Amazon EC2 인스턴스 메타데이터 서비스(활성화된 경우)

## PowerShell 코드 예제를 위한 도구

이 항목의 코드 예제는 AWS Tools for PowerShell with를 사용하는 방법을 보여줍니다 AWS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

교차 서비스 예시는 여러 AWS 서비스전반에서 작동하는 샘플 애플리케이션입니다.

예제

- [다음에 대한 도구를 사용한 작업 및 시나리오 PowerShell](#)

## 다음에 대한 도구를 사용한 작업 및 시나리오 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS 서비스.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

서비스

- [다음과 같은 도구를 사용한 ACM 예제 PowerShell](#)
- [AppStream 도구를 사용한 2.0 예제 PowerShell](#)
- [다음과 같은 도구를 사용한 Aurora 예제 PowerShell](#)
- [다음 툴을 사용한 Auto Scaling 예제 PowerShell](#)
- [AWS Budgets 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [AWS Cloud9 다음과 같은 도구를 사용하는 예 PowerShell](#)

- [AWS CloudFormation 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [CloudFront 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [CloudTrail 도구를 사용하는 예: PowerShell](#)
- [CloudWatch 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [CodeCommit 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [CodeDeploy 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [CodePipeline 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [다음 도구를 사용한 Amazon Cognito 자격 증명 예제 PowerShell](#)
- [AWS Config 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [다음 도구를 사용한 Device Farm 예제 PowerShell](#)
- [AWS Directory Service 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [AWS DMS 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [다음과 같은 도구를 사용한 DynamoDB 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon EC2 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon ECR 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon ECS 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon EFS 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon EKS 예제 PowerShell](#)
- [Elastic Load Balancing - 다음 도구를 사용한 버전 1 예제 PowerShell](#)
- [Elastic Load Balancing - 다음 도구를 사용한 버전 2 예제 PowerShell](#)
- [다음과 같은 도구를 사용하는 Amazon FSx 예제 PowerShell](#)
- [AWS Glue 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [AWS Health 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [다음과 같은 도구를 사용한 IAM 예제 PowerShell](#)
- [다음 툴을 사용한 Kinesis 예제 PowerShell](#)
- [에 대한 도구를 사용한 Lambda 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon ML 예제 PowerShell](#)
- [다음과 같은 도구를 사용한 Macie 예제 PowerShell](#)
- [AWS OpsWorks 다음과 같은 도구를 사용하는 예 PowerShell](#)

- [AWS 가격표 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [예 대한 도구를 사용한 Resource Groups 예제 PowerShell](#)
- [도구를 사용한 리소스 그룹 태깅 API 예제 PowerShell](#)
- [다음과 같은 도구를 사용한 Route 53 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon S3 예제 PowerShell](#)
- [다음과 같은 도구를 사용한 S3 Glacier 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon SES 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon SNS 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon SQS 예제 PowerShell](#)
- [AWS STS 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [AWS Support 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [다음 툴을 사용한 Systems Manager 예제 PowerShell](#)
- [다음 도구를 사용한 Amazon Translate 예제 PowerShell](#)
- [AWS WAFV2 다음과 같은 도구를 사용하는 예 PowerShell](#)
- [WorkSpaces 다음과 같은 도구를 사용하는 예 PowerShell](#)

## 다음과 같은 도구를 사용한 ACM 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with ACM을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Get-ACMCertificate

다음 코드 예시에서는 Get-ACMCertificate을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 인증서의 ARN을 사용하여 인증서 및 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 참조를 참조하십시오 [GetCertificate](#).

### Get-ACMCertificateDetail

다음 코드 예시에서는 Get-ACMCertificateDetail을 사용하는 방법을 보여 줍니다.

예 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 참조를 참조하십시오 [DescribeCertificate](#).

## Get-ACMCertificateList

다음 코드 예시에서는 Get-ACMCertificateList을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 모든 인증서 ARN 목록과 각 인증서 ARN의 도메인 이름을 검색합니다. cmdlet은 자동으로 페이지를 매겨 모든 ARN을 검색합니다. 페이지 매김을 수동으로 제어하려면 - MaxItem 매개 변수를 사용하여 각 서비스 호출에 대해 반환되는 인증서 ARN 수를 제어하고 - NextToken 매개 변수를 사용하여 각 호출의 시작 지점을 지정합니다.

```
Get-ACMCertificateList
```

출력:

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

예 2: 인증서 상태가 제공된 상태와 일치하는 모든 인증서 ARN 목록을 검색합니다.

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

예 3: 이 예제는 us-east-1 지역에서 키 유형이 RSA\_2048이고 확장 키 사용 또는 용도가 CODE\_SIGNING인 모든 인증서 목록을 반환합니다. 이러한 필터링 매개변수의 값은 필터 API 참조 항목인 [https://docs.aws.amazon.com/acm/latest/APIReference/API\\_Filters.html](https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html) 에서 찾을 수 있습니다. ListCertificates

```
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

```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListCertificates](#).

**New-ACMCertificate**

다음 코드 예시에서는 New-ACMCertificate을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 새 인증서를 생성합니다. 이 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
```

예 2: 새 인증서를 생성합니다. 이 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 참조를 참조하십시오 [RequestCertificate](#).

## Remove-ACMCertificate

다음 코드 예시에서는 Remove-ACMCertificate을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 제공된 ARN으로 식별되는 인증서 및 관련 개인 키를 삭제합니다. 계속하기 전에 cmdlet에서 확인 메시지를 표시합니다. 확인을 거부하려면 -Force 스위치를 추가하십시오.

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

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [DeleteCertificate](#) AWS Tools for PowerShell

## Send-ACMValidationEmail

다음 코드 예시에서는 Send-ACMValidationEmail을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 'www.example.com'의 도메인 소유권을 확인하기 위한 이메일을 보내도록 요청합니다. 셀의 ConfirmPreference \$가 'Medium' 이하로 설정된 경우 cmdlet은 계속 진행하기 전에 확인 메시지를 표시합니다. -Force 스위치를 추가하여 확인 메시지를 표시하지 않도록 합니다.

```
$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 Cmdlet 참조를 참조하십시오 [ResendValidationEmail](#).AWS Tools for PowerShell

## AppStream 도구를 사용한 2.0 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with AppStream 2.0을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Add-APSResourceTag

다음 코드 예시에서는 Add-APSResourceTag을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 리소스에 리소스 태그를 추가합니다. AppStream

```
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 참조를 참조하십시오 [TagResource](#).

## Copy-APSIImage

다음 코드 예시에서는 Copy-APSIImage을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 이미지를 다른 지역으로 복사합니다.

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

출력:

```
TestImageCopy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CopyImage](#).

## Disable-APSUser

다음 코드 예시에서는 Disable-APSUser을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 USERPOOL에서 사용자를 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisableUser](#).

## Enable-APSUser

다음 코드 예시에서는 Enable-APSUser을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 USERPOOL에서 비활성화된 사용자를 활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableUser](#).

## Get-APSAssociatedFleetList

다음 코드 예시에서는 Get-APSAssociatedFleetList을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 스택과 연결된 플릿을 표시합니다.

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

출력:

```
PowershellFleet
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAssociatedFleets](#).

## Get-APSAssociatedStackList

다음 코드 예시에서는 Get-APSAssociatedStackList을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 플릿과 관련된 스택을 표시합니다.

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

출력:

```
PowershellStack
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAssociatedStacks](#).

## Get-APSDirectoryConfigList

다음 코드 예시에서는 Get-APSDirectoryConfigList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 에서 만든 디렉터리 구성을 보여 줍니다. AppStream

예 1: 이 샘플은 에서 만든 디렉터리 구성을 보여 줍니다. AppStream

```
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 참조를 참조하십시오 [DescribeDirectoryConfigs](#).

## Get-APSFleetList

다음 코드 예시에서는 Get-APSFleetList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 플릿의 세부 정보를 표시합니다.

예 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 참조를 참조하십시오 [DescribeFleets](#).

## Get-APSIImageBuilderList

다음 코드 예시에서는 Get-APSIImageBuilderList을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플에는 다음과 같은 세부 정보가 표시됩니다. ImageBuilder

```
Get-APSIImageBuilderList -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 참조를 참조하십시오 [DescribeImageBuilders](#).

## Get-APSIImageList

다음 코드 예시에서는 Get-APSIImageList을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 비공개 AppStream 이미지를 표시합니다.

```
Get-APSIImageList -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 참조를 참조하십시오 [DescribeImages](#).

## Get-APSIImagePermission

다음 코드 예시에서는 Get-APSIImagePermission을 사용하는 방법을 보여 줍니다.

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

```
Get-APSIImagePermission -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 참조를 참조하십시오 [DescribeImagePermissions](#).

## Get-APSSessionList

다음 코드 예시에서는 Get-APSSessionList를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DescribeSessions](#).

## Get-APSSStackList

다음 코드 예시에서는 Get-APSSStackList를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 AppStream 스택 목록을 표시합니다.

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

출력:

DisplayName	Arn
-----	---
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 참조를 참조하십시오 [DescribeStacks](#).

## Get-APSTagsForResourceList

다음 코드 예시에서는 Get-APSTagsForResourceList을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 AppStream 리소스에 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 참조를 참조하십시오 [ListTagsForResource](#).

## Get-APSUsageReportSubscription

다음 코드 예시에서는 Get-APSUsageReportSubscription을 사용하는 방법을 보여 줍니다.

예 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 참조를 참조하십시오 [DescribeUsageReportSubscriptions](#).

## Get-APUser

다음 코드 예시에서는 Get-APUser을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 활성화된 상태의 사용자 목록을 표시합니다.

```
Get-APUser -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 참조를 참조하십시오 [DescribeUsers](#).

## Get-APUserStackAssociation

다음 코드 예시에서는 Get-APUserStackAssociation을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DescribeUserStackAssociations](#).

## New-APSDirectoryConfig

다음 코드 예시에서는 New-APSDirectoryConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 디렉터리 구성을 다음과 같이 생성합니다. AppStream

```
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
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 참조를 참조하십시오 [CreateDirectoryConfig](#).

## New-APSFleet

다음 코드 예시에서는 New-APSFleet을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 새 AppStream 플릿을 생성합니다.

예 1: 이 샘플은 새 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 참조를 참조하십시오 [CreateFleet](#).

## New-APSIImageBuilder

다음 코드 예시에서는 New-APSIImageBuilder을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 Image Builder를 에서 생성합니다. AppStream

예 1: 이 샘플은 Image Builder를 에서 생성합니다. AppStream

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

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 참조를 참조하십시오 [CreateImageBuilder](#).

## New-APSIImageBuilderStreamingURL

다음 코드 예시에서는 New-APSIImageBuilderStreamingURL을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 Image Builder를 에서 생성합니다. AppStream

예 1: 이 샘플은 유효 기간이 2시간인 ImageBuilder 스트리밍 URL을 생성합니다.

```
New-APSIImageBuilderStreamingURL -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=eyJ0eXB1IjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRjZCI6IjM5MzQwM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [CreateImageBuilderStreamingURL](#) 을 참조하십시오.

**New-APSSStack**

다음 코드 예시에서는 New-APSSStack을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 새 AppStream 스택을 생성합니다.

```
New-APSSStack -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 참조를 참조하십시오 [CreateStack](#).

## New-APSSstreamingURL

다음 코드 예시에서는 New-APSSstreamingURL을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 샘플은 스택의 스트리밍 URL을 생성합니다.

```
New-APSSstreamingURL -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=eyJ0eXB1IjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJCiI6IjM5M...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [CreateStreamingURL](#)을 참조하십시오.

## New-APSUsageReportSubscription

다음 코드 예시에서는 New-APSUsageReportSubscription을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 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 참조를 참조하십시오 [CreateUsageReportSubscription](#).

## New-APSUser

다음 코드 예시에서는 New-APSUser을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 USERPOOL에 대한 도구 PowerShell

예 1: 이 샘플은 USERPOOL에서 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 참조를 참조하십시오 [CreateUser](#).

## Register-APSFleet

다음 코드 예시에서는 Register-APSFleet을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 폴릿에 대한 도구 PowerShell

예 1: 이 샘플은 폴릿을 스택에 등록합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AssociateFleet](#).

## Register-APSUserStackBatch

다음 코드 예시에서는 Register-APSUserStackBatch을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 USERPOOL에 대한 도구 PowerShell

예 1: 이 샘플은 USERPOOL의 사용자에게 스택을 할당합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [BatchAssociateUserStack](#).

## Remove-APSDirectoryConfig

다음 코드 예시에서는 Remove-APSDirectoryConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 샘플은 AppStream 디렉터리 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 참조를 참조하십시오 [DeleteDirectoryConfig](#).

## Remove-APSFleet

다음 코드 예시에서는 Remove-APSFleet을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 샘플은 플릿을 제거하고 삭제합니다. AppStream

```
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 참조를 참조하십시오 [DeleteFleet](#).

**Remove-APSIImage**

다음 코드 예시에서는 Remove-APSIImage을 사용하는 방법을 보여 줍니다.

**에 대한 도구 PowerShell**

예 1: 이 샘플은 이미지를 삭제합니다.

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

**출력:**

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

```
Applications           : {}
AppstreamAgentVersion  : LATEST
Arn                    : arn:aws:appstream:us-west-2:123456789012:image/
TestImage
BaseImageArn           :
CreatedTime            : 12/27/2019 1:34:10 PM
Description            :
DisplayName            : TestImage
ImageBuilderName       :
ImageBuilderSupported  : True
ImagePermissions       :
Name                   : TestImage
Platform               : WINDOWS
```

```
PublicBaseImageReleasedDate : 6/12/2018 12:00:00 AM
State                        : AVAILABLE
StateChangeReason           :
Visibility                   : PRIVATE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteImage](#).

## Remove-APSIImageBuilder

다음 코드 예시에서는 Remove-APSIImageBuilder을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 다음을 삭제합니다. ImageBuilder

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

출력:

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

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

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteImageBuilder](#).

## Remove-APSIImagePermission

다음 코드 예시에서는 Remove-APSIImagePermission을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 이미지의 권한을 제거합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteImagePermissions](#).

## Remove-APSResourceTag

다음 코드 예시에서는 Remove-APSResourceTag을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 리소스에서 AppStream 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 참조를 참조하십시오 [UntagResource](#).

**Remove-APSStack**

다음 코드 예시에서는 Remove-APSStack을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DeleteStack](#).

**Remove-APSUsageReportSubscription**

다음 코드 예시에서는 Remove-APSUsageReportSubscription을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DeleteUsageReportSubscription](#).

**Remove-APSUser**

다음 코드 예시에서는 Remove-APSUser을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DeleteUser](#).

**Revoke-APSSession**

다음 코드 예시에서는 Revoke-APSSession을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 세션을 플릿으로 취소합니다. AppStream

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ExpireSession](#).

## Start-APSFleet

다음 코드 예시에서는 Start-APSFleet을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 플릿을 시작합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StartFleet](#).

## Start-APSIImageBuilder

다음 코드 예시에서는 Start-APSIImageBuilder을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 다음을 시작합니다. ImageBuilder

```
Start-APSIImageBuilder -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 참조를 참조하십시오 [StartImageBuilder](#).

## Stop-APSFleet

다음 코드 예시에서는 Stop-APSFleet을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 플릿을 정지시킵니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StopFleet](#).

## Stop-APSIImageBuilder

다음 코드 예시에서는 Stop-APSIImageBuilder을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 다음을 중지합니다. ImageBuilder

```
Stop-APSIImageBuilder -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 참조를 참조하십시오 [StopImageBuilder](#).

## Unregister-APSFleet

다음 코드 예시에서는 Unregister-APSFleet을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 스택에서 플릿을 등록 취소합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisassociateFleet](#).

## Unregister-APSUserStackBatch

다음 코드 예시에서는 Unregister-APSUserStackBatch을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오 [BatchDisassociateUserStack](#).

## Update-APSDirectoryConfig

다음 코드 예시에서는 Update-APSDirectoryConfig을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 에서 만든 디렉터리 구성을 업데이트합니다. AppStream

```
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 참조를 참조하십시오 [UpdateDirectoryConfig](#).

## Update-APSFleet

다음 코드 예시에서는 Update-APSFleet을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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 참조를 참조하십시오 [UpdateFleet](#).

## Update-APSIImagePermission

다음 코드 예시에서는 Update-APSIImagePermission을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 다른 계정과 AppStream 이미지를 공유합니다.

```

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

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateImagePermissions](#).

## Update-APSSStack

다음 코드 예시에서는 Update-APSSStack을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예 1: 이 샘플은 스택의 애플리케이션 설정 지속성 및 홈 폴더를 업데이트 (활성화) 합니다.

```
Update-APSSStack -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 참조를 참조하십시오 [UpdateStack](#).

## 다음과 같은 도구를 사용한 Aurora 예제 PowerShell

다음 코드 예제는 Aurora와 AWS Tools for PowerShell 함께 를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

## Get-RDSOrderableDBInstanceOption

다음 코드 예시에서는 Get-RDSOrderableDBInstanceOption을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

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
```

- API에 대한 세부 정보는 [AWS Tools for PowerShell Cmdlet InstanceOptions 참조의 DescribeOrderableDB](#)를 참조하십시오.

## 다음 툴을 사용한 Auto Scaling 예제 PowerShell

다음 코드 예제는 Auto Scaling과 AWS Tools for PowerShell 함께 를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Add-ASLoadBalancer

다음 코드 예시에서는 Add-ASLoadBalancer를 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서를 지정된 Auto Scaling 그룹에 연결합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AttachLoadBalancers](#).

### Complete-ASLifecycleAction

다음 코드 예시에서는 Complete-ASLifecycleAction을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [CompleteLifecycleAction](#).

## Disable-ASMetricsCollection

다음 코드 예시에서는 Disable-ASMetricsCollection을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 지표의 모니터링을 비활성화합니다.

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

예 2: 이 예에서는 지정된 Auto Scaling 그룹에 대한 모든 지표에 대한 모니터링을 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisableMetricsCollection](#).

## Dismount-ASInstance

다음 코드 예시에서는 Dismount-ASInstance을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리하고 Auto Scaling에서 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     :

```

예 2: 이 예에서는 원하는 용량을 줄이지 않고 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리합니다. Auto Scaling은 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 참조를 참조하십시오 [DetachInstances](#).

## Dismount-ASLoadBalancer

다음 코드 예시에서는 Dismount-ASLoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 Auto Scaling 그룹에서 지정된 로드 밸런서를 분리합니다.

```

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

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachLoadBalancers](#).

## Enable-ASMetricsCollection

다음 코드 예시에서는 Enable-ASMetricsCollection을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 지표를 모니터링할 수 있습니다.

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

예 2: 이 예에서는 지정된 Auto Scaling 그룹의 모든 지표를 모니터링할 수 있습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableMetricsCollection](#).

## Enter-ASStandby

다음 코드 예시에서는 Enter-ASStandby을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 인스턴스를 대기 모드로 전환하고 원하는 용량을 줄여서 Auto Scaling에서 대체 인스턴스를 시작하지 않도록 합니다.

```
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  :

```

예 2: 이 예제는 원하는 용량을 줄이지 않고 지정된 인스턴스를 대기 모드로 전환합니다. Auto Scaling은 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 참조를 참조하십시오 [EnterStandby](#).

## Exit-ASStandby

다음 코드 예시에서는 Exit-ASStandby을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 인스턴스를 대기 모드 밖으로 이동합니다.

```

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 참조를 참조하십시오 [ExitStandby](#).

## Get-ASAccountLimit

다음 코드 예시에서는 Get-ASAccountLimit을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 AWS 계정의 Auto Scaling 리소스 한도를 설명합니다.

```
Get-ASAccountLimit
```

출력:

```

MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAccountLimits](#).

## Get-ASAdjustmentType

다음 코드 예시에서는 Get-ASAdjustmentType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제에서는 Auto Scaling에서 지원하는 조정 유형을 설명합니다.

```
Get-ASAdjustmentType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAdjustmentTypes](#).

## Get-ASAutoScalingGroup

다음 코드 예시에서는 Get-ASAutoScalingGroup을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예는 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
```

예 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-1c}
LaunchConfigurationName   : my-1c
LoadBalancerNames         : {}
MaxSize                    : 0
MinSize                    : 0
PlacementGroup            :
Status                    :
SuspendedProcesses        : {}
Tags                       : {}
TerminationPolicies       : {Default}
VPCZoneIdentifier         : subnet-e4f33493,subnet-5264e837

```

예 3: 이 예제에서는 지정된 두 개의 Auto Scaling 그룹을 설명합니다.

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

예 4: 이 예제에서는 지정된 Auto Scaling 그룹의 Auto Scaling 인스턴스를 설명합니다.

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

예 5: 이 예제에서는 모든 Auto Scaling 그룹을 설명합니다.

```
Get-ASAutoScalingGroup
```

예 6: 이 예제에서는 모든 Auto Scaling 그룹을 10개씩 배치로 설명합니다.

```

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

```

예 7: 이 예제에서는 지정된 Auto Scaling 그룹에 LaunchTemplate 대해 설명합니다. 이 예에서는 “인스턴스 구매 옵션”이 “템플릿 시작 준수”로 설정되어 있다고 가정합니다. 이 옵션이 “구매 옵션 및 인스턴스 유형 결합”으로 설정된 경우 “MixedInstancesPolicy를 사용하여 액세스할 LaunchTemplate 수 있습니다. LaunchTemplate”속성.

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

출력:

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAutoScalingGroups](#).

## Get-ASAutoScalingInstance

다음 코드 예시에서는 Get-ASAutoScalingInstance을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제에서는 Auto Scaling 인스턴스의 ID를 나열합니다.

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

출력:

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

예 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

```

예 3: 이 예제에서는 지정된 두 개의 Auto Scaling 인스턴스를 설명합니다.

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

예 4: 이 예제에서는 지정된 Auto Scaling 그룹의 Auto Scaling 인스턴스를 설명합니다.

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

예 5: 이 예제에서는 모든 Auto Scaling 인스턴스를 설명합니다.

```
Get-ASAutoScalingInstance
```

예 6: 이 예제에서는 모든 Auto Scaling 인스턴스를 10개씩 배치로 설명합니다.

```

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

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAutoScalingInstances](#).

## Get-ASAutoScalingNotificationType

다음 코드 예시에서는 Get-ASAutoScalingNotificationType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeAutoScalingNotificationTypes](#).

**Get-ASLaunchConfiguration**

다음 코드 예시에서는 Get-ASLaunchConfiguration을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 시작 구성 이름이 나열되어 있습니다.

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

**출력:**

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

예 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                      :

```

예 3: 이 예에서는 지정된 두 개의 시작 구성을 설명합니다.

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

예 4: 이 예제에서는 모든 시작 구성을 설명합니다.

```
Get-ASLaunchConfiguration
```

예 5: 이 예에서는 모든 시작 구성을 10개씩 배치로 설명합니다.

```

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

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeLaunchConfigurations](#).

## Get-ASLifecycleHook

다음 코드 예시에서는 Get-ASLifecycleHook을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제에서는 지정된 라이프사이클 후크를 설명합니다.

예 1: 이 예제에서는 지정된 라이프사이클 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
```

예 2: 이 예제에서는 지정된 Auto Scaling 그룹의 모든 라이프사이클 후크를 설명합니다.

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

예 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 라이프사이클 후크를 설명합니다.

```
Get-ASLifecycleHook
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeLifecycleHooks](#).

## Get-ASLifecycleHookType

다음 코드 예시에서는 Get-ASLifecycleHookType을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제에서는 Auto Scaling에서 지원하는 라이프사이클 후크 유형을 나열합니다.

예 1: 이 예제에서는 Auto Scaling에서 지원하는 라이프사이클 후크 유형을 나열합니다.

```
Get-ASLifecycleHookType
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeLifecycleHookTypes](#).

## Get-ASLoadBalancer

다음 코드 예시에서는 Get-ASLoadBalancer을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 Auto Scaling 그룹의 로드 밸런서를 설명합니다.

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

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeLoadBalancers](#).

## Get-ASMetricCollectionType

다음 코드 예시에서는 Get-ASMetricCollectionType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 Auto Scaling에서 지원하는 지표 수집 유형을 나열합니다.

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

출력:

Metric
-----
GroupMinSize

```

GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances

```

예 2: 이 예제에서는 해당하는 세부 수준을 나열합니다.

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

출력:

```

Granularity
-----
1Minute

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMetricCollectionTypes](#).

## Get-ASNotificationConfiguration

다음 코드 예시에서는 Get-ASNotificationConfiguration을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹과 관련된 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

```

예 2: 이 예에서는 모든 Auto Scaling 그룹과 관련된 알림 작업을 설명합니다.

```
Get-ASNotificationConfiguration
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeNotificationConfigurations](#).

## Get-ASPolicy

다음 코드 예시에서는 Get-ASPolicy을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 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     : {}
```

예 2: 이 예에서는 지정된 Auto Scaling 그룹에 지정된 정책을 설명합니다.

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

예 3: 이 예에서는 모든 Auto Scaling 그룹에 대한 모든 정책을 설명합니다.

```
Get-ASPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribePolicies](#).

## Get-ASScalingActivity

다음 코드 예시에서는 Get-ASScalingActivity을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

- 예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대한 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     :

```

예 2: 이 예제에서는 지정된 조정 활동을 설명합니다.

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

예 3: 이 예에서는 모든 Auto Scaling 그룹에 대한 지난 6주 동안의 조정 활동을 설명합니다.

```
Get-ASScalingActivity
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeScalingActivities](#).

## Get-ASScalingProcessType

다음 코드 예시에서는 Get-ASScalingProcessType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 Auto Scaling에서 지원하는 프로세스 유형을 나열합니다.

```
Get-ASScalingProcessType
```

출력:

```

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

```

## Terminate

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeScalingProcessTypes](#).

## Get-ASScheduledAction

다음 코드 예시에서는 Get-ASScheduledAction을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹의 스케줄링된 조정 작업을 설명합니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹의 스케줄링된 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
```

예 2: 이 예제에서는 지정된 스케줄링된 조정 작업을 설명합니다.

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

예 3: 이 예제에서는 지정된 시간까지 시작되는 스케줄링된 조정 작업을 설명합니다.

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

예 4: 이 예제에서는 지정된 시간까지 종료되는 스케줄링된 조정 작업을 설명합니다.

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

예 5: 이 예에서는 모든 Auto Scaling 그룹의 스케줄링된 조정 작업을 설명합니다.

```
Get-ASScheduledAction
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeScheduledActions](#).

## Get-ASTag

다음 코드 예시에서는 Get-ASTag을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제에서는 키 값이 'MyTag' 또는 'MyTag2'인 태그를 설명합니다. 가능한 필터 이름 값은 ", '키', auto-scaling-group '값', ""입니다. propagate-at-launch 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
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 Filter 매개 변수에 대한 필터를 만들어야 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 )
```

예 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 태그를 설명합니다.

```
Get-ASTag
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTags](#).

## Get-ASTerminationPolicyType

다음 코드 예시에서는 Get-ASTerminationPolicyType을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 Auto Scaling에서 지원하는 종료 정책을 나열합니다.

```
Get-ASTerminationPolicyType
```

출력:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTerminationPolicyTypes](#).

## Mount-ASInstance

다음 코드 예시에서는 Mount-ASInstance을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 인스턴스를 지정된 Auto Scaling 그룹에 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 참조를 참조하십시오 [AttachInstances](#).

## New-ASAutoScalingGroup

다음 코드 예시에서는 New-ASAutoScalingGroup을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 이름과 속성을 사용하여 Auto Scaling 그룹을 생성합니다. 기본 권장 용량은 최소 크기입니다. 따라서 이 Auto Scaling 그룹은 지정된 두 가용 영역 각각에서 하나씩 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 참조를 참조하십시오 [CreateAutoScalingGroup](#).

## New-ASLaunchConfiguration

다음 코드 예시에서는 New-ASLaunchConfiguration을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 'my-lc'라는 이름의 시작 구성을 생성합니다. 이 시작 구성을 사용하는 Auto Scaling 그룹에서 시작하는 EC2 인스턴스는 지정된 인스턴스 유형, AMI, 보안 그룹 및 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 참조를 참조하십시오 [CreateLaunchConfiguration](#).

## Remove-ASAutoScalingGroup

다음 코드 예시에서는 Remove-ASAutoScalingGroup을 사용하는 방법을 보여 줍니다.

## 예 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"):
```

예 2: Force 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

예제 3: 이 예제에서는 지정된 Auto Scaling 그룹을 삭제하고 해당 그룹에 포함된 실행 중인 모든 인스턴스를 종료합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteAutoScalingGroup](#).

## Remove-ASLaunchConfiguration

다음 코드 예시에서는 Remove-ASLaunchConfiguration을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예에서는 지정된 시작 구성이 Auto Scaling 그룹에 연결되어 있지 않은 경우 해당 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"):
```

예 2: Force 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteLaunchConfiguration](#).

## Remove-ASLifecycleHook

다음 코드 예시에서는 Remove-ASLifecycleHook을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 라이프사이클 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"):
```

예 2: Force 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteLifecycleHook](#).

## Remove-ASNotificationConfiguration

다음 코드 예시에서는 Remove-ASNotificationConfiguration을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 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"):
```

예 2: Force 매개 변수를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteNotificationConfiguration](#).

## Remove-ASPolicy

다음 코드 예시에서는 Remove-ASPolicy을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 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"):
```

예 2: Force 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePolicy](#).

## Remove-ASScheduledAction

다음 코드 예시에서는 Remove-ASScheduledAction을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 스케줄링된 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"):
```

예 2: Force 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteScheduledAction](#).

## Remove-ASTag

다음 코드 예시에서는 Remove-ASTag을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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"):
```

예 2: Force 매개 변수를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

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

예 3: Powershell 버전 2에서는 New-Object를 사용하여 태그 매개 변수에 대한 태그를 만들어야 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteTags](#).AWS Tools for PowerShell

## Resume-ASProcess

다음 코드 예시에서는 Resume-ASProcess을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 재개합니다.

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

예 2: 이 예에서는 지정된 Auto Scaling 그룹에 대해 일시 중단된 모든 Auto Scaling 프로세스를 재개합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ResumeProcesses](#).

## Set-ASDesiredCapacity

다음 코드 예시에서는 Set-ASDesiredCapacity을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹의 크기를 설정합니다.

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

예 2: 이 예제에서는 지정된 Auto Scaling 그룹의 크기를 설정하고 휴지 기간이 완료될 때까지 기다린 후 새 크기로 조정합니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SetDesiredCapacity](#).

## Set-ASInstanceHealth

다음 코드 예시에서는 Set-ASInstanceHealth를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 인스턴스의 상태를 '비정상'으로 설정하여 서비스를 중단시킵니다. Auto Scaling은 인스턴스를 종료하고 교체합니다.

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

예 2: 이 예제에서는 지정된 인스턴스의 상태를 'Health'로 설정하여 서비스를 유지합니다. Auto Scaling 그룹의 상태 점검 유예 기간은 적용되지 않습니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SetInstanceHealth](#).

## Set-ASInstanceProtection

다음 코드 예시에서는 Set-ASInstanceProtection을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 인스턴스에 대한 인스턴스 보호를 활성화합니다.

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

예 2: 이 예에서는 지정된 인스턴스에 대한 인스턴스 보호를 비활성화합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SetInstanceProtection](#).

## Set-ASTag

다음 코드 예시에서는 Set-ASTag을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 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 시작하는 후속 EC2 인스턴스에 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} )
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 Tag 매개 변수에 대한 태그를 만들어야 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateOrUpdateTags](#).

## Start-ASPolicy

다음 코드 예시에서는 Start-ASPolicy을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 실행합니다.

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

예 2: 이 예에서는 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 참조를 참조하십시오 [ExecutePolicy](#).

## Stop-ASInstanceInAutoScalingGroup

다음 코드 예시에서는 Stop-ASInstanceInAutoScalingGroup을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 인스턴스를 종료하고 Auto Scaling 그룹의 원하는 용량을 줄여서 Auto Scaling에서 대체 인스턴스를 시작하지 않도록 합니다.

```
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       :
```

예 2: 이 예에서는 Auto Scaling 그룹의 원하는 용량을 줄이지 않고 지정된 인스턴스를 종료합니다. Auto Scaling은 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 참조를 참조하십시오 [TerminateInstanceInAutoScalingGroup](#).

## Suspend-ASProcess

다음 코드 예시에서는 Suspend-ASProcess를 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 일시 중단합니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 일시 중단합니다.

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

예 2: 이 예에서는 지정된 Auto Scaling 그룹의 모든 Auto Scaling 프로세스를 일시 중단합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SuspendProcesses](#).

## Update-ASAutoScalingGroup

다음 코드 예시에서는 Update-ASAutoScalingGroup를 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹의 최소 및 최대 크기를 업데이트합니다.

예 1: 이 예에서는 지정된 Auto Scaling 그룹의 최소 및 최대 크기를 업데이트합니다.

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

예 2: 이 예에서는 지정된 Auto Scaling 그룹의 기본 휴지 기간을 업데이트합니다.

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

예 3: 이 예에서는 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 참조를 참조하십시오 [UpdateAutoScalingGroup](#).

## Write-ASLifecycleActionHeartbeat

다음 코드 예시에서는 Write-ASLifecycleActionHeartbeat을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [RecordLifecycleActionHeartbeat](#).

## Write-ASLifecycleHook

다음 코드 예시에서는 Write-ASLifecycleHook을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 수명 주기 후크를 지정된 Auto Scaling 그룹에 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 참조를 참조하십시오 [PutLifecycleHook](#).

## Write-ASNotificationConfiguration

다음 코드 예시에서는 Write-ASNotificationConfiguration을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 EC2 인스턴스를 시작할 때 지정된 Auto Scaling 그룹이 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"
```

예 2: 이 예제는 EC2 인스턴스를 시작하거나 종료할 때 지정된 SNS 주제에 알림을 보내도록 지정된 Auto Scaling 그룹을 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 Cmdlet 참조를 참조하십시오 [PutNotificationConfiguration](#). AWS Tools for PowerShell

## Write-ASScalingPolicy

다음 코드 예시에서는 Write-ASScalingPolicy을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 정책을 지정된 Auto Scaling 그룹에 추가합니다. 지정된 조정 유형에 따라 ScalingAdjustment 파라미터 해석 방법이 결정됩니다. ChangeInCapacity"를 사용할 경우 양수 값은 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutScalingPolicy](#).

## Write-ASScheduledUpdateGroupAction

다음 코드 예시에서는 Write-ASScheduledUpdateGroupAction을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 시작 시간에 원하는 용량을 변경하기 위해 일회성 예약 작업을 생성하거나 업데이트합니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutScheduledUpdateGroupAction](#).

## AWS Budgets 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Budgets. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

## New-BGTBudget

다음 코드 예시에서는 New-BGTBudget을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이메일 알림을 통해 지정된 예산 및 시간 제약으로 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 참조를 참조하십시오 [CreateBudget](#).

## AWS Cloud9 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS Cloud9. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### Get-C9EnvironmentData

다음 코드 예시에서는 Get-C9EnvironmentData을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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
```

예 2: 이 예제는 지정된 AWS Cloud9 개발 환경의 라이프사이클 상태에 대한 정보를 가져옵니다.

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

출력:

```
FailureResource Reason Status
-----
                        -----
                        CREATED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeEnvironments](#).

## Get-C9EnvironmentList

다음 코드 예시에서는 Get-C9EnvironmentList을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 사용 가능한 AWS Cloud9 개발 환경 식별자 목록을 가져옵니다.

```
Get-C9EnvironmentList
```

출력:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ListEnvironments](#).AWS Tools for PowerShell

## Get-C9EnvironmentMembershipList

다음 코드 예시에서는 Get-C9EnvironmentMembershipList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 AWS Cloud9 개발 환경의 환경 구성원에 대한 정보를 가져옵니다.

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

출력:

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

EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

예 2: 이 예제는 지정된 AWS Cloud9 개발 환경의 소유자에 대한 정보를 가져옵니다.

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

출력:

```
EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

예 3: 이 예제는 여러 AWS Cloud9 개발 환경의 지정된 환경 구성원에 대한 정보를 가져옵니다.

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

**출력:**

```

EnvironmentId : ffd88420d4824eeeeaea8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROM0UXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROM0UXTBSU6EX

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeEnvironmentMemberships](#).

**Get-C9EnvironmentStatus**

다음 코드 예시에서는 Get-C9EnvironmentStatus을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DescribeEnvironmentStatus](#).

**New-C9EnvironmentEC2**

다음 코드 예시에서는 New-C9EnvironmentEC2을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 도구 PowerShell

예제 1: 이 예제는 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:

```
ffd88420d4824eeeeaeaa8a04bfde8cEX
```

- API 세부 정보는 Cmdlet 참조의 [CreateEnvironmentEc AWS Tools for PowerShell 2](#)를 참조하십시오.

## New-C9EnvironmentMembership

다음 코드 예시에서는 New-C9EnvironmentMembership을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 도구 PowerShell

예 1: 이 예제는 지정된 AWS Cloud9 개발 환경에 지정된 환경 구성원을 추가합니다.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
-EnvironmentId ffd88420d4824eeeeaeaa8a04bfde8cEX -Permission read-write
```

출력:

```
EnvironmentId : ffd88420d4824eeeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA602FMJWCWXHEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateEnvironmentMembership](#).

## Remove-C9Environment

다음 코드 예시에서는 Remove-C9Environment을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 AWS Cloud9 개발 환경을 삭제합니다. Amazon EC2 인스턴스가 환경에 연결되어 있는 경우 인스턴스도 종료합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteEnvironment](#).

## Remove-C9EnvironmentMembership

다음 코드 예시에서는 Remove-C9EnvironmentMembership을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 AWS Cloud9 개발 환경에서 지정된 환경 구성원을 삭제합니다.

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

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeleteEnvironmentMembership](#). AWS Tools for PowerShell

## Update-C9Environment

다음 코드 예시에서는 Update-C9Environment을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 기존 AWS Cloud9 개발 환경의 지정된 설정을 변경합니다.

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateEnvironment](#).

## Update-C9EnvironmentMembership

다음 코드 예시에서는 Update-C9EnvironmentMembership을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 지정된 AWS Cloud9 개발 환경에 대해 지정된 기존 환경 구성원의 설정을 변경합니다.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeeeaaa8a04bfde8cEX -Permission read-
only
```

출력:

```
EnvironmentId : ffd88420d4824eeeeeaaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId       : AIDAJ3BA602FMJWCXHEX
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateEnvironmentMembership](#).

## AWS CloudFormation 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS CloudFormation. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Get-CFNStack

다음 코드 예시에서는 Get-CFNStack을 사용하는 방법을 보여 줍니다.

예 1: 모든 사용자 스택을 설명하는 스택 인스턴스 컬렉션을 반환합니다.

```
Get-CFNStack
```

예 2: 지정된 스택을 설명하는 스택 인스턴스를 반환합니다.

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

예 3: 수동 페이징을 사용하여 모든 사용자 스택을 설명하는 스택 인스턴스 컬렉션을 반환합니다. 매번 호출한 후에는 다음 페이지의 시작 토큰을 검색하며, \$null은 검색할 세부 정보가 더 이상 남아 있지 않음을 나타냅니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeStacks](#).

### Get-CFNStackEvent

다음 코드 예시에서는 Get-CFNStackEvent을 사용하는 방법을 보여 줍니다.

예 1: 지정된 스택에 대한 모든 스택 관련 이벤트를 반환합니다.

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

예 2: 지정된 토큰에서 시작하는 수동 페이지징을 사용하여 지정된 스택에 대한 모든 스택 관련 이벤트를 반환합니다. 매번 호출한 후에는 다음 페이지의 시작 토큰을 검색하며, \$null은 검색할 이벤트가 더 이상 남아 있지 않음을 나타냅니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeStackEvents](#).

## Get-CFNStackResource

다음 코드 예시에서는 Get-CFNStackResource을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 논리적 ID "MyDBInstance"로 지정된 스택과 연결된 템플릿에서 식별된 리소스의 설명을 반환합니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeStackResource](#).

## Get-CFNStackResourceList

다음 코드 예시에서는 Get-CFNStackResourceList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 스택과 연결된 최대 100개의 리소스에 대한 AWS 리소스 설명을 반환합니다. 스택과 관련된 모든 리소스의 세부 정보를 얻으려면 결과의 수동 페이지징도 지원하는 StackResourceSummary Get-CFN을 사용하십시오.

```
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 관련된 모든 리소스의 세부 정보를 얻으려면 결과의 수동 페이징도 지원하는 StackResourceSummary Get-CFN을 사용하십시오.

```
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 Cmdlet 참조를 참조하십시오. [DescribeStackResources](#) AWS Tools for PowerShell

## Get-CFNStackResourceSummary

다음 코드 예시에서는 Get-CFNStackResourceSummary을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 스택과 연결된 모든 리소스에 대한 설명을 반환합니다.

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

예 2: 결과의 수동 페이징을 사용하여 지정된 스택과 연결된 모든 리소스에 대한 설명을 반환합니다. 매번 호출한 후에는 다음 페이지의 시작 토큰을 검색하며, \$null은 검색할 세부 정보가 더 이상 남아 있지 않음을 나타냅니다.

```
$nextToken = $null
do {
    Get-CFNStackResourceSummary -StackName "myStack" -NextToken $nextToken
```

```
$nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListStackResources](#).

## Get-CFNStackSummary

다음 코드 예시에서는 Get-CFNStackSummary을 사용하는 방법을 보여 줍니다.

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")
```

예 4: 결과의 수동 페이징을 사용하여 현재 생성 중이거나 업데이트 중인 모든 스택에 대한 요약 정보를 반환합니다. 매번 호출한 후에는 다음 페이지의 시작 토큰을 검색하며, \$null은 검색할 세부 정보가 더 이상 남아 있지 않음을 나타냅니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListStacks](#).

## Get-CFNTemplate

다음 코드 예시에서는 Get-CFNTemplate을 사용하는 방법을 보여 줍니다.

## 예 1: Get-CFNTemplate에 대한 도구 PowerShell

예 1: 지정된 스택과 연결된 템플릿을 반환합니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetTemplate](#).

## Measure-CFNTemplateCost

다음 코드 예시에서는 Measure-CFNTemplateCost을 사용하는 방법을 보여 줍니다.

### 예 1: Get-CFNTemplate에 대한 도구 PowerShell

예 1: 템플릿을 실행하는 데 필요한 리소스를 설명하는 쿼리 문자열이 포함된 AWS 단순 월별 계산기 URL을 반환합니다. 템플릿은 지정된 Amazon S3 URL에서 가져오고 단일 사용자 지정 파라미터가 적용됩니다. "와 " 대신 '키'와 '값'을 사용하여 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

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

예 2: 템플릿을 실행하는 데 필요한 리소스를 설명하는 쿼리 문자열이 포함된 AWS 단순 월별 계산기 URL을 반환합니다. 제공된 콘텐츠에서 템플릿을 구문 분석하고 사용자 지정 매개 변수를 적용합니다. 이 예제에서는 템플릿 콘텐츠가 두 개의 매개 변수 KeyName InstanceType "와 "를 선언했다고 가정합니다. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

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

예 3: New-Object를 사용하여 템플릿 매개 변수 세트를 만들고 템플릿 실행에 필요한 리소스를 설명하는 쿼리 문자열이 포함된 AWS 단순 월별 계산기 URL을 반환합니다. 제공된 콘텐츠에서

사용자 지정 매개 변수를 사용하여 템플릿을 구문 분석합니다. 이 예제에서는 템플릿 콘텐츠가 'KeyName' 및 "라는 두 개의 매개 변수를 선언했다고 가정합니다. InstanceType

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

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EstimateTemplateCost](#).

## New-CFNStack

다음 코드 예시에서는 New-CFNStack을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue 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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue 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 매개 변수 ('PK1'은 템플릿 콘텐츠에 선언된 매개 변수의 이름을 나타내고, 'PV1'은 매개 변수의 값을 나타냄) 가 있는 Amazon S3 URL에서 가져옵니다. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue 스택 생성이 실패하면 스택이 롤백됩니다 (-DisableRollback \$false 지정과 동일).

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

예 4: 지정된 이름으로 새 스택을 생성합니다. 템플릿은 사용자 지정 매개 변수 ('PK1'은 템플릿 콘텐츠에 선언된 매개 변수의 이름을 나타내고, 'PV1'은 매개 변수의 값을 나타냄) 가 있는 Amazon S3 URL에서 가져옵니다. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue 스택 생성이 실패하면 스택이 롤백됩니다 (-DisableRollback \$false를 지정하는 것과 동일). 지정된 알림 AEN은 게시된 스택 관련 이벤트를 수신합니다.

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

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateStack](#).

## Remove-CFNStack

다음 코드 예시에서는 Remove-CFNStack을 사용하는 방법을 보여 줍니다.

예 1: 지정된 스택을 삭제합니다.

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteStack](#).

## Resume-CFNUpdateRollback

다음 코드 예시에서는 Resume-CFNUpdateRollback을 사용하는 방법을 보여 줍니다.

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ContinueUpdateRollback](#).

## Stop-CFNUpdateStack

다음 코드 예시에서는 Stop-CFNUpdateStack을 사용하는 방법을 보여 줍니다.

예 1: 지정된 스택에 대한 업데이트를 취소합니다.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CancelUpdateStack](#).

## Test-CFNStack

다음 코드 예시에서는 Test-CFNStack을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 스택이 UPDATE\_ROLLBACK\_COMPLETE, CREATE\_COMPLETE, 롤백\_완료 또는 업데이트\_완료 상태 중 하나에 도달했는지 테스트합니다.

```
Test-CFNStack -StackName MyStack
```

출력:

```
False
```

예 2: 스택이 UPDATE\_COMPLETE 또는 UPDATE\_ROLLBACK\_COMPLETE 상태에 도달했는지 테스트합니다.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

출력:

```
True
```

- [API에 대한 자세한 내용은 Cmdlet 참조의 Test-CFNstack을 참조하십시오.AWS Tools for PowerShell](#)

## Test-CFNTemplate

다음 코드 예시에서는 Test-CFNTemplate을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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/mytemplates/
templatefile.template
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ValidateTemplate](#).

## Update-CFNStack

다음 코드 예시에서는 Update-CFNStack을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 지정된 템플릿 및 사용자 지정 파라미터를 사용하여 'myStack' 스택을 업데이트합니다. 'PK1'은 템플릿에 선언된 파라미터의 이름을 나타내고 'PV1'은 해당 값을 나타냅니다. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

```
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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

```
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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

```
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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
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. " 및 " 대신 '키'와 '값'을 사용하여 사용자 지정 매개 변수를 지정할 수도 있습니다. ParameterKey ParameterValue IAM 리소스가 포함된 스택에서는 -Capability\_IAM 매개변수를 지정해야 합니다. 그렇지 않으면 업데이트가 실패하고 " 오류가 발생합니다. InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/mytemplates/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
@{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오. [UpdateStack](#) AWS Tools for PowerShell

## Wait-CFNStack

다음 코드 예시에서는 Wait-CFNStack을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 스택이 UPDATE\_ROLLBACK\_COMPLETE, CREATE\_COMPLETE, 롤백\_완료 또는 업데이트\_완료 상태 중 하나에 도달했는지 테스트합니다. 스택이 다음 상태 중 하나가 아닌 경우 명령이 2 초 동안 휴면 상태로 있다가 상태를 다시 테스트합니다. 스택이 요청된 상태 중 하나에 도달하거나

기본 제한 시간인 60초가 경과할 때까지 이 과정이 반복됩니다. 제한 시간이 초과되면 예외가 발생합니다. 제한 시간 내에 스택이 요청된 상태 중 하나에 도달하면 파이프라인으로 반환됩니다.

```
$stack = Wait-CFNStack -StackName MyStack
```

예 2: 이 예에서는 스택이 지정된 상태 중 하나에 도달할 때까지 총 5분 (300초) 동안 기다립니다. 이 예제에서는 제한 시간이 초과되기 전에 상태에 도달하므로 스택 개체가 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
```

예 3: 이 예제는 제한 시간 (이 경우 기본 기간인 60초) 내에 스택이 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

```

- API에 대한 세부 정보는 Cmdlet 참조의 [wait-CFNstack](#)을 참조하십시오.AWS Tools for PowerShell

## CloudFront 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CloudFront. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

### 주제

- [작업](#)

## 작업

### Get-CFCloudFrontOriginAccessIdentity

다음 코드 예시에서는 Get-CFCloudFrontOriginAccessIdentity을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

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 참조를 참조하십시오 [GetCloudFrontOriginAccessIdentity](#).

### Get-CFCloudFrontOriginAccessIdentityConfig

다음 코드 예시에서는 Get-CFCloudFrontOriginAccessIdentityConfig을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

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 참조를 참조하십시오 [GetCloudFrontOriginAccessIdentityConfig](#).

## Get-CFCloudFrontOriginAccessIdentityList

다음 코드 예시에서는 Get-CFCloudFrontOriginAccessIdentityList을 사용하는 방법을 보여줍니다.

예에 대한 도구 PowerShell

- 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, E1YWXXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXXX9B
Quantity    : 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListCloudFrontOriginAccessIdentities](#).

## Get-CFDistribution

다음 코드 예시에서는 Get-CFDistribution을 사용하는 방법을 보여줍니다.

예에 대한 도구 PowerShell

- 예 1: 특정 배포에 대한 정보를 검색합니다.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDistribution](#).

## Get-CFDistributionConfig

다음 코드 예시에서는 Get-CFDistributionConfig을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 특정 배포에 대한 구성을 검색합니다.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDistributionConfig](#).

## Get-CFDistributionList

다음 코드 예시에서는 Get-CFDistributionList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 배포판을 반환합니다.

```
Get-CFDistributionList
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDistributions](#).

## New-CFDistribution

다음 코드 예시에서는 New-CFDistribution을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 로깅 및 캐싱으로 구성된 기본 CloudFront 배포를 만듭니다.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
```

```

$origin.DomainName = "ps-cmdlet-sample.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
    -DistributionConfig_Enabled $true `
    -DistributionConfig_Comment "Test distribution" `
    -Origins_Item $origin `
    -Origins_Quantity 1 `
    -Logging_Enabled $true `
    -Logging_IncludeCookie $true `
    -Logging_Bucket ps-cmdlet-sample-logging.s3.amazonaws.com `
    -Logging_Prefix "help/" `
    -DistributionConfig_CallerReference Client1 `
    -DistributionConfig_DefaultRootObject index.html `
    -DefaultCacheBehavior_TargetOriginId $origin.Id `
    -ForwardedValues_QueryString $true `
    -Cookies_Forward all `
    -WhitelistedNames_Quantity 0 `
    -TrustedSigners_Enabled $false `
    -TrustedSigners_Quantity 0 `
    -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
    -DefaultCacheBehavior_MinTTL 1000 `
    -DistributionConfig_PriceClass "PriceClass_All" `
    -CacheBehaviors_Quantity 0 `
    -Aliases_Quantity 0

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateDistribution](#).

## New-CFInvalidation

다음 코드 예시에서는 New-CFInvalidation을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제에서는 ID가 EXAMPLNSTXAXE인 배포에 대해 새 무효화를 생성합니다. CallerReference 는 사용자가 선택한 고유 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 "EXAMPLNSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
3
```

출력:

```
Invalidation          Location
-----
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLNSTXAXE/invalidation/EXAMPLE8NOK9H
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateInvalidation](#).

## New-CFSignedCookie

다음 코드 예시에서는 New-CFSignedCookie을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 미리 준비된 정책을 사용하여 지정된 리소스에 서명된 쿠키를 만듭니다. 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]
```

예 2: 사용자 지정 정책을 사용하여 지정된 리소스에 서명된 쿠키를 만듭니다. 쿠키는 24시간 동안 유효하며 일주일 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...
```

예 3: 사용자 지정 정책을 사용하여 지정된 리소스에 서명된 쿠키를 만듭니다. 쿠키는 24시간 동안 유효하며 일주일 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...
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet SignedCookie 참조의 [New-CF](#)를 참조하십시오.

## New-CFSignedUrl

다음 코드 예시에서는 New-CFSignedUrl을 사용하는 방법을 보여 줍니다.

### 예에 대한 도구 PowerShell

예 1: 미리 준비된 정책을 사용하여 지정된 리소스에 서명된 URL을 만듭니다. URL은 1시간 동안 유효합니다. 서명된 URL을 포함하는 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
```

예 2: 사용자 지정 정책을 사용하여 지정된 리소스에 서명된 URL을 만듭니다. URL은 24시간 후에 유효하며 일주일 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
```

예 3: 사용자 지정 정책을 사용하여 지정된 리소스에 서명된 URL을 만듭니다. URL은 24시간 후에 유효하며 일주일 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
```

```
New-CFSignedUrl @params
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet SignedUrl 참조의 [New-CF](#)를 참조하십시오.

## CloudTrail 도구를 사용하는 예: PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
CloudTrail. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Find-CTEvent

다음 코드 예시에서는 Find-CTEvent를 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 지난 7일 동안 발생한 모든 이벤트를 반환합니다. 기본적으로 cmdlet은 모든 이벤트를 전달하기 위해 자동으로 여러 번 호출하며, 서비스에서 더 이상 사용할 수 있는 데이터가 없다고 표시되면 종료됩니다.

```
Find-CTEvent
```

예 2: 현재 셸 기본값이 아닌 지역을 지정하여 지난 7일 동안 발생한 모든 이벤트를 반환합니다.

```
Find-CTEvent -Region eu-central-1
```

예 3: RunInstances API 호출과 관련된 모든 이벤트를 반환합니다.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
    AttributeValue="RunInstances" }
```

예 4: 사용 가능한 처음 5개의 이벤트를 반환합니다. 추가 이벤트를 검색하는 데 사용할 토큰은 NextToken "이라는 이름의 노트 속성으로 `$AWSHistory.LastServiceResponse` 멤버에 첨부됩니다.

```
Find-CTEvent -MaxResult 5
```

예 5: 시퀀스에서 이벤트 반환을 시작할 위치를 나타내는 이전 호출의 '다음 페이지' 토큰을 사용하여 다음 10개 이벤트를 반환합니다.

```
Find-CTEvent -MaxResult 10 -NextToken $AWSHistory.LastServiceResponse.NextToken
```

예 6: 이 예제에서는 호출당 최대 5개의 이벤트를 가져오는 수동 페이지징을 사용하여 사용 가능한 이벤트를 반복하는 방법을 보여줍니다.

```
$nextToken = $null
do
{
    Find-CTEvent -MaxResult 5 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [LookupEvents](#).AWS Tools for PowerShell

## Get-CTTrail

다음 코드 예시에서는 Get-CTTrail을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 계정의 현재 지역과 관련된 모든 트레일의 설정을 반환합니다.

```
Get-CTTrail
```

예 2: 지정된 트레일의 설정을 반환합니다.

```
Get-CTTrail -TrailNameList trail1, trail2
```

예 3: 현재 셸 기본값이 아닌 지역 (이 경우 프랑크푸르트 (eu-central-1) 지역) 에서 생성된 지정된 트레일의 설정을 반환합니다.

```
Get-CTTrail -TrailNameList trailABC, trailDEF -Region eu-central-1
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [DescribeTrails](#) AWS Tools for PowerShell

## Get-CTTrailStatus

다음 코드 예시에서는 Get-CTTrailStatus을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이름이 myExampleTrail "인 트레일의 상태 정보를 반환합니다. 반환되는 데이터에는 전송 오류, Amazon SNS 및 Amazon S3 오류, 트레일의 시작 및 중지 로깅 시간에 대한 정보가 포함됩니다. 이 예에서는 현재 셸 기본값과 동일한 리전에 트레일이 생성되었다고 가정합니다.

```
Get-CTTrailStatus -Name myExampleTrail
```

예 2: 현재 셸 기본값이 아닌 지역 (이 경우에는 프랑크푸르트 (eu-central-1) 지역) 에서 생성된 트레일의 상태 정보를 반환합니다.

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [GetTrailStatus](#). AWS Tools for PowerShell

## New-CTTrail

다음 코드 예시에서는 New-CTTrail을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예제 1: 로그 파일 스토리지로 'mycloudtrailbucket' 버킷을 사용할 트레일을 생성합니다.

```
New-CTTrail -Name="awscloudtrail-example" -S3BucketName="mycloudtrailbucket"
```

예제 2: 로그 파일 저장에 'mycloudtrailbucket' 버킷을 사용할 트레일을 생성합니다. 로그를 나타내는 S3 객체에는 'mylogs'라는 공통 키 접두사가 붙습니다. 새 로그가 버킷으로 전송되면 SNS 주제 'mlog-deliverytopic'에 알림이 전송됩니다. 이 예제에서는 스포래팅을 사용하여 cmdlet에 매개 변수 값을 제공합니다.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="mycloudtrailbucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CreateTrail](#). AWS Tools for PowerShell

## Remove-CTTrail

다음 코드 예시에서는 Remove-CTTrail을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 지정된 트레일을 삭제합니다. 명령을 실행하기 전에 확인 메시지가 표시됩니다. 확인을 표시하지 않으려면 -Force 스위치 파라미터를 추가하십시오.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteTrail](#).

## Start-CTLogging

다음 코드 예시에서는 Start-CTLogging을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이름이 myExampleTrail "인 트레일에 대한 AWS API 호출 및 로그 파일 전송 기록을 시작합니다. 이 예제에서는 트레일이 현재 셸 기본값과 동일한 리전에 생성되었다고 가정합니다.

```
Start-CTLogging -Name myExampleTrail
```

예 2: 현재 셸 기본값이 아닌 리전 (이 경우에는 프랑크푸르트 (eu-central-1) 리전) 에서 생성된 트레일에 대한 AWS API 호출 및 로그 파일 전송 기록을 시작합니다.

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [StartLogging](#).AWS Tools for PowerShell

## Stop-CTLogging

다음 코드 예시에서는 Stop-CTLogging을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이름이 myExampleTrail "인 트레일에 대한 AWS API 호출 기록 및 로그 파일 전송을 일시 중단합니다. 이 예제에서는 트레일이 현재 셸 기본값과 동일한 리전에 생성되었다고 가정합니다.

```
Stop-CTLogging -Name myExampleTrail
```

예 2: 현재 셸 기본값이 아닌 리전 (이 경우에는 프랑크푸르트 (eu-central-1) 리전) 에서 생성된 트레일에 대한 AWS API 호출 기록 및 로그 파일 전송을 일시 중단합니다.

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [StopLogging](#)AWS Tools for PowerShell

## Update-CTTrail

다음 코드 예시에서는 Update-CTTrail을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 글로벌 서비스 이벤트 (예: IAM의 이벤트) 가 기록되도록 지정된 트레일을 업데이트하고 전달되는 로그 파일의 공통 키 접두사를 'globallogs'로 변경합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -S3KeyPrefix "globallogs"
```

예 2: 새 로그 전송에 대한 알림이 지정된 SNS 주제로 전송되도록 지정된 트레일을 업데이트합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

예 3: 로그가 다른 버킷으로 전송되도록 지정된 트레일을 업데이트합니다.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateTrail](#).

## CloudWatch 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CloudWatch. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Get-CWDashboard

다음 코드 예시에서는 Get-CWDashboard을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예시 1: 지정된 대시보드의 arn 본문을 반환합니다.

```
Get-CWDashboard -DashboardName Dashboard1
```

출력:

```
DashboardArn          DashboardBody
-----
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDashboard](#).

## Get-CWDashboardList

다음 코드 예시에서는 Get-CWDashboardList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDashboards](#).

## Remove-CWDashboard

다음 코드 예시에서는 Remove-CWDashboard을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 지정된 대시보드를 삭제하고 계속하기 전에 확인을 위해 승격합니다. 확인을 우회하려면 명령에 -Force 스위치를 추가합니다.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteDashboards](#).

## Write-CWDashboard

다음 코드 예시에서는 Write-CWDashboard을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예시 1: 지표 위젯 2개를 나란히 포함하도록 'Dashboard1'이라는 대시보드를 생성하거나 업데이트 합니다.

```
$dashBody = @"
{
  "widgets":[
    {
      "type":"metric",
      "x":0,
      "y":0,
      "width":12,
      "height":6,
      "properties":{"
        "metrics":[
          [
            "AWS/EC2",
            "CPUUtilization",
            "InstanceId",
            "i-012345"
          ]
        ],
        "period":300,
        "stat":"Average",
        "region":"us-east-1",
        "title":"EC2 Instance CPU"
      }
    },
    {
      "type":"metric",
      "x":12,
      "y":0,
```

```

        "width":12,
        "height":6,
        "properties":{
            "metrics":[
                [
                    "AWS/S3",
                    "BucketSizeBytes",
                    "BucketName",
                    "MyBucketName"
                ]
            ],
            "period":86400,
            "stat":"Maximum",
            "region":"us-east-1",
            "title":"MyBucketName bytes"
        }
    ]
}
"@

```

```
Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

예시 2: 대시보드를 생성하거나 업데이트하여 대시보드를 설명하는 콘텐츠를 cmdlet에 전달합니다.

```

$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutDashboard](#).

## Write-CWMetricData

다음 코드 예시에서는 Write-CWMetricData을 사용하는 방법을 보여 줍니다.

## 예 1: 새 MetricDatum 객체를 생성하여 Amazon Web Services CloudWatch 메트릭스에 씁니다.

예 1: 새 MetricDatum 객체를 생성하여 Amazon Web Services CloudWatch 메트릭스에 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 참조를 참조하십시오 [PutMetricData](#).

## CodeCommit 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다. CodeCommit. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-CCBranch

다음 코드 예시에서는 Get-CCBranch을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 저장소의 지정된 브랜치에 대한 정보를 가져옵니다.

예 1: 이 예제는 지정된 저장소의 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 참조를 참조하십시오 [GetBranch](#).

## Get-CCBranchList

다음 코드 예시에서는 Get-CCBranchList을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 지정된 리포지토리의 브랜치 이름 목록을 가져옵니다.

예 1: 이 예제는 지정된 리포지토리의 브랜치 이름 목록을 가져옵니다.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

출력:

```
master
MyNewBranch
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListBranches](#).

## Get-CCRepository

다음 코드 예시에서는 Get-CCRepository을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 지정된 저장소에 대한 정보를 가져옵니다.

예 1: 이 예제는 지정된 저장소에 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 참조를 참조하십시오 [GetRepository](#).

**Get-CCRepositoryBatch**

다음 코드 예시에서는 Get-CCRepositoryBatch을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 리포지토리 중 발견된 리포지토리와 찾을 수 없는 리포지토리를 확인합니다.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

**출력:**

```

Repositories                RepositoriesNotFound
-----
{MyDemoRepo, MyNewRepo}     {AMissingRepo}

```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [BatchGetRepositories](#).AWS Tools for PowerShell

**Get-CCRepositoryList**

다음 코드 예시에서는 Get-CCRepositoryList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 모든 리포지토리를 리포지토리 이름을 기준으로 오름차순으로 나열합니다.

예 1: 이 예에서는 모든 리포지토리를 리포지토리 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 Cmdlet 참조를 참조하십시오 [ListRepositories](#).AWS Tools for PowerShell

## New-CCBranch

다음 코드 예시에서는 New-CCBranch를 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 지정된 리포지토리의 지정된 이름과 지정된 커밋 ID를 사용하여 새 브랜치를 만듭니다.

예 1: 이 예제는 지정된 리포지토리의 지정된 이름과 지정된 커밋 ID를 사용하여 새 브랜치를 만듭니다.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId 7763222d...561fc9c9
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateBranch](#).

## New-CCRepository

다음 코드 예시에서는 New-CCRepository를 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 지정된 이름과 지정된 설명을 사용하여 새 저장소를 만듭니다.

예 1: 이 예제는 지정된 이름과 지정된 설명을 사용하여 새 저장소를 만듭니다.

```
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 참조를 참조하십시오 [CreateRepository](#).

**Remove-CCRepository**

다음 코드 예시에서는 Remove-CCRepository을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

예 1: 이 예제는 지정된 저장소를 강제로 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 리포지토리를 삭제하려면 -Force 매개 변수를 추가합니다.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

**출력:**

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteRepository](#).

**Update-CCDefaultBranch**

다음 코드 예시에서는 Update-CCDefaultBranch을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 리포지토리의 기본 브랜치를 지정된 브랜치로 변경합니다.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateDefaultBranch](#).

## Update-CCRepositoryDescription

다음 코드 예시에서는 Update-CCRepositoryDescription을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 저장소의 설명을 변경합니다.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
"This is an updated description."
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateRepositoryDescription](#).

## Update-CCRepositoryName

다음 코드 예시에서는 Update-CCRepositoryName을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 저장소의 이름을 변경합니다.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateRepositoryName](#).

## CodeDeploy 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CodeDeploy. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Add-CDOnPremiseInstanceTag

다음 코드 예시에서는 Add-CDOnPremiseInstanceTag을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 온프레미스 인스턴스의 지정된 키와 값이 포함된 온프레미스 인스턴스 태그를 추가합니다.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
"Value" = "CodeDeployDemo-OnPrem"}
```

- API 세부 정보는 [AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 AddTagsToOnPremisesInstances.](#)

### Get-CDApplication

다음 코드 예시에서는 Get-CDApplication을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [GetApplication](#).

## Get-CDApplicationBatch

다음 코드 예시에서는 Get-CDApplicationBatch을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [BatchGetApplications](#).

## Get-CDApplicationList

다음 코드 예시에서는 Get-CDApplicationList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 사용 가능한 애플리케이션 목록을 가져옵니다.

예 1: 이 예제는 사용 가능한 애플리케이션 목록을 가져옵니다.

```
Get-CDApplicationList
```

출력:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListApplications](#).

## Get-CDApplicationRevision

다음 코드 예시에서는 Get-CDApplicationRevision을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 애플리케이션 버전에 대한 정보를 가져옵니다.

예 1: 이 예제는 지정된 애플리케이션 버전에 대한 정보를 가져옵니다.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication -
S3Location_Bucket MyBucket -Revision_RevisionType S3 -S3Location_Key 5xd27EX.zip -
S3Location_BundleType zip -S3Location_ETag 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

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 참조를 참조하십시오 [GetApplicationRevision](#).

## Get-CDApplicationRevisionList

다음 코드 예시에서는 Get-CDApplicationRevisionList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 애플리케이션에 사용할 수 있는 수정 버전에 대한 정보를 가져옵니다.

예 1: 이 예에서는 지정된 애플리케이션에 사용할 수 있는 수정 버전에 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 = MyBucket, BundleType = zip, ETag =
  4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListApplicationRevisions](#).

## Get-CDDeployment

다음 코드 예시에서는 Get-CDDeployment을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 배포에 대한 요약 정보를 가져옵니다.

예 1: 이 예제는 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

```

예 2: 이 예제는 지정된 배포에 참여하고 있는 인스턴스의 상태에 대한 정보를 가져옵니다.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

출력:

```

Failed      : 0
InProgress  : 0
Pending     : 0
Skipped     : 0
Succeeded   : 3

```

예 3: 이 예에서는 지정된 배포의 애플리케이션 수정 버전에 대한 정보를 가져옵니다.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

출력:

```

Bucket      : MyBucket
BundleType  : zip
ETag        : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key         : clzfqEX
Version     :

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDeployment](#).

## Get-CDDeploymentBatch

다음 코드 예시에서는 Get-CDDeploymentBatch을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [BatchGetDeployments](#).

## Get-CDDeploymentConfig

다음 코드 예시에서는 Get-CDDeploymentConfig을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 배포 구성에 대한 요약 정보를 가져옵니다.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

출력:

CreateTime	DeploymentConfigId	DeploymentConfigName
10/3/2014 4:32:30 PM	518a3950-d034-46a1-9d2c-3c949EXAMPLE	ThreeQuartersHealthy
MinimumHealthyHosts	Amazon.CodeDeploy.Model.MinimumHealthyHosts	

예 2: 이 예제에서는 지정된 배포 구성의 정의에 대한 정보를 가져옵니다.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy).MinimumHealthyHosts)
```

출력:

Type	Value
FLEET_PERCENT	75

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDeploymentConfig](#).

## Get-CDDeploymentConfigList

다음 코드 예시에서는 Get-CDDeploymentConfigList을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 사용 가능한 배포 구성 목록을 가져옵니다.

```
Get-CDDeploymentConfigList
```

출력:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDeploymentConfigs](#).

## Get-CDDeploymentGroup

다음 코드 예시에서는 Get-CDDeploymentGroup을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 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-09MWP7XTL8EX
TargetRevision            : Amazon.CodeDeploy.Model.RevisionLocation
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDeploymentGroup](#).

## Get-CDDeploymentGroupList

다음 코드 예시에서는 Get-CDDeploymentGroupList를 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 애플리케이션의 배포 그룹 목록을 가져옵니다.

예 1: 이 예제는 지정된 애플리케이션의 배포 그룹 목록을 가져옵니다.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

출력:

```
ApplicationName      DeploymentGroups
NextToken
-----
-----
CodeDeployDemoApplication  {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDeploymentGroups](#).

## Get-CDDeploymentInstance

다음 코드 예시에서는 Get-CDDeploymentInstance를 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 배포의 지정된 인스턴스에 대한 정보를 가져옵니다.

예 1: 이 예제는 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 참조를 참조하십시오 [GetDeploymentInstance](#).

## Get-CDDeploymentInstanceList

다음 코드 예시에서는 Get-CDDeploymentInstanceList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예제 1: 이 예제는 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 참조를 참조하십시오 [ListDeploymentInstances](#).

## Get-CDDeploymentList

다음 코드 예시에서는 Get-CDDeploymentList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [ListDeployments](#).

## Get-CDOnPremiseInstance

다음 코드 예시에서는 Get-CDOnPremiseInstance을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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 참조를 참조하십시오 [GetOnPremisesInstance](#).

## Get-CDOnPremiseInstanceBatch

다음 코드 예시에서는 Get-CDOnPremiseInstanceBatch을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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 참조를 참조하십시오 [BatchGetOnPremisesInstances](#).

## Get-CDOnPremiseInstanceList

다음 코드 예시에서는 Get-CDOnPremiseInstanceList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 예제는 사용 가능한 온프레미스 인스턴스 이름 목록을 가져옵니다.

```
Get-CDOnPremiseInstanceList
```

출력:

```

AssetTag12010298EX
AssetTag12010298EX-2

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListOnPremisesInstances](#).

## New-CDApplication

다음 코드 예시에서는 New-CDApplication을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 예제는 지정된 이름으로 새 애플리케이션을 만듭니다.

```
New-CDApplication -ApplicationName MyNewApplication
```

출력:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateApplication](#).

## New-CDDeployment

다음 코드 예시에서는 New-CDDeployment을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 배포 구성 및 응용 프로그램 개정을 사용하여 지정된 응용 프로그램 및 배포 그룹에 대한 새 배포를 만듭니다.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket
-S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime -
DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

출력:

```
d-ZHROG7UEX
```

예 2: 이 예제는 블루/그린 배포를 위한 대체 환경에 포함시키기 위해 인스턴스를 식별해야 하는 EC2 인스턴스 태그 그룹을 지정하는 방법을 보여줍니다.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket MyBucket
-S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
-DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
-S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
@(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

출력:

```
d-ZHROG7UEX
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [CreateDeployment](#). AWS Tools for PowerShell

## New-CDDeploymentConfig

다음 코드 예시에서는 New-CDDeploymentConfig을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 이름과 동작으로 새 배포 구성을 만듭니다.

예 1: 이 예에서는 지정된 이름과 동작으로 새 배포 구성을 만듭니다.

```
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 참조를 참조하십시오 [CreateDeploymentConfig](#).

## New-CDDeploymentGroup

다음 코드 예시에서는 New-CDDeploymentGroup을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 애플리케이션에 대해 지정된 이름, Auto Scaling 그룹, 배포 구성, 태그, 서비스 역할을 가진 배포 그룹을 만듭니다.

예 1: 이 예에서는 지정된 애플리케이션에 대해 지정된 이름, Auto Scaling 그룹, 배포 구성, 태그, 서비스 역할을 가진 배포 그룹을 만듭니다.

```
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
```

예 2: 이 예제는 블루/그린 배포의 대체 환경에 포함시키기 위해 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 Cmdlet 참조를 참조하십시오 [CreateDeploymentGroup](#). AWS Tools for PowerShell

## Register-CDApplicationRevision

다음 코드 예시에서는 Register-CDApplicationRevision을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 Amazon S3 위치에 지정된 애플리케이션의 애플리케이션 수정 버전을 등록합니다.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
MyBucket -S3Location_BundleType zip -S3Location_Key aws-codedeploy_linux-master.zip
-Revision_RevisionType S3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterApplicationRevision](#).

## Register-CDOnPremiseInstance

다음 코드 예시에서는 Register-CDOnPremiseInstance을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 온프레미스 인스턴스를 지정된 이름 및 IAM 사용자로 등록합니다.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/  
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [RegisterOnPremisesInstance](#). AWS Tools for PowerShell

## Remove-CDApplication

다음 코드 예시에서는 Remove-CDApplication을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 이름의 애플리케이션을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 애플리케이션을 삭제하려면 -Force 매개 변수를 추가합니다.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteApplication](#).

## Remove-CDDeploymentConfig

다음 코드 예시에서는 Remove-CDDeploymentConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 이름의 배포 구성을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 배포 구성을 삭제하려면 -Force 매개 변수를 추가합니다.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteDeploymentConfig](#).

## Remove-CDDeploymentGroup

다음 코드 예시에서는 Remove-CDDeploymentGroup을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 애플리케이션에 대해 지정된 이름을 가진 배포 그룹을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 배포 그룹을 삭제하려면 `-Force` 매개 변수를 추가합니다.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName  
MyNewDeploymentGroup
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet [DeleteDeploymentGroup](#) 참조의 내용을 참조하십시오.

## Remove-CDOnPremiseInstanceTag

다음 코드 예시에서는 `Remove-CDOnPremiseInstanceTag`을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 이름을 가진 온프레미스 인스턴스의 지정된 태그를 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 태그를 삭제하려면 `-Force` 파라미터를 추가합니다.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =  
"Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RemoveTagsFromOnPremisesInstances](#).

## Stop-CDDeployment

다음 코드 예시에서는 `Stop-CDDeployment`을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 배포 ID로 배포를 중지하려고 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 참조를 참조하십시오 [StopDeployment](#).

## Unregister-CDOnPremiseInstance

다음 코드 예시에서는 Unregister-CDOnPremiseInstance을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제는 지정된 이름의 온프레미스 인스턴스 등록을 취소합니다.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeregisterOnPremisesInstance](#). AWS Tools for PowerShell

## Update-CDApplication

다음 코드 예시에서는 Update-CDApplication을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제는 지정된 애플리케이션의 이름을 변경합니다.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
MyNewApplication-2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateApplication](#).

## Update-CDDeploymentGroup

다음 코드 예시에서는 Update-CDDeploymentGroup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예에서는 지정된 애플리케이션의 지정된 배포 그룹 이름을 변경합니다.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
MyNewDeploymentGroup-2
```

예 2: 이 예제는 블루/그린 배포의 대체 환경에 포함시키기 위해 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 Cmdlet 참조를 참조하십시오 [UpdateDeploymentGroup](#). AWS Tools for PowerShell

## CodePipeline 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CodePipeline. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Confirm-CPJob

다음 코드 예시에서는 Confirm-CPJob을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [AcknowledgeJob](#).

## Disable-CPStageTransition

다음 코드 예시에서는 Disable-CPStageTransition을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [DisableStageTransition](#).

## Enable-CPStageTransition

다음 코드 예시에서는 Enable-CPStageTransition을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 파이프라인의 지정된 단계에 대한 인바운드 전환을 활성화합니다.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -TransitionType Inbound
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableStageTransition](#).

## Get-CPActionType

다음 코드 예시에서는 Get-CPActionType을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 소유자가 수행할 수 있는 모든 작업에 대한 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 참조를 참조하십시오 [ListActionTypes](#).

## Get-CPActionableJobList

다음 코드 예시에서는 Get-CPActionableJobList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 작업 범주, 소유자, 공급자, 버전 및 쿼리 매개변수에 대해 실행 가능한 모든 작업에 대한 정보를 가져옵니다.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
-ACTIONTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
@{"ProjectName" = "MyProjectName"}
```

출력:

AccountId	Data	Id
----- -----	-----	--
80398EXAMPLE f57a0EXAMPLE	Amazon.CodePipeline.Model.JobData 3	0de392f5-712d-4f41-ace3-

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PollForJobs](#).

## Get-CPJobDetail

다음 코드 예시에서는 Get-CPJobDetail을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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		

예 2: 이 예제에서는 지정된 작업에 대한 세부 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 참조를 참조하십시오 [GetJobDetails](#).

## Get-CPPipeline

다음 코드 예시에서는 Get-CPPipeline을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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

```

예 2: 이 예제에서는 지정된 파이프라인에 대한 세부 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 = MyBucketName
  Type = S3
Stages:
  Name = Source
```

```
Actions:
  Name = Source
  Category = Source
  Owner = ThirdParty
  Provider = GitHub
  Version = 1
  Configuration:
    Branch = master
    OAuthToken = ****
    Owner = my-user-name
    Repo = MyRepoName
  InputArtifacts:
    MyApp
  RunOrder = 1
Name = Build
Actions:
  Name = Build
  Category = Build
  Owner = Custom
  Provider = MyCustomProviderName
  Version = 1
  Configuration:
    ProjectName = MyProjectName
  InputArtifacts:
    MyApp
    MyAppBuild
  RunOrder = 1
Name = Beta
Actions:
  Name = CodePipelineDemoFleet
  Category = Deploy
  Owner = AWS
  Provider = CodeDeploy
  Version = 1
  Configuration:
    ApplicationName = CodePipelineDemoApplication
    DeploymentGroupName = CodePipelineDemoFleet
  InputArtifacts:
    MyAppBuild
  RunOrder = 1
Name = TestStage
Actions:
  Name = MyJenkinsTestAction
  Category = Test
```

```

Owner = Custom
Provider = MyCustomTestProvider
Version = 1
Configuration:
  ProjectName = MyJenkinsProjectName
InputArtifacts:
  MyAppBuild
RunOrder = 1

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPipeline](#).

## Get-CPPipelineList

다음 코드 예시에서는 Get-CPPipelineList을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 사용 가능한 파이프라인 목록을 가져옵니다.

예제 1: 이 예제는 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 참조를 참조하십시오 [ListPipelines](#).

## Get-CPPipelineState

다음 코드 예시에서는 Get-CPPipelineState을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 파이프라인의 단계에 대한 일반 정보를 가져옵니다.

예 1: 이 예제는 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

```

예 2: 이 예제에서는 지정된 파이프라인 상태에 대한 세부 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 참조를 참조하십시오 [GetPipelineState](#).

## New-CPCustomActionType

다음 코드 예시에서는 New-CPCustomActionType을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 속성을 사용하여 새 사용자 지정 액션을 만듭니다.

```

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 참조를 참조하십시오 [CreateCustomActionType](#).

## New-CPPipeline

다음 코드 예시에서는 New-CPPipeline을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예제 1: 이 예제는 지정된 설정으로 새 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", "MyBucketName")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
    "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
    "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"
```

```

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "MyBucketName"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline

```

**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 참조를 참조하십시오 [CreatePipeline](#).

**Remove-CPCustomActionType**

다음 코드 예시에서는 Remove-CPCustomActionType을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

예 1: 이 예제는 지정된 사용자 지정 작업을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인 하라는 메시지가 표시됩니다. 프롬프트 없이 사용자 지정 동작을 삭제하려면 -Force 매개 변수를 추가합니다.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteCustomActionType](#).

## Remove-CPPipeline

다음 코드 예시에서는 Remove-CPPipeline을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 지정된 파이프라인을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 프롬프트 없이 파이프라인을 삭제하려면 -Force 매개 변수를 추가합니다.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePipeline](#).

## Start-CPPipelineExecution

다음 코드 예시에서는 Start-CPPipelineExecution을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 지정된 파이프라인 실행을 시작합니다.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StartPipelineExecution](#).

## Update-CPPipeline

다음 코드 예시에서는 Update-CPPipeline을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예제 1: 이 예제는 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", "MyBucketName")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
  "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
  "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "MyBucketName"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline

```

**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 참조를 참조하십시오 [UpdatePipeline](#).

## 다음 도구를 사용한 Amazon Cognito 자격 증명 예제 PowerShell

다음 코드 예제는 Amazon Cognito Identity와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### Get-CGIIIdentityPool

다음 코드 예시에서는 Get-CGIIIdentityPool을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: ID를 기준으로 특정 자격 증명 풀에 대한 정보를 검색합니다.

```
Get-CGIIIdentityPool -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 참조를 참조하십시오 [DescribeIdentityPool](#).

## Get-CGIIdentityPoolList

다음 코드 예시에서는 Get-CGIIdentityPoolList을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

- 예 1: 기존 자격 증명 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 참조를 참조하십시오 [ListIdentityPools](#).

## Get-CGIIdentityPoolRole

다음 코드 예시에서는 Get-CGIIdentityPoolRole을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

- 예 1: 특정 자격 증명 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 참조를 참조하십시오 [GetIdentityPoolRoles](#).

**New-CGIIIdentityPool**

다음 코드 예시에서는 New-CGIIIdentityPool을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 인증되지 않은 ID를 허용하는 새 자격 증명 풀을 생성합니다.

```

New-CGIIIdentityPool -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 Cmdlet 참조를 참조하십시오 [CreateIdentityPool](#).AWS Tools for PowerShell

**Remove-CGIIIdentityPool**

다음 코드 예시에서는 Remove-CGIIIdentityPool을 사용하는 방법을 보여 줍니다.

## 예 1: 특정 자격 증명 풀을 삭제합니다.

예 1: 특정 자격 증명 풀을 삭제합니다.

```
Remove-CGIIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteIdentityPool](#).

## Set-CGIIIdentityPoolRole

다음 코드 예시에서는 Set-CGIIIdentityPoolRole을 사용하는 방법을 보여 줍니다.

### 예 1: 인증되지 않은 IAM 역할을 갖도록 특정 자격 증명 풀을 구성합니다.

예 1: 인증되지 않은 IAM 역할을 갖도록 특정 자격 증명 풀을 구성합니다.

```
Set-CGIIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [SetIdentityPoolRoles](#) AWS Tools for PowerShell

## Update-CGIIIdentityPool

다음 코드 예시에서는 Update-CGIIIdentityPool을 사용하는 방법을 보여 줍니다.

### 예 1: 일부 자격 증명 풀 속성 (이 경우에는 자격 증명 풀의 이름) 을 업데이트합니다.

예 1: 일부 자격 증명 풀 속성 (이 경우에는 자격 증명 풀의 이름) 을 업데이트합니다.

```
Update-CGIIIdentityPool -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 참조를 참조하십시오 [UpdateIdentityPool](#).

## AWS Config 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS Config. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Add-CFGResourceTag

다음 코드 예시에서는 Add-CFGResourceTag를 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예에서는 지정된 태그를 리소스 ARN에 연결합니다. 이 경우에는 config-rule/config-rule-16iyn0입니다.

```

Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}

```

- API [TagResource](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Get-CFGAggregateComplianceByConfigRuleList

다음 코드 예시에서는 Get-CFGAggregateComplianceByConfigRuleList을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 구성 규칙에 대한 ConfigurationAggregator 'kaju' 필터링에서 세부 정보를 가져와 규칙의 'Compliance'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
```

예 2: 이 예제는 주어진 ConfigurationAggregator 데이터에서 세부 정보를 가져와 애그리게이터에서 다루는 모든 지역의 해당 계정에 대해 필터링한 다음 모든 규칙의 규정 준수를 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 Cmdlet 참조를 참조하십시오.

[DescribeAggregateComplianceByConfigRules](#) AWS Tools for PowerShell

## Get-CFGAggregateComplianceDetailsByConfigRule

다음 코드 예시에서는 Get-CFGAggregateComplianceDetailsByConfigRule을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 계정, 애그리게이터, 지역 및 구성 규칙에 대해 'COMPLIANT' 상태인 AWS Config 규칙 desired-instance-type '에 대한 리소스 ID 및 리소스 유형의 출력을 선택하여 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 Cmdlet 참조를 참조하십시오.

[GetAggregateComplianceDetailsByConfigRule](#) AWS Tools for PowerShell

## Get-CFGAggregateConfigRuleComplianceSummary

다음 코드 예시에서는 Get-CFGAggregateConfigRuleComplianceSummary을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 애그리게이터에 대해 규정을 준수하지 않는 규칙의 수를 반환합니다.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
  raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

출력:

```
CapExceeded CappedCount
-----
False      5
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오  
[GetAggregateConfigRuleComplianceSummary](#).AWS Tools for PowerShell

## Get-CFGAggregateDiscoveredResourceCount

다음 코드 예시에서는 Get-CFGAggregateDiscoveredResourceCount을 사용하는 방법을 보여줍니다.

예에 대한 도구 PowerShell

- 예 1: 이 예제는 us-east-1 지역에 대해 필터링된 지정된 애그리게이터의 리소스 수를 반환합니다.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

출력:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
-----
{} 455
```

- 예 2: 이 예제는 지정된 애그리게이터의 필터링된 영역에 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 Cmdlet 참조를 참조하십시오. [GetAggregateDiscoveredResourceCountsAWS Tools for PowerShell](#)

## Get-CFGAggregateDiscoveredResourceList

다음 코드 예시에서는 Get-CFGAggregateDiscoveredResourceList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 '아일랜드' 애그리게이터에서 집계된 지정된 리소스 유형의 리소스 식별자를 반환합니다. 리소스 유형 목록은 [https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/T.html&toCID=Amazon\\_\\_](https://docs.aws.amazon.com/sdkfornet/v3/apidocs/index.html?page=ConfigService/T.html&toCID=Amazon__)을 확인하십시오. ConfigServiceResourceType ConfigService ResourceType

```
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

```

예 2: 이 예제는 us-east-1 영역으로 필터링된 지정된 애그리게이터에 대해 'default'라는 **AwsEC2SecurityGroup** 이름의 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 Cmdlet 참조를 참조하십시오. [ListAggregateDiscoveredResources](#) AWS Tools for PowerShell

## Get-CFGAggregateResourceConfig

다음 코드 예시에서는 Get-CFGAggregateResourceConfig을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 집계된 해당 리소스의 구성 항목을 반환하고 구성을 확장합니다.

예 1: 이 예제는 집계된 해당 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 Cmdlet 참조의 [GetAggregateResourceconfig-service](#)를 AWS Tools for PowerShell 참조하십시오.

## Get-CFGAggregateResourceConfigBatch

다음 코드 예시에서는 Get-CFGAggregateResourceConfigBatch을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 지정된 애그리게이터에 있는 리소스 (식별됨) 의 현재 구성 항목을 가져옵니다.

예 1: 이 예제는 지정된 애그리게이터에 있는 리소스 (식별됨) 의 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}

```

- API에 대한 세부 정보는 Cmdlet 참조의 [BatchGetAggregateResourceconfig-service](#)를 참조하십시오. AWS Tools for PowerShell

## Get-CFGAggregationAuthorizationList

다음 코드 예시에서는 Get-CFGAggregationAuthorizationList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 애그리게이터에게 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 Cmdlet 참조를 참조하십시오. [DescribeAggregationAuthorizations](#) AWS Tools for PowerShell

## Get-CFGComplianceByConfigRule

다음 코드 예시에서는 Get-CFGComplianceByConfigRule을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 규칙에 대한 규정 준수 세부 정보를 검색합니다. 규칙에 ebs-optimized-instance에 대한 최신 평가 결과가 없으므로 IMPUCIENT\_DATA를 반환합니다.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

출력:

```
ComplianceContributorCount ComplianceType
-----
INSUFFICIENT_DATA
```

예 2: 이 예제는 ALB\_HTTP\_TO\_HTTPS\_REDIRECTION\_CHECK 규칙에 대해 규정을 준수하지 않는 리소스 수를 반환합니다.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

출력:

```
CapExceeded CappedCount
-----
False      2
```

- API [DescribeComplianceByConfigRule](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Get-CFGComplianceByResource

다음 코드 예시에서는 Get-CFGComplianceByResource을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 **AWS::SSM::ManagedInstanceInventory** 리소스 유형에서 'COMPLIANT' 규정 준수 유형을 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 참조를 참조하십시오 [DescribeComplianceByResource](#).

## Get-CFGComplianceDetailsByConfigRule

다음 코드 예시에서는 Get-CFGComplianceDetailsByConfigRule을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 도구 PowerShell

예 1: 이 예제는 규칙에 대한 평가 결과를 access-keys-rotated 가져와 규정 준수 유형별로 그룹화된 결과를 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...
```

예 2: 이 예에서는 COMPLIANT 리소스에 대한 규칙의 규정 준수 세부 정보를 쿼리합니다. access-keys-rotated

```
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 참조를 참조하십시오 [GetComplianceDetailsByConfigRule](#).

## Get-CFGComplianceDetailsByResource

다음 코드 예시에서는 Get-CFGComplianceDetailsByResource을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예시는 해당 리소스에 대한 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 참조를 참조하십시오 [GetComplianceDetailsByResource](#).

## Get-CFGComplianceSummaryByConfigRule

다음 코드 예시에서는 Get-CFGComplianceSummaryByConfigRule을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 규정을 준수하지 않는 Config 규칙의 수를 반환합니다.

```
Get-CFGComplianceSummaryByConfigRule -Select
ComplianceSummary.NonCompliantResourceCount
```

출력:

```
CapExceeded CappedCount
-----
False      9
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [GetComplianceSummaryByConfigRule.AWS Tools for PowerShell](#)

## Get-CFGComplianceSummaryByResourceType

다음 코드 예시에서는 Get-CFGComplianceSummaryByResourceType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예제 1: 이 샘플은 규정 준수 또는 비준수 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 Cmdlet 참조를 참조하십시오. [GetComplianceSummaryByResourceType](#) AWS Tools for PowerShell

## Get-CFGConfigRule

다음 코드 예시에서는 Get-CFGConfigRule을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 선택한 속성과 함께 계정의 구성 규칙을 나열합니다.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
ConfigRuleState
```

출력:

ConfigRuleName	ConfigRuleId	ConfigRuleArn
-----	-----	-----
ALB_REDIRECTION_CHECK	config-rule-12iyn3	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3	ACTIVE	
access-keys-rotated	config-rule-aospfr	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr	ACTIVE	
autoscaling-group-elb-healthcheck-required	config-rule-cn1f2x	arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x	ACTIVE	

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeConfigRules](#).

## Get-CFGConfigRuleEvaluationStatus

다음 코드 예시에서는 Get-CFGConfigRuleEvaluationStatus을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지정된 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 참조를 참조하십시오 [DescribeConfigRuleEvaluationStatus](#).

**Get-CFGConfigurationAggregatorList**

다음 코드 예시에서는 Get-CFGConfigurationAggregatorList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지역/계정의 모든 애그리게이터를 반환합니다.

```
Get-CFGConfigurationAggregatorList
```

**출력:**

```

AccountAggregationSources      :
  {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn     : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabcalme
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 Cmdlet 참조를 참조하십시오 [DescribeConfigurationAggregators](#). AWS Tools for PowerShell

## Get-CFGConfigurationAggregatorSourcesStatus

다음 코드 예시에서는 Get-CFGConfigurationAggregatorSourcesStatus을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 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 참조를 참조하십시오 [DescribeConfigurationAggregatorSourcesStatus](#).

## Get-CFGConfigurationRecorder

다음 코드 예시에서는 Get-CFGConfigurationRecorder를 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제는 컨피그레이션 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 참조를 참조하십시오 [DescribeConfigurationRecorders](#).

## Get-CFGConfigurationRecorderStatus

다음 코드 예시에서는 Get-CFGConfigurationRecorderStatus를 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 샘플은 컨피그레이션 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 참조를 참조하십시오 [DescribeConfigurationRecorderStatus](#).

## Get-CFGConformancePack

다음 코드 예시에서는 Get-CFGConformancePack을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 샘플은 모든 적합성 팩을 나열합니다.

```
Get-CFGConformancePack
```

출력:

```
ConformancePackArn      : arn:aws:config:eu-west-1:123456789012:conformance-
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 Cmdlet 참조를 참조하십시오 [DescribeConformancePacks](#).AWS Tools for PowerShell

## Get-CFGDeliveryChannel

다음 코드 예시에서는 Get-CFGDeliveryChannel을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예시에서는 해당 지역의 배송 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 참조를 참조하십시오 [DescribeDeliveryChannels](#).

## Get-CFGResourceTag

다음 코드 예시에서는 Get-CFGResourceTag을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제에는 해당 리소스의 관련 태그가 나열되어 있습니다.

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

출력:

Key	Value
---	-----
Version	1.3

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTagsForResource](#).

## Remove-CFGConformancePack

다음 코드 예시에서는 Remove-CFGConformancePack을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 해당 팩에 대한 모든 규칙, 수정 조치 및 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 Cmdlet 참조를 참조하십시오 [DeleteConformancePack](#). AWS Tools for PowerShell

**Write-CFGConformancePack**

다음 코드 예시에서는 Write-CFGConformancePack을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 샘플은 지정된 yaml 파일에서 템플릿을 가져와 적합성 팩을 만듭니다.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket kt-ps-examples
-TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [PutConformancePack](#) AWS Tools for PowerShell

**Write-CFGDeliveryChannel**

다음 코드 예시에서는 Write-CFGDeliveryChannel을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예시는 기존 전송 채널의 전송 빈도 속성을 변경합니다.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName config-
bucket-NA -DeliveryChannel_S3KeyPrefix my
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutDeliveryChannel](#).

## 다음 도구를 사용한 Device Farm 예제 PowerShell

다음 코드 예제는 with Device Farm을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### New-DFUpload

다음 코드 예시에서는 New-DFUpload을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예시에서는 Android 앱을 위한 AWS Device Farm 업로드를 생성합니다. 새 DFProject 또는 Get-DF의 출력에서 프로젝트 ARN을 가져올 수 있습니다. ProjectList New-DFUpload 출력의 서명된 URL을 사용하여 Device Farm에 파일을 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 Cmdlet 참조를 참조하십시오. [CreateUpload](#) AWS Tools for PowerShell

## AWS Directory Service 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS Directory Service. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

### 작업

#### Add-DSIpRoute

다음 코드 예시에서는 Add-DSIpRoute을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 ID에 할당된 리소스 태그를 제거합니다.

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -UpdateSecurityGroupForDirectoryController $true
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AddIpRoutes](#).AWS Tools for PowerShell

#### Add-DSResourceTag

다음 코드 예시에서는 Add-DSResourceTag을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 ID에 리소스 태그를 추가합니다.

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AddTagsToResource](#). AWS Tools for PowerShell

## Approve-DSTrust

다음 코드 예시에서는 Approve-DSTrust을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 Trustid에 대한 AWS 디렉터리 서비스 VerifyTrust API 작업을 호출합니다.

```
Approve-DSTrust -TrustId t-9067157123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [VerifyTrust](#).

## Confirm-DSSharedDirectory

다음 코드 예시에서는 Confirm-DSSharedDirectory을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 디렉터리 소유자가 보낸 디렉터리 공유 요청을 AWS 계정수락합니다.

```
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 참조를 참조하십시오 [AcceptSharedDirectory](#).

## Connect-DSDirectory

다음 코드 예시에서는 Connect-DSDirectory을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 온-프레미스 디렉터리에 연결할 AD Connector를 만듭니다.

```
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 참조를 참조하십시오 [ConnectDirectory](#).

## Deny-DSSharedDirectory

다음 코드 예시에서는 Deny-DSSharedDirectory을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 디렉터리 소유자 계정에서 보낸 디렉터리 공유 요청을 거부합니다.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

출력:

```
d-9067012345
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RejectSharedDirectory](#).

## Disable-DSDirectoryShare

다음 코드 예시에서는 Disable-DSDirectoryShare을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 디렉터리 소유자와 소비자 계정 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 참조를 참조하십시오 [UnshareDirectory](#).

## Disable-DSLADAPS

다음 코드 예시에서는 Disable-DSLADAPS을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 디렉터리에 대한 LDAP 보안 호출을 비활성화합니다.

```
Disable-DSLADAPS -DirectoryId d-123456ijkl -Type Client
```

- API에 대한 세부 정보는 Cmdlet 참조의 [DisableADAPS](#)를 참조하십시오. AWS Tools for PowerShell

## Disable-DSRadius

다음 코드 예시에서는 Disable-DSRadius을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 AD Connector 또는 Microsoft AD 디렉터리에 대해 구성된 RADIUS 서버를 비활성화합니다.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisableRadius](#).

## Disable-DSSso

다음 코드 예시에서는 Disable-DSSso을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 디렉터리에 대한 싱글 사인온을 비활성화합니다.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DisableSso](#).AWS Tools for PowerShell

## Enable-DSDirectoryShare

다음 코드 예시에서는 Enable-DSDirectoryShare을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 핸드셰이크 방법을 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 참조를 참조하십시오 [ShareDirectory](#).

## Enable-DSLdapS

다음 코드 예시에서는 Enable-DSLdapS을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 항상 LDAP 보안 호출을 사용하도록 특정 디렉터리의 스위치를 활성화합니다.

```
Enable-DSLdapS -DirectoryId d-123456ijkl -Type Client
```

- API에 대한 자세한 내용은 Cmdlet 참조의 [EnableDAPS 사용](#)을 참조하십시오.AWS Tools for PowerShell

## Enable-DSRadius

다음 코드 예시에서는 Enable-DSRadius을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 AD Connector 또는 Microsoft AD 디렉터리에 대해 제공된 RADIUS 서버 구성을 사용하여 멀티 팩터 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableRadius](#).

## Enable-DSSso

다음 코드 예시에서는 Enable-DSSso을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예시에서는 디렉터리에 SSO (Single Sign-On) 를 활성화합니다.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableSso](#).

## Get-DSCertificate

다음 코드 예시에서는 Get-DSCertificate을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 보안 LDAP 연결에 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 참조를 참조하십시오 [DescribeCertificate](#).

**Get-DSCertificateList**

다음 코드 예시에서는 Get-DSCertificateList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 디렉터리의 보안 LDAP 연결에 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 참조를 참조하십시오 [ListCertificates](#).

**Get-DSConditionalForwarder**

다음 코드 예시에서는 Get-DSConditionalForwarder을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 ID의 구성된 모든 조건부 전달자를 가져옵니다.

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

출력:

```
DnsIpAddr      RemoteDomainName ReplicationScope
-----
{172.31.77.239} contoso.com      Domain
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DescribeConditionalForwarders](#).AWS Tools for PowerShell

## Get-DSDirectory

다음 코드 예시에서는 Get-DSDirectory을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 이 계정에 속한 디렉터리에 대한 정보를 가져옵니다.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddr, Type
```

출력:

```
DirectoryId  Name                DnsIpAddr                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 참조를 참조하십시오 [DescribeDirectories](#).

## Get-DSDirectoryLimit

다음 코드 예시에서는 Get-DSDirectoryLimit을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 us-east-1 지역의 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 Cmdlet 참조를 참조하십시오. [GetDirectoryLimits](#) AWS Tools for PowerShell

## Get-DSDomainControllerList

다음 코드 예시에서는 Get-DSDomainControllerList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 명령은 언급된 디렉터리 ID에 대해 시작된 도메인 컨트롤러의 세부 목록을 가져옵니다.

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

출력:

```

AvailabilityZone      : us-east-1b
DirectoryId          : d-123456ijkl
DnsIpAddress         : 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
DnsIpAddress         : 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 Cmdlet 참조를 참조하십시오 [DescribeDomainControllers](#).AWS Tools for PowerShell

## Get-DSEventTopic

다음 코드 예시에서는 Get-DSEventTopic을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 명령은 디렉터리 상태가 변경될 때 알림을 받도록 구성된 SNS 주제의 정보를 표시합니다.

```
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 참조를 참조하십시오 [DescribeEventTopics](#).

## Get-DSIpRouteList

다음 코드 예시에서는 Get-DSIpRouteList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 명령은 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-123456ijk1
IpRouteStatusMsg : Added
IpRouteStatusReason :
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListIpRoutes](#).

## Get-DSLdapSetting

다음 코드 예시에서는 Get-DSLdapSetting을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예에서는 지정된 디렉터리의 LDAP 보안 상태를 설명합니다.

```
Get-DSLdapSetting -DirectoryId d-123456ijk1
```

출력:

```
LastUpdatedDateTime  LDAPSStatus  LDAPSStatusReason
-----
4/15/2020 6:51:03 PM Enabled      LDAPS is enabled successfully.
```

- API에 대한 세부 정보는 Cmdlet 참조의 [DescribeDAPSettings](#)를 참조하십시오. AWS Tools for PowerShell

## Get-DSLogSubscriptionList

다음 코드 예시에서는 Get-DSLogSubscriptionList을 사용하는 방법을 보여 줍니다.

에 대한 도구: PowerShell

- 예 1: 이 명령은 지정된 디렉토리 ID의 로그 구독 정보를 가져옵니다.

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijk1
```

출력:

```
DirectoryId  LogGroupName
SubscriptionCreatedDateTime
```

```

-----
-----
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
AM

```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ListLogSubscriptions](#).AWS Tools for PowerShell

## Get-DSResourceTag

다음 코드 예시에서는 Get-DSResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 명령은 지정된 디렉터리의 모든 태그를 가져옵니다.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

출력:

```

Key      Value
---      -
myTag    myTagValue

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTagsForResource](#).

## Get-DSSchemaExtension

다음 코드 예시에서는 Get-DSSchemaExtension을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예에서는 Microsoft AD 디렉터리에 적용된 모든 스키마 확장을 나열합니다.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

출력:

```

Description          : ManagedADSchemaExtension
DirectoryId          : d-123456ijkl
EndTime              : 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

```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListSchemaExtensions](#).

## Get-DSSharedDirectory

다음 코드 예시에서는 Get-DSSharedDirectory을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예시는 계정의 공유 디렉터리를 가져옵니다. AWS

```
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 참조를 참조하십시오 [DescribeSharedDirectories](#).

## Get-DSSnapshot

다음 코드 예시에서는 Get-DSSnapshot을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 명령은 이 계정에 속하는 지정된 디렉터리 스냅샷에 대한 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 참조를 참조하십시오 [DescribeSnapshots](#).

## Get-DSSnapshotLimit

다음 코드 예시에서는 Get-DSSnapshotLimit을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 명령은 지정된 디렉터리의 수동 스냅샷 한도를 가져옵니다.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

출력:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
-----
0                            5                            False
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetSnapshotLimits](#).

## Get-DSTrust

다음 코드 예시에서는 Get-DSTrust을 사용하는 방법을 보여 줍니다.

## 예 1: 이 명령은 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 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 Cmdlet 참조를 참조하십시오 [DescribeTrusts](#).AWS Tools for PowerShell

## New-DSAlias

다음 코드 예시에서는 New-DSAlias을 사용하는 방법을 보여 줍니다.

## 예 1: 이 명령은 도구 PowerShell

예 1: 이 명령은 디렉터리의 별칭을 만들고 지정된 디렉터리 ID에 별칭을 할당합니다.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

출력:

```
Alias      DirectoryId
-----      -
myorgname d-123456ijkl
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [CreateAlias](#)AWS Tools for PowerShell

## New-DSComputer

다음 코드 예시에서는 New-DSComputer을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 새 Active Directory 컴퓨터 개체를 만듭니다.

예 1: 이 예제에서는 새 Active Directory 컴퓨터 개체를 만듭니다.

```
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 [CreateComputer](#) 참조의 내용을 참조하십시오.

## New-DSConditionalForwarder

다음 코드 예시에서는 New-DSConditionalForwarder을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 AWS 디렉터리 ID에 조건부 전달자를 만듭니다.

예 1: 이 예에서는 지정된 AWS 디렉터리 ID에 조건부 전달자를 만듭니다.

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress
172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CreateConditionalForwarder](#).AWS Tools for PowerShell

## New-DSDirectory

다음 코드 예시에서는 New-DSDirectory을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 새 Simple AD 디렉터리를 만듭니다.

```
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 참조를 참조하십시오 [CreateDirectory](#).

## New-DSLogSubscription

다음 코드 예시에서는 New-DSLogSubscription을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 실시간 Directory Service 도메인 컨트롤러 보안 로그를 사용자의 지정된 Amazon CloudWatch 로그 그룹에 전달하는 구독을 생성합니다 AWS 계정.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/  
d-123456ijkl-lan2.example.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateLogSubscription](#).

## New-DSMicrosoftAD

다음 코드 예시에서는 New-DSMicrosoftAD을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 새 Microsoft AD 디렉터리를 만듭니다 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 참조의 [CreateMicrosoftAD](#)를 참조하십시오.

## New-DSSnapshot

다음 코드 예시에서는 New-DSSnapshot을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 디렉터리 PowerShell

예 1: 이 예에서는 디렉터리 스냅샷을 생성합니다.

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateSnapshot](#).

## New-DSTrust

다음 코드 예시에서는 New-DSTrust을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 디렉터리 PowerShell

예 1: 이 예에서는 관리형 AWS Microsoft AD 디렉터리와 기존 온-프레미스 Microsoft Active Directory 간에 포리스트 전체에 대한 양방향 트러스트를 만듭니다.

```
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 Cmdlet 참조의 내용을 참조하십시오 [CreateTrust](#).AWS Tools for PowerShell

## Register-DSCertificate

다음 코드 예시에서는 Register-DSCertificate을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 디렉터리 PowerShell

예 1: 이 예에서는 보안 LDAP 연결을 위한 인증서를 등록합니다.

```
$Certificate = Get-Content contoso.cer -Raw  
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

출력:

```
c-906731e350
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [RegisterCertificate](#).AWS Tools for PowerShell

## Register-DSEventTopic

다음 코드 예시에서는 Register-DSEventTopic을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 디렉터리를 게시자로서 SNS 주제와 연결합니다.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterEventTopic](#).

## Remove-DSConditionalForwarder

다음 코드 예시에서는 Remove-DSConditionalForwarder을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 AWS 디렉터리에 설정된 조건부 전달자를 제거합니다.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName  
contoso.com
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeleteConditionalForwarder](#).AWS Tools for PowerShell

## Remove-DSDirectory

다음 코드 예시에서는 Remove-DSDirectory을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 AWS 디렉터리 서비스 디렉터리 (단순 AD/Microsoft AD/AD 커넥터) 를 삭제합니다.

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [DeleteDirectory](#).AWS Tools for PowerShell

## Remove-DSIpRoute

다음 코드 예시에서는 Remove-DSIpRoute을 사용하는 방법을 보여 줍니다.

예 1: 이 명령은 디렉터리 ID에 대한 도구 PowerShell

예 1: 이 명령은 디렉터리 ID의 구성된 IP 경로에서 지정된 IP를 제거합니다.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [RemoveIpRoutes](#).AWS Tools for PowerShell

## Remove-DSLogSubscription

다음 코드 예시에서는 Remove-DSLogSubscription을 사용하는 방법을 보여 줍니다.

예 1: 이 명령은 지정된 디렉터리 ID에 대한 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 ID의 로그 구독을 제거합니다.

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteLogSubscription](#).AWS Tools for PowerShell

## Remove-DSResourceTag

다음 코드 예시에서는 Remove-DSResourceTag을 사용하는 방법을 보여 줍니다.

예 1: 이 명령은 지정된 디렉터리 ID에 대한 도구 PowerShell

예 1: 이 명령은 지정된 디렉터리 ID에 할당된 리소스 태그를 제거합니다.

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [RemoveTagsFromResource](#).AWS Tools for PowerShell

## Remove-DSSnapshot

다음 코드 예시에서는 Remove-DSSnapshot을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 수동으로 생성한 스냅샷을 제거합니다.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteSnapshot](#).

## Remove-DSTrust

다음 코드 예시에서는 Remove-DSTrust을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 AWS 관리형 AD 디렉터리와 외부 도메인 간의 기존 신뢰 관계를 제거합니다.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

출력:

```
t-9067157123
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteTrust](#).AWS Tools for PowerShell

## Reset-DSUserPassword

다음 코드 예시에서는 Reset-DSUserPassword을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 AWS 관리형 Microsoft AD 또는 Simple AD Directory에서 ADUser라는 Active Directory 사용자의 비밀번호를 재설정합니다.

```
Reset-DSUserPassword -UserName ADUser -DirectoryId d-123456ijkl -NewPassword $Password
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [ResetUserPassword](#).AWS Tools for PowerShell

## Restore-DSFromSnapshot

다음 코드 예시에서는 Restore-DSFromSnapshot을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 기존 디렉터리 스냅샷을 사용하여 디렉터리를 복원합니다.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [RestoreFromSnapshot](#).AWS Tools for PowerShell

## Set-DSDomainControllerCount

다음 코드 예시에서는 Set-DSDomainControllerCount을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 디렉터리 ID의 도메인 컨트롤러 수를 3으로 설정합니다.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [UpdateNumberOfDomainControllers](#).AWS Tools for PowerShell

## Start-DSSchemaExtension

다음 코드 예시에서는 Start-DSSchemaExtension을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StartSchemaExtension](#).

## Stop-DSSchemaExtension

다음 코드 예시에서는 Stop-DSSchemaExtension을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 Microsoft AD 디렉터리에 대한 진행 중인 스키마 확장을 취소합니다.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CancelSchemaExtension](#).

## Unregister-DSCertificate

다음 코드 예시에서는 Unregister-DSCertificate을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 보안 LDAP 연결에 등록된 인증서를 시스템에서 삭제합니다.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeregisterCertificate](#). AWS Tools for PowerShell

## Unregister-DSEventTopic

다음 코드 예시에서는 Unregister-DSEventTopic을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 SNS 주제에 대한 게시자로서 지정된 디렉토리를 제거합니다.

예 1: 이 예에서는 지정된 SNS 주제에 대한 게시자로서 지정된 디렉토리를 제거합니다.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeregisterEventTopic](#). AWS Tools for PowerShell

## Update-DSConditionalForwarder

다음 코드 예시에서는 Update-DSConditionalForwarder을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제는 AWS 디렉터리에 설정된 조건부 전달자를 업데이트합니다.

예 1: 이 예제는 AWS 디렉터리에 설정된 조건부 전달자를 업데이트합니다.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddress 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateConditionalForwarder](#).

## Update-DSRadius

다음 코드 예시에서는 Update-DSRadius을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예에서는 AD Connector 또는 Microsoft AD 디렉터리에 대한 RADIUS 서버 정보를 업데이트합니다.

예 1: 이 예에서는 AD Connector 또는 Microsoft AD 디렉터리에 대한 RADIUS 서버 정보를 업데이트합니다.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet [UpdateRadius](#) 참조의 내용을 참조하십시오.

## Update-DSTrust

다음 코드 예시에서는 Update-DSTrust을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 trust-id의 SelectiveAuth 매개변수를 비활성화됨에서 활성화로 업데이트합니다.

예 1: 이 예에서는 지정된 trust-id의 SelectiveAuth 매개변수를 비활성화됨에서 활성화로 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 참조를 참조하십시오 [UpdateTrust](#).

## AWS DMS 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS DMS. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### New-DMSReplicationTask

다음 코드 예시에서는 New-DMSReplicationTask을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예 1: 이 예에서는 CdcStartTime 대신 을 사용하는 새 AWS Database Migration Service 복제 작업을 생성합니다 CdcStartPosition. MigrationType 는 "full-load-and-cdc"로 설정되어 있습니다. 즉, 대상 테이블은 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 만들기 ([https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Tasks.Creating.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html)) 를 참조하십시오.

```
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"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateReplicationTask](#).

## 다음과 같은 도구를 사용한 DynamoDB 예제 PowerShell

다음 코드 예제는 DynamoDB와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 GitHub 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다.

## 주제

- [작업](#)

## 작업

**Add-DDBIndexSchema**

다음 코드 예시에서는 Add-DDBIndexSchema을 사용하는 방법을 보여 줍니다.

## 예 1: 빈 TableSchema 객체에 대한 도구 PowerShell

예 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}	

예 2: 객체를 파이프라인에 다시 쓰기 전에 제공된 TableSchema 객체에 새 로컬 보조 인덱스 정의를 추가합니다. TableSchema -Schema 매개 변수를 사용하여 TableSchema 객체를 제공할 수도 있습니다.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
"LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

## 출력:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----

```
{LastPostDateTime}
    {LastPostIndex}
```

- API에 대한 자세한 내용은 Cmdlet IndexSchema 참조의 [AWS Tools for PowerShell Add-DDB를](#) 참조하십시오.

## Add-DDBKeySchema

다음 코드 예시에서는 Add-DDBKeySchema을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 빈 TableSchema 객체를 생성하고 파이프라인에 객체를 쓰기 전에 지정된 키 데이터를 사용하여 빈 TableSchema 객체에 키와 속성 정의 항목을 추가합니다. 키 유형은 기본적으로 'HASH'로 선언됩니다. 범위 키를 선언하려면 값이 'RANGE'인 - KeyType 매개변수를 사용하십시오.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

출력:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	
-----	-----
-----	
{ForumName}	{ForumName}
{}	

예 2: 파이프라인에 객체를 쓰기 전에 제공된 TableSchema 객체에 새 키와 속성 정의 항목을 추가합니다. TableSchema 키 유형은 기본적으로 'HASH'로 선언됩니다. 범위 키를 선언하려면 값이 'RANGE'인 - KeyType 매개변수를 사용하십시오. -Schema 매개 변수를 사용하여 TableSchema 객체를 제공할 수도 있습니다.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

출력:

AttributeSchema	KeySchema
LocalSecondaryIndexSchema	

```

-----
-----
{ForumName}           {ForumName}
  {}

```

- API에 대한 자세한 내용은 Cmdlet KeySchema 참조의 [AWS Tools for PowerShell Add-DDB를 참조하십시오.](#)

## ConvertFrom-DDBItem

다음 코드 예시에서는 ConvertFrom-DDBItem을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: ConvertFrom -DDBItem은 Get-DDBItem의 결과를 AttributeValues DynamoDB 해시 테이블에서 문자열 및 더블과 같은 일반적인 형식의 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 ConvertFrom AWS Tools for PowerShell Cmdlet 참조의 -DDBItem을 참조하십시오.](#)

## ConvertTo-DDBItem

다음 코드 예시에서는 ConvertTo-DDBItem을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예 1: 해시 테이블을 DynamoDB 속성 값 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: 해시 테이블을 DynamoDB 속성 값 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 Cmdlet 참조의 [ConvertTo-ddBItem](#)을 참조하십시오. AWS Tools for PowerShell

## Get-DDBBatchItem

다음 코드 예시에서는 Get-DDBBatchItem을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: DynamoDB 테이블 'Music' 및 SongTitle 'Songs'에서 '어딘가 먼 길'이 있는 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 Cmdlet 참조를 참조하십시오 [BatchGetItem](#).AWS Tools for PowerShell

## Get-DDBItem

다음 코드 예시에서는 Get-DDBItem을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 파티션 SongTitle 키와 정렬 키 아티스트가 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 참조를 참조하십시오 [GetItem](#).

## Get-DDBTable

다음 코드 예시에서는 Get-DDBTable을 사용하는 방법을 보여 줍니다.

## 예 1: 지정된 테이블의 세부 정보를 반환합니다.

예 1: 지정된 테이블의 세부 정보를 반환합니다.

```
Get-DDBTable -TableName "myTable"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTable](#).

## Get-DDBTableList

다음 코드 예시에서는 Get-DDBTableList을 사용하는 방법을 보여 줍니다.

### 예 1: 서비스에 더 이상 테이블이 없다고 표시될 때까지 자동으로 반복하여 모든 테이블의 세부 정보를 반환합니다.

예 1: 서비스에 더 이상 테이블이 없다고 표시될 때까지 자동으로 반복하여 모든 테이블의 세부 정보를 반환합니다.

```
Get-DDBTableList
```

예 2: 서비스에 더 이상 테이블이 없다고 표시될 때까지 수동으로 반복하여 모든 테이블의 세부 정보를 직접 호출당 테이블 최대 10개까지 반환합니다.

```
$nextToken = $null
do {
  Get-DDBTableList -ExclusiveStartTableName $nextToken -Limit 10
  $nextToken = $AWSHistory.LastServiceResponse.LastEvaluatedTableName
} while ($nextToken -ne $null)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTables](#).

## Invoke-DDBQuery

다음 코드 예시에서는 Invoke-DDBQuery을 사용하는 방법을 보여 줍니다.

### 예 1: 지정된 및 아티스트와 함께 DynamoDB 항목을 반환하는 쿼리를 호출합니다. SongTitle

예 1: 지정된 및 아티스트와 함께 DynamoDB 항목을 반환하는 쿼리를 호출합니다. SongTitle

```
$invokeDDBQuery = @{
  TableName = 'Music'
  KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
```

```

    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem

```

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 참조의 [Query](#)를 참조하세요.

## Invoke-DDBScan

다음 코드 예시에서는 Invoke-DDBScan을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

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 CriticRating 9보다 크거나 같은 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 참조의 [Scan](#)을 참조하세요.

## New-DDBTable

다음 코드 예시에서는 New-DDBTable을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 " (키 유형 해시) 및 'SubjectForumName' (키 유형 범위) 로 구성된 기본 키를 가진 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 : {}
```

예제 2: 이 예제에서는 " (키 유형 해시) 및 'SubjectForumName' (키 유형 범위) 로 구성된 기본 키를 가진 Thread라는 테이블을 만듭니다. 로컬 보조 인덱스도 정의됩니다. 로컬 보조 인덱스의 키는 테이블 () ForumName 의 기본 해시 키에서 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}
```

예제 3: 이 예제에서는 단일 파이프라인을 사용하여 " (키 유형 해시) 및 'SubjectForumName' (키 유형 범위) 로 구성된 기본 키와 로컬 보조 인덱스가 있는 Thread라는 테이블을 생성하는 방법을 보

여줍니다. 파이프라인 또는 -Schema 파라미터에서 TableSchema 객체를 제공하지 않는 경우 Add-DDB KeySchema 및 Add-DDB는 새 객체를 자동으로 IndexSchema 생성합니다.

```
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}
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [CreateTable](#) AWS Tools for PowerShell

## New-DDBTableSchema

다음 코드 예시에서는 New-DDBTableSchema을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 새 Amazon DynamoDB 테이블을 생성하는 데 사용할 키 및 인덱스 정의를 받아들이 준 비가 된 빈 TableSchema 객체를 생성합니다. 반환된 객체를 Add-DDBKeySchema, Add-DDB IndexSchema 및 New-DDBTable cmdlet으로 파이프하거나 각 cmdlet의 -Schema 파라미터를 사용하여 해당 cmdlet에 전달할 수 있습니다.

```
New-DDBTableSchema
```

출력:

```

AttributeSchema                                KeySchema
  LocalSecondaryIndexSchema
-----
-----
{}                                              {}
  {}

```

- [TableSchemaAPI에 AWS Tools for PowerShell 대한 세부 정보는 Cmdlet 참조의 New-DDB를 참조하십시오.](#)

## Remove-DDBItem

다음 코드 예시에서는 Remove-DDBItem을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteItem](#).

## Remove-DDBTable

다음 코드 예시에서는 Remove-DDBTable을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteTable](#).

## Set-DDBBatchItem

다음 코드 예시에서는 Set-DDBBatchItem을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

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
```

출력:

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [BatchWriteItem](#).

## Set-DDBItem

다음 코드 예시에서는 Set-DDBItem을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutItem](#).

## Update-DDBItem

다음 코드 예시에서는 Update-DDBItem을 사용하는 방법을 보여 줍니다.

예 1: 파티션 SongTitle 키와 정렬 키 아티스트를 사용하여 DynamoDB 항목의 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 Cmdlet 참조를 참조하십시오 [UpdateItem](#).AWS Tools for PowerShell

## Update-DDBTable

다음 코드 예시에서는 Update-DDBTable을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 주어진 테이블의 프로비저닝된 처리량을 업데이트합니다.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateTable](#).

## 다음 도구를 사용한 Amazon EC2 예제 PowerShell

다음 코드 예제는 Amazon EC2와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

## Add-EC2CapacityReservation

다음 코드 예시에서는 Add-EC2CapacityReservation을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [CreateCapacityReservation](#).

**Add-EC2InternetGateway**

다음 코드 예시에서는 Add-EC2InternetGateway을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 인터넷 게이트웨이를 지정된 VPC에 연결합니다.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

예 2: 이 예제는 VPC와 인터넷 게이트웨이를 만든 다음 인터넷 게이트웨이를 VPC에 연결합니다.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AttachInternetGateway](#).AWS Tools for PowerShell

## Add-EC2NetworkInterface

다음 코드 예시에서는 Add-EC2NetworkInterface을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 네트워크 인터페이스를 지정된 인스턴스에 연결합니다.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -DeviceIndex 1
```

출력:

```
eni-attach-1a2b3c4d
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AttachNetworkInterface](#).

## Add-EC2Volume

다음 코드 예시에서는 Add-EC2Volume을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 볼륨을 지정된 인스턴스에 연결하고 지정된 장치 이름과 함께 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AttachVolume](#).

## Add-EC2VpnGateway

다음 코드 예시에서는 Add-EC2VpnGateway을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [AttachVpnGateway](#).

## Approve-EC2VpcPeeringConnection

다음 코드 예시에서는 Approve-EC2VpcPeeringConnection을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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
```

- AWS Tools for PowerShell API에 대한 [AcceptVpcPeeringConnection](#) 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Confirm-EC2ProductInstance

다음 코드 예시에서는 Confirm-EC2ProductInstance을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 제품 코드가 지정된 인스턴스와 연결되어 있는지 여부를 확인합니다.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ConfirmProductInstance](#).

## Copy-EC2Image

다음 코드 예시에서는 Copy-EC2Image을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 'EU (아일랜드)' 지역의 지정된 AMI를 '미국 서부 (오레곤)' 지역으로 복사합니다. -Region이 지정되지 않은 경우 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 참조를 참조하십시오 [CopyImage](#).

## Copy-EC2Snapshot

다음 코드 예시에서는 Copy-EC2Snapshot을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 스냅샷을 EU (아일랜드) 지역에서 미국 서부 (오레곤) 지역으로 복사합니다.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-west-2
```

예 2: 기본 지역을 설정하고 지역 매개 변수를 생략한 경우 기본 대상 지역이 기본 지역이 됩니다.

```
Set-DefaultAWSRegion us-west-2  
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CopySnapshot](#).

## Deny-EC2VpcPeeringConnection

다음 코드 예시에서는 Deny-EC2VpcPeeringConnection을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 위 예시에서는 요청 ID pcx-01a2b3ce45fe67eb8에 대한 VpcPeering 요청을 거부합니다.

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API에 [RejectVpcPeeringConnection](#) 대한 AWS Tools for PowerShell 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Disable-EC2VgwRoutePropagation

다음 코드 예시에서는 Disable-EC2VgwRoutePropagation을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 VGW가 경로를 지정된 라우팅 테이블에 자동으로 전파하지 못하도록 합니다.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DisableVgwRoutePropagation](#). AWS Tools for PowerShell

## Disable-EC2VpcClassicLink

다음 코드 예시에서는 Disable-EC2VpcClassicLink을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예제 1: 이 예에서는 VpcClassicLink vpc-01e23c4a5d6db78e9에 대해 EC2를 비활성화합니다. True 또는 False를 반환합니다.

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisableVpcClassicLink](#).

## Disable-EC2VpcClassicLinkDnsSupport

다음 코드 예시에서는 Disable-EC2VpcClassicLinkDnsSupport을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예제 1: 이 예에서는 vpc-0b12d3456a7e8910d에 대한 ClassicLink DNS 지원을 비활성화합니다.

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API에 [DisableVpcClassicLinkDnsSupport](#)대한 자세한 AWS Tools for PowerShell 내용은 Cmdlet 참조를 참조하십시오.

## Dismount-EC2InternetGateway

다음 코드 예시에서는 Dismount-EC2InternetGateway을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 이 예제는 지정된 VPC에서 지정된 인터넷 게이트웨이를 분리합니다.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachInternetGateway](#).

## Dismount-EC2NetworkInterface

다음 코드 예시에서는 Dismount-EC2NetworkInterface을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 네트워크 인터페이스와 인스턴스 간의 지정된 연결을 제거합니다.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachNetworkInterface](#).

## Dismount-EC2Volume

다음 코드 예시에서는 Dismount-EC2Volume을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 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
```

예 2: 올바른 볼륨을 분리하고 있는지 확인하기 위해 인스턴스 ID와 디바이스 이름을 지정할 수도 있습니다.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachVolume](#).

## Dismount-EC2VpnGateway

다음 코드 예시에서는 Dismount-EC2VpnGateway을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 VPC에서 지정된 가상 프라이빗 게이트웨이를 분리합니다.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachVpnGateway](#).

## Edit-EC2CapacityReservation

다음 코드 예시에서는 Edit-EC2CapacityReservation을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예제 1: 이 예제에서는 인스턴스 수를 1로 변경하여 CapacityReservationId cr-0c1f2345db6f7cdba 를 수정합니다.

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -  
InstanceCount 1
```

출력:

```
True
```

- AWS Tools for PowerShell API에 [ModifyCapacityReservation](#) 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Edit-EC2Host

다음 코드 예시에서는 Edit-EC2Host을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예에서는 전용 호스트 h-01e23f4cd567890f3의 AutoPlacement 설정을 끄기로 수정합니다.

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

출력:

```

Successful                Unsuccessful
-----
{h-01e23f4cd567890f3} {}

```

- AWS Tools for PowerShell API에 [ModifyHosts](#) 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Edit-EC2IdFormat

다음 코드 예시에서는 Edit-EC2IdFormat을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제는 지정된 리소스 유형에 대해 더 긴 ID 형식을 활성화합니다.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

- 예 2: 이 예제는 지정된 리소스 유형에 대해 더 긴 ID 형식을 비활성화합니다.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyIdFormat](#).

## Edit-EC2ImageAttribute

다음 코드 예시에서는 Edit-EC2ImageAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제는 지정된 AMI에 대한 설명을 업데이트합니다.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

- 예 2: 이 예제는 AMI를 퍼블릭 (예: 누구나 사용할 AWS 계정 수 있도록) 으로 설정합니다.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

- 예 3: 이 예에서는 AMI를 비공개로 설정합니다 (예: 소유자인 사용자만 사용할 수 있음).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

예 4: 이 예제는 지정된 항목에 시작 권한을 부여합니다 AWS 계정.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

예 5: 이 예제는 지정된 권한에서 시작 권한을 제거합니다 AWS 계정.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyImageAttribute](#).

## Edit-EC2InstanceAttribute

다음 코드 예시에서는 Edit-EC2InstanceAttribute을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 인스턴스의 인스턴스 유형을 수정합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

예 2: 이 예제는 단일 루트 I/O 가상화 (SR-IOV) 네트워크 지원 파라미터의 값으로 "simple"을 지정하여 지정된 인스턴스의 향상된 네트워킹을 활성화합니다. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

예 3: 이 예에서는 지정된 인스턴스의 보안 그룹을 수정합니다. 인스턴스는 VPC에 있어야 합니다. 이름이 아닌 각 보안 그룹의 ID를 지정해야 합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
"sg-45678901" )
```

예 4: 이 예에서는 지정된 인스턴스에 대해 EBS I/O 최적화를 활성화합니다. 모든 인스턴스 유형에서 이 기능을 사용할 수 있는 것은 아닙니다. EBS 최적화 인스턴스를 사용할 경우 추가 사용 요금이 적용됩니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

예 5: 이 예제에서는 지정된 인스턴스의 원본/목적지 확인을 활성화합니다. NAT 인스턴스가 NAT를 수행하려면 값이 'false'여야 합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

예 6: 이 예제는 지정된 인스턴스의 종료를 비활성화합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

예 7: 이 예제는 인스턴스에서 종료가 시작될 때 종료되도록 지정된 인스턴스를 변경합니다.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
terminate
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [ModifyInstanceAttribute](#). AWS Tools for PowerShell

## Edit-EC2InstanceCreditSpecification

다음 코드 예시에서는 Edit-EC2InstanceCreditSpecification을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이렇게 하면 i-01234567890abcdef 인스턴스에 T2 무제한 크레딧이 활성화됩니다.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API에 대한 [ModifyInstanceCreditSpecification AWS Tools for PowerShell](#) 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Edit-EC2NetworkInterfaceAttribute

다음 코드 예시에서는 Edit-EC2NetworkInterfaceAttribute을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예에서는 종료 시 지정된 첨부 파일이 삭제되도록 지정된 네트워크 인터페이스를 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

예 2: 이 예에서는 지정된 네트워크 인터페이스의 설명을 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
description"
```

예 3: 이 예에서는 지정된 네트워크 인터페이스의 보안 그룹을 수정합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
sg-1a2b3c4d
```

예 4: 이 예에서는 지정된 네트워크 인터페이스의 원본/대상 검사를 비활성화합니다.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
>false
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ModifyNetworkInterfaceAttribute](#). AWS Tools for PowerShell

## Edit-EC2ReservedInstance

다음 코드 예시에서는 Edit-EC2ReservedInstance을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [ModifyReservedInstances](#).

## Edit-EC2SnapshotAttribute

다음 코드 예시에서는 Edit-EC2SnapshotAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예제에서는 CreateVolumePermission 속성을 설정하여 지정된 스냅샷을 공개합니다.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission -OperationType Add -GroupName all
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifySnapshotAttribute](#).

## Edit-EC2SpotFleetRequest

다음 코드 예시에서는 Edit-EC2SpotFleetRequest을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

- 예 1: 이 예에서는 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 참조를 참조하십시오 [ModifySpotFleetRequest](#).

## Edit-EC2SubnetAttribute

다음 코드 예시에서는 Edit-EC2SubnetAttribute을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 서브넷의 퍼블릭 IP 주소 지정을 활성화합니다.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

예 2: 이 예에서는 지정된 서브넷의 퍼블릭 IP 주소 지정을 비활성화합니다.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ModifySubnetAttribute](#). AWS Tools for PowerShell

## Edit-EC2VolumeAttribute

다음 코드 예시에서는 Edit-EC2VolumeAttribute을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 지정된 볼륨의 지정된 속성을 수정합니다. 데이터가 일치하지 않을 수 있어 일시 중단된 후 볼륨의 I/O 작업이 자동으로 재개됩니다.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyVolumeAttribute](#).

## Edit-EC2VpcAttribute

다음 코드 예시에서는 Edit-EC2VpcAttribute을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 VPC의 DNS 호스트 이름을 지원할 수 있습니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

예 2: 이 예제에서는 지정된 VPC의 DNS 호스트 이름 지원을 비활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

예 3: 이 예제에서는 지정된 VPC에 대한 DNS 확인을 지원할 수 있습니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

예 4: 이 예제에서는 지정된 VPC에 대한 DNS 확인 지원을 비활성화합니다.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyVpcAttribute](#).

## Enable-EC2VgwRoutePropagation

다음 코드 예시에서는 Enable-EC2VgwRoutePropagation을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제를 사용하면 지정된 VGW가 경로를 지정된 라우팅 테이블에 자동으로 전파할 수 있습니다.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [EnableVgwRoutePropagation](#).AWS Tools for PowerShell

## Enable-EC2VolumeIO

다음 코드 예시에서는 Enable-EC2VolumeIO을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [EnableVolumeIO](#).

## Enable-EC2VpcClassicLink

다음 코드 예시에서는 Enable-EC2VpcClassicLink을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예에서는 다음과 같은 VPC vpc-0123456b789b0d12f를 활성화합니다. ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

출력:

```
True
```

- API에 대한 [EnableVpcClassicLink](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Enable-EC2VpcClassicLinkDnsSupport

다음 코드 예시에서는 Enable-EC2VpcClassicLinkDnsSupport을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제를 사용하면 vpc-0b12d3456a7e8910d에서 다음과 같은 DNS 호스트 이름 확인을 지원할 수 있습니다. ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API에 대한 AWS Tools for PowerShell 자세한 내용은 Cmdlet 참조를 참조하십시오. [EnableVpcClassicLinkDnsSupport](#)

## Get-EC2AccountAttribute

다음 코드 예시에서는 Get-EC2AccountAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 해당 지역의 EC2-Classical 및 EC2-VPC 버전으로 인스턴스를 시작할 수 있는지 아니면 EC2-VPC 에서만 시작할 수 있는지를 설명합니다.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

출력:

```
AttributeValue
-----
EC2
VPC
```

예 2: 이 예제는 기본 VPC를 설명하며, 해당 지역에 기본 VPC가 없는 경우에는 '없음'으로 표시됩니다.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

출력:

```
AttributeValue
-----
vpc-12345678
```

예 3: 이 예제에서는 실행할 수 있는 온디맨드 인스턴스의 최대 수를 설명합니다.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

출력:

```
AttributeValue
-----
20
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAccountAttributes](#).

## Get-EC2Address

다음 코드 예시에서는 Get-EC2Address을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 EC2-Classic의 인스턴스에 지정된 엘라스틱 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
```

예 2: 이 예제에서는 VPC의 인스턴스에 대한 엘라스틱 IP 주소를 설명합니다. 이 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

예 3: 이 예에서는 EC2-Classic의 인스턴스에 지정된 엘라스틱 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
```

예 4: 이 예제에서는 EC2-Classic의 인스턴스에 대한 엘라스틱 IP 주소를 설명합니다. 이 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

예 5: 이 예에서는 모든 엘라스틱 IP 주소를 설명합니다.

```
Get-EC2Address
```

예 6: 이 예제는 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 예제는 할당 ID, 연결 ID 및 인스턴스 ID와 함께 모든 엘라스틱 IP를 검색합니다.

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId, AllocationId, PublicIp
```

출력:

```
InstanceId          AssociationId        AllocationId        PublicIp
-----
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
```

예 8: 이 예제는 값이 'Prod'인 태그 키 '카테고리'와 일치하는 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 Cmdlet 참조를 참조하십시오 [DescribeAddresses](#).AWS Tools for PowerShell

**Get-EC2AvailabilityZone**

다음 코드 예시에서는 Get-EC2AvailabilityZone을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

예 1: 이 예에서는 현재 지역에서 사용할 수 있는 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

예 2: 이 예에서는 장애가 발생한 모든 가용 영역을 설명합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

예 3: PowerShell 버전 2에서는 `New-Object`를 사용하여 필터를 만들어야 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 참조를 참조하십시오 [DescribeAvailabilityZones](#).

## Get-EC2BundleTask

다음 코드 예시에서는 `Get-EC2BundleTask`을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 지정된 번들 작업을 설명합니다.

```
Get-EC2BundleTask -BundleId bun-12345678
```

예 2: 이 예에서는 상태가 '완료' 또는 '실패'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 Cmdlet 참조를 참조하십시오 [DescribeBundleTasks](#).AWS Tools for PowerShell

## Get-EC2CapacityReservation

다음 코드 예시에서는 `Get-EC2CapacityReservation`을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 해당 지역의 용량 예약 중 하나 이상을 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 참조를 참조하십시오 [DescribeCapacityReservations](#).

**Get-EC2ConsoleOutput**

다음 코드 예시에서는 Get-EC2ConsoleOutput을 사용하는 방법을 보여 줍니다.

**예 1: 이 예제는 지정된 Linux 인스턴스의 콘솔 출력을 가져옵니다. 콘솔 출력이 인코딩됩니다.**

예 1: 이 예제는 지정된 Linux 인스턴스의 콘솔 출력을 가져옵니다. 콘솔 출력이 인코딩됩니다.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

**출력:**

```

InstanceId      Output
-----
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

```

예 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 참조를 참조하십시오 [GetConsoleOutput](#).

## Get-EC2CustomerGateway

다음 코드 예시에서는 Get-EC2CustomerGateway을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 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
```

예 2: 이 예에서는 상태가 보류 중이거나 사용 가능한 모든 고객 게이트웨이를 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

예 3: 이 예에서는 모든 고객 게이트웨이를 설명합니다.

```
Get-EC2CustomerGateway
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeCustomerGateways](#).

## Get-EC2DhcpOption

다음 코드 예시에서는 Get-EC2DhcpOption을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 DHCP 옵션 세트를 나열합니다.

예 1: 이 예에서는 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	{}

예 2: 이 예에서는 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 참조를 참조하십시오 [DescribeDhcpOptions](#).

## Get-EC2FlowLog

다음 코드 예시에서는 Get-EC2FlowLog을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 로그 대상 유형이 's3'인 하나 이상의 흐름 로그를 설명합니다.

예 1: 이 예에서는 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               : f1-01b2e3d45f67f8901
FlowLogStatus           : ACTIVE
LogDestination           : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType      : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeFlowLogs](#).

## Get-EC2Host

다음 코드 예시에서는 Get-EC2Host을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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                : {}

```

예 2: 이 예에서는 호스트 AvailableInstanceCapacity h-01e23f4cd567899f1을 쿼리합니다.

```

Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
AvailableCapacity | Select-Object -expand AvailableInstanceCapacity

```

출력:

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11
```

- AWS Tools for PowerShell API에 [DescribeHosts](#) 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Get-EC2HostReservationOffering

다음 코드 예시에서는 Get-EC2HostReservationOffering을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 필터 '인스턴스 패밀리'에 대해 구매할 수 있는 전용 호스트 예약을 설명합니다. 여기서 PaymentOption " NoUpfront

```
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 Cmdlet 참조를 참조하십시오 [DescribeHostReservationOfferings](#).AWS Tools for PowerShell

## Get-EC2HostReservationPurchasePreview

다음 코드 예시에서는 Get-EC2HostReservationPurchasePreview을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 전용 호스트의 구성과 일치하는 구성으로 예약 구매를 미리 보여줍니다.

```
h-01e23f4cd567890f1
```

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

출력:

CurrencyCode	Purchase	TotalHourlyPrice	TotalUpfrontPrice
-----	-----	-----	-----
	{}	1.307	0.000

- API [GetHostReservationPurchasePreview](#) 세부AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Get-EC2IdFormat

다음 코드 예시에서는 Get-EC2IdFormat을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 지정된 리소스 유형의 ID 형식을 설명합니다.

```
Get-EC2IdFormat -Resource instance
```

출력:

Resource	UseLongIds
-----	-----
instance	False

예 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 참조를 참조하십시오 [DescribeIdFormat](#).

## Get-EC2IdentityIdFormat

다음 코드 예시에서는 Get-EC2IdentityIdFormat을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 주어진 역할에 대한 리소스 '이미지'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 참조를 참조하십시오 [DescribeIdentityIdFormat](#).

## Get-EC2Image

다음 코드 예시에서는 Get-EC2Image을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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
```

예 2: 이 예제는 소유하고 있는 AMI를 설명합니다.

```
Get-EC2Image -owner self
```

예 3: 이 예에서는 Microsoft Windows Server를 실행하는 퍼블릭 AMI를 설명합니다.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

예 4: 이 예에서는 'us-west-2' 지역의 모든 퍼블릭 AMI를 설명합니다.

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

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DescribeImages](#).AWS Tools for PowerShell

## Get-EC2ImageAttribute

다음 코드 예시에서는 Get-EC2ImageAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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      :
```

예 2: 이 예제는 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      :
```

예 3: 이 예시에서는 향상된 네트워킹이 활성화되었는지 여부를 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 참조를 참조하십시오 [DescribeImageAttribute](#).

## Get-EC2ImageByName

다음 코드 예시에서는 Get-EC2ImageByName을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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

```

예 2: 이 예제에서는 지정된 AMI를 설명합니다. 매월 최신 업데이트가 포함된 새 Windows AMI가 AWS 릴리스되므로 이 명령을 사용하여 AMI를 찾는 것이 유용합니다. 지정된 필터에 대해 현재 AMI를 사용하여 인스턴스를 New-EC2Instance 시작하도록 ImageId "를 지정할 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-xxxxxxxxx
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 참조를 참조하십시오 [Get-EC2ImageByName](#).

## Get-EC2ImportImageTask

다음 코드 예시에서는 Get-EC2ImportImageTask을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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  :
```

예 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 참조를 참조하십시오 [DescribeImportImageTasks](#).

## Get-EC2ImportSnapshotTask

다음 코드 예시에서는 Get-EC2ImportSnapshotTask을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 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

예 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 참조를 참조하십시오 [DescribeImportSnapshotTasks](#).

## Get-EC2Instance

다음 코드 예시에서는 Get-EC2Instance을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 지정된 인스턴스를 설명합니다.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

출력:

```
AmiLaunchIndex : 0
```

```

Architecture       : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken        : T1eEy1448154045270
EbsOptimized       : False
Hypervisor         : xen
IamInstanceProfile : Amazon.EC2.Model.IamInstanceProfile
ImageId            : ami-12345678
InstanceId         : i-12345678
InstanceLifecycle  :
InstanceType       : t2.micro
KernelId           :
KeyName            : my-key-pair
LaunchTime         : 12/4/2015 4:44:40 PM
Monitoring         : Amazon.EC2.Model.Monitoring
NetworkInterfaces  : {ip-10-0-2-172.us-west-2.compute.internal}
Placement          : Amazon.EC2.Model.Placement
Platform           : Windows
PrivateDnsName     : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress   : 10.0.2.172
ProductCodes       : {}
PublicDnsName      :
PublicIpAddress    :
RamdiskId          :
RootDeviceName     : /dev/sda1
RootDeviceType     : ebs
SecurityGroups     : {default}
SourceDestCheck    : True
SpotInstanceRequestId :
SriovNetSupport    :
State              : Amazon.EC2.Model.InstanceState
StateReason        :
StateTransitionReason :
SubnetId           : subnet-12345678
Tags               : {Name}
VirtualizationType : hvm
VpcId              : vpc-12345678

```

예 2: 이 예에서는 현재 지역의 모든 인스턴스를 예약별로 그룹화하여 설명합니다. 인스턴스 세부 정보를 보려면 각 예약 개체 내의 인스턴스 컬렉션을 확장하십시오.

```
Get-EC2Instance
```

출력:

```

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 226008221399
ReservationId   : r-c5df370c

GroupNames      : {}
Groups          : {}
Instances       : {}
OwnerId         : 123456789012
RequesterId     : 854251627541
ReservationId   : r-63e65bab
...

```

예 3: 이 예제는 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		

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DescribeInstances](#).AWS Tools for PowerShell

## Get-EC2InstanceAttribute

다음 코드 예시에서는 Get-EC2InstanceAttribute을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 인스턴스의 인스턴스 유형을 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

출력:

```
InstanceType           : t2.micro
```

예 2: 이 예제에서는 지정된 인스턴스에 향상된 네트워킹이 활성화되어 있는지 여부를 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

출력:

```
SriovNetSupport        : simple
```

예 3: 이 예에서는 지정된 인스턴스의 보안 그룹을 설명합니다.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

출력:

```
GroupId  
-----  
sg-12345678  
sg-45678901
```

예 4: 이 예제에서는 지정된 인스턴스에 대해 EBS 최적화가 활성화되었는지 여부를 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

출력:

```
EbsOptimized           : False
```

예 5: 이 예제에서는 지정된 인스턴스의 disableApiTermination " 속성을 설명합니다.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

출력:

```
DisableApiTermination          : False
```

예 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 참조를 참조하십시오 [DescribeInstanceAttribute](#).

## Get-EC2InstanceMetadata

다음 코드 예시에서는 Get-EC2InstanceMetadata을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 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
```

예 2: 인스턴스를 시작하는 데 사용된 Amazon 머신 이미지 (AMI) 의 ID를 반환합니다.

```
Get-EC2InstanceMetadata -Category AmiId
```

출력:

```
ami-b2e756ca
```

예 3: 이 예제에서는 인스턴스의 JSON 형식 자격 증명 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"
}
```

예 4: 이 예제에서는 경로 쿼리를 사용하여 인스턴스의 네트워크 인터페이스 mac을 가져옵니다.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

출력:

```
02:80:7f:ef:4c:e0/
```

예 5: 인스턴스와 연결된 IAM 역할이 있는 경우, 인스턴스 LastUpdated 날짜 InstanceProfileArn, 및 를 포함하여 인스턴스 프로필이 마지막으로 업데이트된 시간에 대한 정보를 반환합니다.

InstanceProfileId

```
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 참조를 참조하십시오 [Get-EC2InstanceMetadata](#).

## Get-EC2InstanceStatus

다음 코드 예시에서는 Get-EC2InstanceStatus을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오  
[DescribeInstanceStatus](#).

## Get-EC2InternetGateway

다음 코드 예시에서는 Get-EC2InternetGateway을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 인터넷 게이트웨이를 설명합니다.

예 1: 이 예에서는 지정된 인터넷 게이트웨이를 설명합니다.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

출력:

Attachments	InternetGatewayId	Tags
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}

예 2: 이 예에서는 모든 인터넷 게이트웨이를 설명합니다.

```
Get-EC2InternetGateway
```

출력:

Attachments	InternetGatewayId	Tags
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- API 세부 정보는 AWS Tools for PowerShell Cmdlet [DescribeInternetGateways](#) 참조의 내용을 참조하십시오.

## Get-EC2KeyPair

다음 코드 예시에서는 Get-EC2KeyPair을 사용하는 방법을 보여 줍니다.

### 예 1: 이 예제에서는 지정된 키 쌍을 설명합니다.

예 1: 이 예제에서는 지정된 키 쌍을 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
-----
```

예 2: 이 예제에서는 모든 키 페어를 설명합니다.

```
Get-EC2KeyPair
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeKeyPairs](#).

## Get-EC2NetworkAcl

다음 코드 예시에서는 Get-EC2NetworkAcl을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 지정된 네트워크 ACL의 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

```

예 3: 이 예에서는 모든 네트워크 ACL을 설명합니다.

```
Get-EC2NetworkACL
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeNetworkAcls](#).

## Get-EC2NetworkInterface

다음 코드 예시에서는 Get-EC2NetworkInterface을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 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
```

예 2: 이 예에서는 모든 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterface
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeNetworkInterfaces](#).

## Get-EC2NetworkInterfaceAttribute

다음 코드 예시에서는 Get-EC2NetworkInterfaceAttribute을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Attachment
```

출력:

```
Attachment : Amazon.EC2.Model.NetworkInterfaceAttachment
```

예 2: 이 예에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute Description
```

출력:

```
Description : My description
```

예 3: 이 예에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
GroupSet
```

출력:

```
Groups          : {my-security-group}
```

예 4: 이 예에서는 지정된 네트워크 인터페이스를 설명합니다.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
SourceDestCheck
```

출력:

```
SourceDestCheck : True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeNetworkInterfaceAttribute](#).

## Get-EC2PasswordData

다음 코드 예시에서는 Get-EC2PasswordData을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 Amazon EC2가 지정된 Windows 인스턴스의 관리자 계정에 할당한 암호를 해독합니다. pem 파일이 지정되었으므로 -Decrypt 스위치의 설정이 자동으로 가정됩니다.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

출력:

```
mYZ(PA9?C)Q
```

예 2: (PowerShell Windows만 해당) 인스턴스를 검사하여 인스턴스를 시작하는 데 사용된 키페어의 이름을 확인한 다음 Visual Studio용 AWS 툴킷의 구성 저장소에서 해당 키페어 데이터를 찾으려고 합니다. 키페어 데이터를 찾으면 암호가 해독됩니다.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

출력:

```
mYZ(PA9?C)Q
```

예 3: 인스턴스의 암호화된 암호 데이터를 반환합니다.

```
Get-EC2PasswordData -InstanceId i-12345678
```

출력:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPasswordData](#).

## Get-EC2PlacementGroup

다음 코드 예시에서는 Get-EC2PlacementGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribePlacementGroups](#).

## Get-EC2PrefixList

다음 코드 예시에서는 Get-EC2PrefixList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 해당 지역의 접두사 목록 AWS 서비스 형식으로 사용할 수 있는 항목을 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 Cmdlet 참조를 참조하십시오 [DescribePrefixLists](#). AWS Tools for PowerShell

## Get-EC2Region

다음 코드 예시에서는 Get-EC2Region을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 사용할 수 있는 지역을 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 참조를 참조하십시오 [DescribeRegions](#).

## Get-EC2RouteTable

다음 코드 예시에서는 Get-EC2RouteTable을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 :
```

예 2: 이 예제는 지정된 라우팅 테이블의 세부 정보를 반환합니다.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

예 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 참조를 참조하십시오 [DescribeRouteTables](#).

**Get-EC2ScheduledInstance**

다음 코드 예시에서는 Get-EC2ScheduledInstance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 지정된 예약 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
```

예 2: 이 예제에서는 모든 정기 인스턴스를 설명합니다.

```
Get-EC2ScheduledInstance
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet [DescribeScheduledInstances](#) 참조의 내용을 참조하십시오.

## Get-EC2ScheduledInstanceAvailability

다음 코드 예시에서는 Get-EC2ScheduledInstanceAvailability을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 매주 일요일에 지정된 날짜에 시작되는 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
...
```

예 2: 결과 범위를 좁히기 위해 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 참조를 참조하십시오  
[DescribeScheduledInstanceAvailability](#).

## Get-EC2SecurityGroup

다음 코드 예시에서는 Get-EC2SecurityGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 VPC의 지정된 보안 그룹을 설명합니다. VPC에 속한 보안 그룹을 사용할 때는 이름 (- 파라미터) 이 아닌 보안 그룹 ID (- GroupId 파라미터) 를 사용하여 그룹을 참조해야 합니다.  
GroupName

```
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
```

예 2: 이 예제에서는 EC2-Classic에 지정된 보안 그룹을 설명합니다. EC2-Classic용 보안 그룹을 사용할 때는 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             :

```

예 3: 이 예에서는 vpc-0fc1ff23456b789eb에 대한 모든 보안 그룹을 검색합니다.

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API에 대한 AWS Tools for PowerShell 세부 정보는 Cmdlet 참조를 참조하십시오.  
[DescribeSecurityGroups](#)

## Get-EC2Snapshot

다음 코드 예시에서는 Get-EC2Snapshot을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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
```

예 2: 이 예에서는 'Name' 태그가 있는 스냅샷을 설명합니다.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

예 3: 이 예에서는 'Name' 태그에 값이 'TestValue'인 스냅샷을 설명합니다.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
  $_.Tags.Value -eq "TestValue" }
```

예 4: 이 예에서는 모든 스냅샷을 설명합니다.

```
Get-EC2Snapshot -Owner self
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSnapshots](#).

## Get-EC2SnapshotAttribute

다음 코드 예시에서는 Get-EC2SnapshotAttribute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 지정된 스냅샷의 지정된 속성을 설명합니다.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

출력:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----
{}                          {}               snap-12345678
```

예 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 참조를 참조하십시오 [DescribeSnapshotAttribute](#).

## Get-EC2SpotDatafeedSubscription

다음 코드 예시에서는 Get-EC2SpotDatafeedSubscription을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 스팟 인스턴스 데이터 피드를 설명합니다.

```
Get-EC2SpotDatafeedSubscription
```

출력:

```
Bucket      : my-s3-bucket
Fault       :
OwnerId      : 123456789012
Prefix       : spotdata
State        : Active
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSpotDatafeedSubscription](#).

## Get-EC2SpotFleetInstance

다음 코드 예시에서는 Get-EC2SpotFleetInstance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeSpotFleetInstances](#).

## Get-EC2SpotFleetRequest

다음 코드 예시에서는 Get-EC2SpotFleetRequest을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 모든 스팟 플릿 요청을 설명합니다.

```
Get-EC2SpotFleetRequest
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSpotFleetRequests](#).

## Get-EC2SpotFleetRequestHistory

다음 코드 예시에서는 Get-EC2SpotFleetRequestHistory을 사용하는 방법을 보여 줍니다.

**도구: PowerShell**

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeSpotFleetRequestHistory](#).

**Get-EC2SpotInstanceRequest**

다음 코드 예시에서는 Get-EC2SpotInstanceRequest을 사용하는 방법을 보여 줍니다.

**도구: PowerShell**

예 1: 이 예에서는 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

```

예 2: 이 예에서는 모든 스팟 인스턴스 요청을 설명합니다.

```
Get-EC2SpotInstanceRequest
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSpotInstanceRequests](#).

**Get-EC2SpotPriceHistory**

다음 코드 예시에서는 Get-EC2SpotPriceHistory를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 인스턴스 유형 및 가용 영역에 대한 스팟 가격 기록의 최근 10개 항목을 가져옵니다. 참고로 - AvailabilityZone 매개 변수에 지정된 값은 cmdlet의 -Region 매개 변수 (예에는 표시되지 않음)에 제공된 지역 값에 유효하거나 셸에서 기본값으로 설정되어 있어야 합니다. 이 예제 명령은 환경에 기본 지역인 'us-west-2'가 설정되어 있다고 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
...

```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DescribeSpotPriceHistory](#).AWS Tools for PowerShell

## Get-EC2Subnet

다음 코드 예시에서는 Get-EC2Subnet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 모든 서브넷을 설명합니다.

```
Get-EC2Subnet
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSubnets](#).

## Get-EC2Tag

다음 코드 예시에서는 Get-EC2Tag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 리소스 유형 '이미지'의 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

예 2: 이 예제는 모든 리소스의 모든 태그를 가져와서 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}

```

예 3: 이 예에서는 지정된 지역에 대해 'auto delete' 태그가 있고 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 예제는 값이 'no' 인 'auto-delete'라는 태그가 있는 모든 리소스를 가져오고 다음 파이프에서 필터를 추가하여 '인스턴스' 리소스 유형만 파싱하고 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 예제는 모든 인스턴스 리소스의 태그와 'Name' 키를 가져와서 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 참조를 참조하십시오 [DescribeTags](#).

## Get-EC2Volume

다음 코드 예시에서는 Get-EC2Volume을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 상태가 '사용 가능'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
...
```

예 3: 이 예에서는 모든 EBS 볼륨을 설명합니다.

```
Get-EC2Volume
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeVolumes](#).

## Get-EC2VolumeAttribute

다음 코드 예시에서는 Get-EC2VolumeAttribute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeVolumeAttribute](#).

## Get-EC2VolumeStatus

다음 코드 예시에서는 Get-EC2VolumeStatus을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeVolumeStatus](#).

## Get-EC2Vpc

다음 코드 예시에서는 Get-EC2Vpc을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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
```

예 2: 이 예제에서는 기본 VPC를 설명합니다 (지역당 하나만 있을 수 있음). 계정이 이 지역에서 EC2-Classic을 지원하는 경우 기본 VPC는 없습니다.

```
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
```

예 3: 이 예에서는 지정된 필터와 일치하는 VPC (즉, '10.0.0.0/16' 값과 일치하고 '사용 가능' 상태인 CIDR이 있는) 를 설명합니다.

```
Get-EC2Vpc -Filter @{Name="cidr";
  Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

예 4: 이 예제에서는 모든 VPC에 대해 설명합니다.

```
Get-EC2Vpc
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeVpcs](#).

## Get-EC2VpcAttribute

다음 코드 예시에서는 Get-EC2VpcAttribute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 'enableDnsSupport' 속성에 대해 설명합니다.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

출력:

```
EnableDnsSupport
-----
True
```

예 2: 이 예제에서는 'enableDnsHostnames' 속성에 대해 설명합니다.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

출력:

```
EnableDnsHostnames
-----
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeVpcAttribute](#).

## Get-EC2VpcClassicLink

다음 코드 예시에서는 Get-EC2VpcClassicLink을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 위 예시는 해당 지역의 ClassicLinkEnabled 상태와 함께 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 참조를 참조하십시오 [DescribeVpcClassicLink](#).

## Get-EC2VpcClassicLinkDnsSupport

다음 코드 예시에서는 Get-EC2VpcClassicLinkDnsSupport을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 eu-west-1 지역에 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 Cmdlet 참조를 참조하십시오. [DescribeVpcClassicLinkDnsSupport](#) AWS Tools for PowerShell

## Get-EC2VpcEndpoint

다음 코드 예시에서는 Get-EC2VpcEndpoint을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 eu-west-1 지역의 VPC 엔드포인트 중 하나 이상을 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 지역의 모든 vpc 엔드포인트를 설명하고 VpcEndpointId, VpcId 속성을 선택하여 표 형식으로 표시합니다. ServiceName PrivateDnsEnabled

```
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 예에서는 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 [DescribeVpcEndpoints](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Get-EC2VpcEndpointService

다음 코드 예시에서는 Get-EC2VpcEndpointService을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 필터 (이 경우에는 com.amazonaws.eu-west-1.ecs) 를 사용하는 EC2 VPC 엔드포인트 서비스를 설명합니다. 또한 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
```

예 2: 이 예제는 모든 EC2 VPC 엔드포인트 서비스를 검색하고 일치하는 “ssm”을 반환합니다.  
ServiceNames

```
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 Cmdlet 참조를 참조하십시오 [DescribeVpcEndpointServices](#). AWS Tools for PowerShell

## Get-EC2VpnConnection

다음 코드 예시에서는 Get-EC2VpnConnection을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 상태가 보류 중이거나 사용 가능한 모든 VPN 연결을 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )
```

```
Get-EC2VpnConnection -Filter $filter
```

예 3: 이 예에서는 모든 VPN 연결을 설명합니다.

```
Get-EC2VpnConnection
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeVpnConnections](#).

## Get-EC2VpnGateway

다음 코드 예시에서는 Get-EC2VpnGateway을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 가상 프라이빗 게이트웨이를 설명합니다.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

출력:

```
AvailabilityZone :
State            : available
Tags            : {}
Type            : ipsec.1
VpcAttachments  : {vpc-12345678}
VpnGatewayId    : vgw-1a2b3c4d
```

예 2: 이 예에서는 상태가 보류 중이거나 사용 가능한 모든 가상 프라이빗 게이트웨이를 설명합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

예 3: 이 예에서는 모든 가상 사설 게이트웨이를 설명합니다.

```
Get-EC2VpnGateway
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeVpnGateways](#).

## Grant-EC2SecurityGroupEgress

다음 코드 예시에서는 Grant-EC2SecurityGroupEgress을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 EC2-VPC 보안 그룹에 대한 송신 규칙을 정의합니다. 이 규칙은 TCP 포트 80의 지정된 IP 주소 범위에 대한 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
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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
```

예 3: 이 예제는 TCP 포트 80의 지정된 소스 보안 그룹에 대한 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 참조를 참조하십시오 [AuthorizeSecurityGroupEgress](#).

## Grant-EC2SecurityGroupIngress

다음 코드 예시에서는 Grant-EC2SecurityGroupIngress을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 EC2-VPC 보안 그룹의 인그레스 규칙을 정의합니다. 이 규칙은 SSH (포트 22) 및 RDC (포트 3389)의 특정 IP 주소에 대한 액세스 권한을 부여합니다. 보안 그룹 이름이 아닌 보안 그룹 ID를 사용하여 EC2-VPC 보안 그룹을 식별해야 한다는 점에 유의하십시오. 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 )
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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 )
```

예 3: 이 예에서는 EC2-Classic의 보안 그룹에 대한 인그레스 규칙을 정의합니다. 이 규칙은 SSH (포트 22) 및 RDC (포트 3389)의 특정 IP 주소에 대한 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 )
```

예 4: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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 )
```

예 5: 이 예에서는 지정된 소스 보안 그룹 (sg-1a2b3c4d) 에서 지정된 보안 그룹 (sg-12345678) 에 대한 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 } )
```

예 6: 이 예에서는 TCP 포트 22 트래픽에 대한 CIDR 5.5.5.5/32를 TCP 포트 22 트래픽에 대한 보안 그룹 sg-1234abcd의 인그레스 규칙에 설명과 함께 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 [AuthorizeSecurityGroupIngress](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## Import-EC2Image

다음 코드 예시에서는 Import-EC2Image을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 지정된 Amazon S3 버킷에서 동일 토큰을 사용하여 Amazon EC2로 단일 디스크 가상 머신 이미지를 가져옵니다. 이 예제에서는 VM Import 사전 요구 사항 항목에 설명된 대로, 지정된 버킷에 대한 Amazon EC2 액세스를 허용하는 정책과 함께 기본 이름이 'vmimport'인 VM 가져오기 서비스 역할이 있어야 합니다. 사용자 지정 역할을 사용하려면 파라미터를 사용하여 역할 이름을 지정하십시오. **-RoleName**

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "myVirtualMachineImages"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @params
```

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 참조를 참조하십시오 [ImportImage](#).

## Import-EC2KeyPair

다음 코드 예시에서는 Import-EC2KeyPair를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 퍼블릭 키를 EC2로 가져옵니다. 첫 번째 줄은 퍼블릭 키 파일 (\*.pub)의 콘텐츠를 변수에 저장합니다. **\$publickey** 다음으로, 예제에서는 공개 키 파일의 UTF8 형식을 Base64로 인코딩된 문자열로 변환하고 변환된 문자열을 변수에 저장합니다. **\$pkbase64** 마지막 줄에서는 변환된 공개 키를 EC2로 가져옵니다. cmdlet은 키 핑거프린트와 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 참조를 참조하십시오 [ImportKeyPair](#).

## Import-EC2Snapshot

다음 코드 예시에서는 Import-EC2Snapshot을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 'VMDK' 형식의 VM 디스크 이미지를 Amazon EBS 스냅샷으로 가져옵니다. 이 예제에는 AWSEC <http://docs.aws.amazon.com/WindowsGuide/ImportPrerequisites/2/latest/VM.html>의 항목에 설명된 대로 Amazon EC2가 지정된 버킷에 액세스할 수 있도록 허용하는 정

책과 함께 기본 이름이 'vmimport'인 **VM Import Prerequisites** VM 가져오기 서비스 역할이 필요합니다. 사용자 지정 역할을 사용하려면 파라미터를 사용하여 역할 이름을 지정하십시오. -

### RoleName

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "myVirtualMachineImages"
    "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 참조를 참조하십시오 [ImportSnapshot](#).

## Move-EC2AddressToVpc

다음 코드 예시에서는 Move-EC2AddressToVpc을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 퍼블릭 IP 주소가 12.345.67.89인 EC2 인스턴스를 미국 동부 (버지니아 북부) 지역의 EC2-VPC 플랫폼으로 이동합니다.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

예 2: 이 예제는 명령 결과를 cmdlet으로 파이프합니다. Get-EC2Instance Move-EC2AddressToVpc 이 Get-EC2Instance 명령은 인스턴스 ID로 지정된 인스턴스를 가져온 다음 인스턴스의 퍼블릭 IP 주소 속성을 반환합니다.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [MoveAddressToVpc](#).

## New-EC2Address

다음 코드 예시에서는 New-EC2Address를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 VPC의 인스턴스에 사용할 엘라스틱 IP 주소를 할당합니다.

```
New-EC2Address -Domain Vpc
```

출력:

AllocationId	Domain	PublicIp
-----	-----	-----
eipalloc-12345678	vpc	198.51.100.2

예 2: 이 예시에서는 EC2-Classic의 인스턴스에 사용할 엘라스틱 IP 주소를 할당합니다.

```
New-EC2Address
```

출력:

AllocationId	Domain	PublicIp
-----	-----	-----
	standard	203.0.113.17

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AllocateAddress](#).AWS Tools for PowerShell

## New-EC2CustomerGateway

다음 코드 예시에서는 New-EC2CustomerGateway를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [CreateCustomerGateway](#).

## New-EC2DhcpOption

다음 코드 예시에서는 New-EC2DhcpOption을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 지정된 DHCP 옵션 세트를 만듭니다. 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	{}

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 각 DHCP 옵션을 생성해야 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 Cmdlet 참조를 참조하십시오 [CreateDhcpOptions](#). AWS Tools for PowerShell

## New-EC2FlowLog

다음 코드 예시에서는 New-EC2FlowLog을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 '관리자' 역할의 권한을 사용하여 모든 'REJECT' 트래픽에 대해 서브넷 서브넷-1d234567에 대한 EC2 플로우 로그를 cloud-watch-log 지정된 'subnet1-log'에 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//VUK15EU6cF7gQL0xcqNefvjeTGw= {f1-012fc34eed5678c9d} {}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오. [CreateFlowLogs](#)

## New-EC2Host

다음 코드 예시에서는 New-EC2Host을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [AllocateHosts](#).

## New-EC2HostReservation

다음 코드 예시에서는 New-EC2HostReservation을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 전용 호스트 h-01e23f4cd567890f1의 구성과 일치하는 구성을 갖춘 예약 오퍼링 hro-0c1f23456789d0ab를 구매합니다.

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
h-01e23f4cd567890f1
```

출력:

```
ClientToken      :
CurrencyCode     :
Purchase         : {hr-0123f4b5d67bedc89}
TotalHourlyPrice : 1.307
TotalUpfrontPrice : 0.000
```

- [PurchaseHostReservation AWS Tools for PowerShell](#) API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

## New-EC2Image

다음 코드 예시에서는 New-EC2Image을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 지정된 인스턴스에서 지정된 이름과 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"
```

예 2: 이 예제는 지정된 인스턴스에서 지정된 이름과 설명으로 AMI를 생성합니다. Amazon EC2는 인스턴스를 종료하고 다시 시작하지 않고 이미지를 생성하므로 생성된 이미지의 파일 시스템 무결성을 보장할 수 없습니다.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web server AMI" -NoReboot $true
```

예 3: 이 예제는 세 개의 볼륨으로 AMI를 생성합니다. 첫 번째 볼륨은 Amazon EBS 스냅샷을 기반으로 합니다. 두 번째 볼륨은 비어 있는 100GiB 아마존 EBS 볼륨입니다. 세 번째 볼륨은 인스턴스 스토어 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 참조를 참조하십시오 [CreateImage](#).

## New-EC2Instance

다음 코드 예시에서는 New-EC2Instance을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 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
```

예 2: 이 예제는 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
```

예 3: EBS 볼륨 또는 인스턴스 스토어 볼륨을 추가하려면 블록 디바이스 매핑을 정의하고 명령에 추가합니다. 이 예제에서는 인스턴스 스토어 볼륨을 추가합니다.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

예 4: 현재 Windows AMI 중 하나를 지정하려면 `Get-EC2ImageByName`을 사용하여 AMI ID를 가져오십시오. 이 예제는 윈도우 서버 2016의 현재 기본 AMI에서 인스턴스를 시작합니다.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

예 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
```

예 6: 이 요청은 두 개의 인스턴스를 시작하고 웹 서버 키와 프로덕션 값이 포함된 태그를 인스턴스에 적용합니다. 또한 요청은 `cost-center` 키와 `cc123` 값을 가진 태그를 생성된 볼륨 (이 경우 각 인스턴스의 루트 볼륨)에 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 Cmdlet 참조를 참조하십시오 [RunInstances](#).AWS Tools for PowerShell

## New-EC2InstanceExportTask

다음 코드 예시에서는 New-EC2InstanceExportTask을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 중지된 인스턴스를 가상 하드 디스크 (VHD) 로 S3 **testbucket-export-instances-2019** 버킷으로 내보냅니다. **i-0800b00a00EXAMPLE** 대상 환경은 이며 **Microsoft**, 사용자의 기본 Region은 **us-east-1**이 아닌 반면 인스턴스는 지역에 있기 때문에 AWS 지역 매개변수가 추가됩니다. **us-east-1** 내보내기 작업의 상태를 확인하려면 이 명령의 결과에서 **ExportTaskId** 값을 복사한 다음 실행하십시오. **Get-EC2ExportTask -ExportTaskId export\_task\_ID\_from\_results.**

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "testbucket-export-
instances-2019" -TargetEnvironment Microsoft -Region us-east-1
```

출력:

```
Description          :
ExportTaskId         : export-i-077c73108aEXAMPLE
ExportToS3Task       : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                : active
StatusMessage        :
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateInstanceExportTask](#).

## New-EC2InternetGateway

다음 코드 예시에서는 New-EC2InternetGateway을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 인터넷 게이트웨이를 생성합니다.

```
New-EC2InternetGateway
```

출력:

Attachments	InternetGatewayId	Tags
-----	-----	----
{}	igw-1a2b3c4d	{}

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateInternetGateway](#).

## New-EC2KeyPair

다음 코드 예시에서는 New-EC2KeyPair을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 이 예제에서는 키 페어를 생성하고 지정된 이름의 파일에 PEM으로 인코딩된 RSA 개인 키를 캡처합니다. 를 사용하는 경우 유효한 키를 PowerShell 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 참조하십시오.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -FilePath C:\path\my-key-pair.pem
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateKeyPair](#).

## New-EC2NetworkACL

다음 코드 예시에서는 New-EC2NetworkACL을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 이 예제는 지정된 VPC에 대한 네트워크 ACL을 생성합니다.

```
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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateNetworkAcl](#).

**New-EC2NetworkAclEntry**

다음 코드 예시에서는 New-EC2NetworkAclEntry를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 네트워크 ACL에 대한 항목을 생성합니다. 이 규칙은 UDP 포트 53 (DNS)의 모든 위치 (0.0.0.0/0) 에서 연결된 서브넷으로 들어오는 인바운드 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [CreateNetworkAclEntry](#) AWS Tools for PowerShell

**New-EC2NetworkInterface**

다음 코드 예시에서는 New-EC2NetworkInterface를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [CreateNetworkInterface](#).

**New-EC2PlacementGroup**

다음 코드 예시에서는 New-EC2PlacementGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예시에서는 지정된 이름의 배치 그룹을 생성합니다.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreatePlacementGroup](#).

**New-EC2Route**

다음 코드 예시에서는 New-EC2Route을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 지정된 라우팅 테이블에 대해 지정된 경로를 생성합니다. 경로는 모든 트래픽을 매칭하여 지정된 인터넷 게이트웨이로 전송합니다.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId igw-1a2b3c4d
```

출력:

```
True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateRoute](#).

## New-EC2RouteTable

다음 코드 예시에서는 New-EC2RouteTable을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [CreateRouteTable](#).

## New-EC2ScheduledInstance

다음 코드 예시에서는 New-EC2ScheduledInstance을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 지정된 정기 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 참조를 참조하십시오 [RunScheduledInstances](#).

## New-EC2ScheduledInstancePurchase

다음 코드 예시에서는 New-EC2ScheduledInstancePurchase을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 정기 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 참조를 참조하십시오 [PurchaseScheduledInstances](#).

## New-EC2SecurityGroup

다음 코드 예시에서는 New-EC2SecurityGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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
```

예 2: 이 예제에서는 EC2-Classic용 보안 그룹을 생성합니다.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

출력:

```
sg-45678901
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateSecurityGroup](#).

## New-EC2Snapshot

다음 코드 예시에서는 New-EC2Snapshot을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [CreateSnapshot](#).

**New-EC2SpotDatafeedSubscription**

다음 코드 예시에서는 New-EC2SpotDatafeedSubscription을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 스팟 인스턴스 데이터 피드를 생성합니다.

```
New-EC2SpotDatafeedSubscription -Bucket my-s3-bucket -Prefix spotdata
```

**출력:**

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateSpotDatafeedSubscription](#).

## New-EC2Subnet

다음 코드 예시에서는 New-EC2Subnet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 CIDR을 사용하여 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateSubnet](#).

## New-EC2Tag

다음 코드 예시에서는 New-EC2Tag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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" }
```

예 2: 이 예제는 지정된 리소스를 업데이트하거나 지정된 태그를 추가합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
@{ Key="test"; Value="anotherTagValue" } )
```

예 3: PowerShell 버전 2에서는 `New-Object`를 사용하여 Tag 매개 변수에 대한 태그를 만들어야 합니다.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateTags](#).

## New-EC2Volume

다음 코드 예시에서는 `New-EC2Volume`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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
```

예 2: 이 예제 요청은 볼륨을 생성하고 스택 키와 프로덕션 값이 포함된 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 참조를 참조하십시오 [CreateVolume](#).

## New-EC2Vpc

다음 코드 예시에서는 New-EC2Vpc을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 지정된 CIDR을 사용하여 VPC를 생성합니다. 또한 Amazon VPC는 VPC를 위해 기본 DHCP 옵션 세트, 기본 라우팅 테이블, 기본 네트워크 ACL 등을 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
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오 [CreateVpc](#). AWS Tools for PowerShell

## New-EC2VpcEndpoint

다음 코드 예시에서는 New-EC2VpcEndpoint을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예시에서는 vpc-0fc1ff23f45b678eb에 com.amazonaws.eu-west-1.s3 서비스를 위한 새 VPC 엔드포인트를 생성합니다.

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

출력:

```
ClientToken VpcEndpoint
-----
                Amazon.EC2.Model.VpcEndpoint
```

- API [CreateVpcEndpoint](#) 세부 AWS Tools for PowerShell 정보는 Cmdlet 참조를 참조하십시오.

## New-EC2VpnConnection

다음 코드 예시에서는 New-EC2VpnConnection을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 가상 프라이빗 게이트웨이와 지정된 고객 게이트웨이 간에 VPN 연결을 생성합니다. 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
```

예 2: 이 예제는 VPN 연결을 만들고 지정된 이름의 파일에 구성을 캡처합니다.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

예 3: 이 예에서는 지정된 가상 프라이빗 게이트웨이와 지정된 고객 게이트웨이 간에 정적 라우팅을 사용하여 VPN 연결을 생성합니다.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateVpnConnection](#).

## New-EC2VpnConnectionRoute

다음 코드 예시에서는 New-EC2VpnConnectionRoute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 VPN 연결에 대해 지정된 고정 경로를 만듭니다.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock 11.12.0.0/16
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateVpnConnectionRoute](#).

## New-EC2VpnGateway

다음 코드 예시에서는 New-EC2VpnGateway을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [CreateVpnGateway](#).

## Register-EC2Address

다음 코드 예시에서는 Register-EC2Address을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 엘라스틱 IP 주소를 VPC의 지정된 인스턴스와 연결합니다.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

출력:

```
eipassoc-12345678
```

예 2: 이 예제는 지정된 엘라스틱 IP 주소를 EC2-Classic의 지정된 인스턴스와 연결합니다.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AssociateAddress](#).AWS Tools for PowerShell

## Register-EC2DhcpOption

다음 코드 예시에서는 Register-EC2DhcpOption을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 DHCP 옵션 세트를 지정된 VPC와 연결합니다.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

예 2: 이 예제는 기본 DHCP 옵션 세트를 지정된 VPC와 연결합니다.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [AssociateDhcpOptions](#).AWS Tools for PowerShell

## Register-EC2Image

다음 코드 예시에서는 Register-EC2Image을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 Amazon S3의 지정된 매니페스트 파일을 사용하여 AMI를 등록합니다.

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/image.manifest.xml -
Name my-web-server-ami
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterImage](#).

## Register-EC2PrivateIpAddress

다음 코드 예시에서는 Register-EC2PrivateIpAddress을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 이 예에서는 지정된 보조 사설 IP 주소를 지정된 네트워크 인터페이스에 할당합니다.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
10.0.0.82
```

예 2: 이 예에서는 두 개의 보조 사설 IP 주소를 만들어 지정된 네트워크 인터페이스에 할당합니다.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AssignPrivateIpAddresses](#).

## Register-EC2RouteTable

다음 코드 예시에서는 Register-EC2RouteTable을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [AssociateRouteTable](#).

## Remove-EC2Address

다음 코드 예시에서는 Remove-EC2Address를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 VPC의 인스턴스에 대해 지정된 엘라스틱 IP 주소를 해제합니다.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

예 2: 이 예제는 EC2-Classic의 인스턴스에 대해 지정된 엘라스틱 IP 주소를 릴리스합니다.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ReleaseAddress](#).

## Remove-EC2CapacityReservation

다음 코드 예시에서는 Remove-EC2CapacityReservation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 용량 예약을 취소합니다. cr-0c1f2345db6f7cdba

```
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
```

- AWS Tools for PowerShell API에 대한 [CancelCapacityReservation](#) 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Remove-EC2CustomerGateway

다음 코드 예시에서는 Remove-EC2CustomerGateway을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 고객 게이트웨이를 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteCustomerGateway](#).

## Remove-EC2DhcpOption

다음 코드 예시에서는 Remove-EC2DhcpOption을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 DHCP 옵션 세트를 삭제합니다. Force 매개 변수도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteDhcpOptions](#).

## Remove-EC2FlowLog

다음 코드 예시에서는 Remove-EC2FlowLog을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 주어진 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
"f1-01a2b3456a789c01".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- AWS Tools for PowerShell API에 대한 [DeleteFlowLogs](#) 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Remove-EC2Host

다음 코드 예시에서는 Remove-EC2Host을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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} {}

```

- AWS Tools for PowerShell API에 [ReleaseHosts](#)에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

**Remove-EC2Instance**

다음 코드 예시에서는 Remove-EC2Instance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 인스턴스를 종료합니다 (인스턴스가 실행 중이거나 '중지' 상태일 수 있음). 계속하기 전에 cmdlet에서 확인 메시지를 표시합니다. 프롬프트를 표시하지 않으려면 -Force 스위치를 사용하십시오.

```
Remove-EC2Instance -InstanceId i-12345678
```

**출력:**

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [TerminateInstances](#) AWS Tools for PowerShell

**Remove-EC2InternetGateway**

다음 코드 예시에서는 Remove-EC2InternetGateway을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 지정된 인터넷 게이트웨이를 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteInternetGateway](#).

## Remove-EC2KeyPair

다음 코드 예시에서는 Remove-EC2KeyPair를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 지정된 키 쌍을 삭제합니다. Force 매개 변수도 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteKeyPair](#).

## Remove-EC2NetworkAcl

다음 코드 예시에서는 Remove-EC2NetworkAcl을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 네트워크 ACL을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-EC2NetworkAcl -NetworkAclId ac1-12345678
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
"ac1-12345678".
[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 참조를 참조하십시오 [DeleteNetworkAcl](#).

## Remove-EC2NetworkAclEntry

다음 코드 예시에서는 Remove-EC2NetworkAclEntry을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 네트워크 ACL에서 지정된 규칙을 제거합니다. Force 매개 변수도 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-EC2NetworkAclEntry -NetworkAclId ac1-12345678 -Egress $false -RuleNumber 100
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on Target
"ac1-12345678".
```

```
[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 참조를 참조하십시오 [DeleteNetworkAclEntry](#).

## Remove-EC2NetworkInterface

다음 코드 예시에서는 Remove-EC2NetworkInterface을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 네트워크 인터페이스를 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteNetworkInterface](#).

## Remove-EC2PlacementGroup

다음 코드 예시에서는 Remove-EC2PlacementGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 배치 그룹을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeletePlacementGroup](#).

**Remove-EC2Route**

다음 코드 예시에서는 Remove-EC2Route을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 라우팅 테이블에서 지정된 경로를 삭제합니다. Force 파라미터도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteRoute](#).

**Remove-EC2RouteTable**

다음 코드 예시에서는 Remove-EC2RouteTable을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 라우팅 테이블을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteRouteTable](#).

## Remove-EC2SecurityGroup

다음 코드 예시에서는 Remove-EC2SecurityGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 EC2-VPC 보안 그룹을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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"):
```

예 2: 이 예에서는 EC2-Classic에 대해 지정된 보안 그룹을 삭제합니다.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API에 대한 세부 정보는 Cmdlet [DeleteSecurityGroup](#) 참조의 내용을 참조하십시오. AWS Tools for PowerShell

## Remove-EC2Snapshot

다음 코드 예시에서는 Remove-EC2Snapshot을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 스냅샷을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteSnapshot](#).

## Remove-EC2SpotDatafeedSubscription

다음 코드 예시에서는 Remove-EC2SpotDatafeedSubscription을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 스팟 인스턴스 데이터 피드를 삭제합니다. Force 파라미터를 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteSpotDatafeedSubscription](#).

## Remove-EC2Subnet

다음 코드 예시에서는 Remove-EC2Subnet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 서브넷을 삭제합니다. Force 매개 변수도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteSubnet](#).

## Remove-EC2Tag

다음 코드 예시에서는 Remove-EC2Tag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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
```

예 2: 이 예제는 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
```

예 3: 이 예제는 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
```

예 4: 이 예제는 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteTags](#).

## Remove-EC2Volume

다음 코드 예시에서는 Remove-EC2Volume을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 볼륨을 분리합니다. Force 매개 변수도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteVolume](#).

## Remove-EC2Vpc

다음 코드 예시에서는 Remove-EC2Vpc을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 VPC를 삭제합니다. Force 파라미터도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteVpc](#).

## Remove-EC2VpnConnection

다음 코드 예시에서는 Remove-EC2VpnConnection을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 VPN 연결을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteVpnConnection](#).

## Remove-EC2VpnConnectionRoute

다음 코드 예시에서는 Remove-EC2VpnConnectionRoute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 VPN 연결에서 지정된 고정 경로를 제거합니다. Force 매개 변수도 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteVpnConnectionRoute](#).

## Remove-EC2VpnGateway

다음 코드 예시에서는 Remove-EC2VpnGateway을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 가상 프라이빗 게이트웨이를 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteVpnGateway](#).

## Request-EC2SpotFleet

다음 코드 예시에서는 Request-EC2SpotFleet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 인스턴스 유형에 대해 최저 가격으로 가용 영역에 스팟 플릿 요청을 생성합니다. 계정이 EC2-VPC 전용을 지원하는 경우 스팟 플릿은 기본 서브넷이 있는 최저 가격의 가용 영역에서 인스턴스를 시작합니다. 계정이 EC2-Classic을 지원하는 경우 스팟 플릿은 최저 가격의 가용 영역에서 EC2-Classic의 인스턴스를 시작합니다. 지불하는 가격은 요청에 지정된 스팟 가격을 초과하지 않는다는 점에 유의하십시오.

```
$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 참조를 참조하십시오 [RequestSpotFleet](#).

## Request-EC2SpotInstance

다음 코드 예시에서는 Request-EC2SpotInstance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 서브넷에서 일회성 스팟 인스턴스를 요청합니다. 보안 그룹은 지정된 서브넷이 포함된 VPC에 대해 생성되어야 하며, 네트워크 인터페이스를 사용하여 ID로 지정해야 합니다.

다. 네트워크 인터페이스를 지정할 때는 네트워크 인터페이스를 사용하여 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 참조를 참조하십시오 [RequestSpotInstances](#).

## Reset-EC2ImageAttribute

다음 코드 예시에서는 Reset-EC2ImageAttribute을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제에서는 'LaunchPermission' 속성을 기본값으로 재설정합니다. 기본적으로 AMI는 비공개입니다.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ResetImageAttribute](#).

## Reset-EC2InstanceAttribute

다음 코드 예시에서는 Reset-EC2InstanceAttribute을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 이 예제는 지정된 인스턴스의 sriovNetSupport " 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

예 2: 이 예제에서는 지정된 인스턴스의 'EBSOptimized' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

예 3: 이 예제에서는 지정된 인스턴스의 'sourceDestCheck' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

예 4: 이 예제에서는 지정된 인스턴스의 disableApiTermination " 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

예 5: 이 예제에서는 지정된 인스턴스의 'instanceInitiatedShutdownBehavior' 속성을 재설정합니다.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ResetInstanceAttribute](#). AWS Tools for PowerShell

## Reset-EC2NetworkInterfaceAttribute

다음 코드 예시에서는 `Reset-EC2NetworkInterfaceAttribute`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 네트워크 인터페이스의 원본/대상 검사를 재설정합니다.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ResetNetworkInterfaceAttribute](#). AWS Tools for PowerShell

## Reset-EC2SnapshotAttribute

다음 코드 예시에서는 `Reset-EC2SnapshotAttribute`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 스냅샷의 지정된 속성을 재설정합니다.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ResetSnapshotAttribute](#).

## Restart-EC2Instance

다음 코드 예시에서는 `Restart-EC2Instance`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 인스턴스를 재부팅합니다.

```
Restart-EC2Instance -InstanceId i-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RebootInstances](#).

## Revoke-EC2SecurityGroupEgress

다음 코드 예시에서는 Revoke-EC2SecurityGroupEgress을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 EC2-VPC 보안 그룹에 대한 규칙을 제거합니다. 이렇게 하면 TCP 포트 80에서 지정된 IP 주소 범위에 대한 액세스가 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
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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
```

예 3: 이 예에서는 TCP 포트 80에서 지정된 소스 보안 그룹에 대한 액세스를 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 Cmdlet 참조를 참조하십시오 [RevokeSecurityGroupEgress](#).AWS Tools for PowerShell

## Revoke-EC2SecurityGroupIngress

다음 코드 예시에서는 Revoke-EC2SecurityGroupIngress을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예에서는 EC2-VPC 보안 그룹의 지정된 주소 범위에서 TCP 포트 22에 대한 액세스를 취소합니다. 보안 그룹 이름이 아닌 보안 그룹 ID를 사용하여 EC2-VPC 보안 그룹을 식별해야 한다는 점에 유의하십시오. 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
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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
```

예 3: 이 예에서는 EC2-Classic의 지정된 보안 그룹에 대해 지정된 주소 범위에서 TCP 포트 22에 대한 액세스를 취소합니다. 이 예제에서 사용하는 구문에는 버전 3 이상이 필요합니다 PowerShell .

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

예 4: PowerShell 버전 2에서는 New-Object를 사용하여 객체를 생성해야 합니다. IpPermission

```
$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 참조를 참조하십시오 [RevokeSecurityGroupIngress](#).

## Send-EC2InstanceStatus

다음 코드 예시에서는 Send-EC2InstanceStatus를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 인스턴스에 대한 상태 피드백을 보고합니다.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode unresponsive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ReportInstanceStatus](#).

## Set-EC2NetworkAclAssociation

다음 코드 예시에서는 Set-EC2NetworkAclAssociation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 네트워크 ACL을 지정된 네트워크 ACL 연결의 서브넷과 연결합니다.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId aclassoc-1a2b3c4d
```

출력:

```
aclassoc-87654321
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [ReplaceNetworkAclAssociation](#).AWS Tools for PowerShell

## Set-EC2NetworkAclEntry

다음 코드 예시에서는 Set-EC2NetworkAclEntry를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 네트워크 ACL의 지정된 항목을 대체합니다. 새 규칙은 지정된 주소에서 연결된 모든 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 참조를 참조하십시오 [ReplaceNetworkAclEntry](#).

## Set-EC2Route

다음 코드 예시에서는 Set-EC2Route을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [ReplaceRoute](#).

## Set-EC2RouteTableAssociation

다음 코드 예시에서는 Set-EC2RouteTableAssociation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 라우팅 테이블을 지정된 라우팅 테이블 연결의 서브넷과 연결합니다.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId  
rtbassoc-12345678
```

출력:

```
rtbassoc-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ReplaceRouteTableAssociation](#).

## Start-EC2Instance

다음 코드 예시에서는 Start-EC2Instance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 인스턴스를 시작합니다.

```
Start-EC2Instance -InstanceId i-12345678
```

출력:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

예 2: 이 예제는 지정된 인스턴스를 시작합니다.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

예 3: 이 예제는 현재 중지된 인스턴스 세트를 시작합니다. 에서 반환된 인스턴스 Get-EC2Instance 객체는 파이프로 연결됩니다. Start-EC2Instance 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances  
| Start-EC2Instance
```

예 4: PowerShell 버전 2에서는 New-Object를 사용하여 Filter 매개 변수에 대한 필터를 만들어야 합니다.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "instance-state-name"  
$filter.Values = "stopped"  
  
(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StartInstances](#).

## Start-EC2InstanceMonitoring

다음 코드 예시에서는 Start-EC2InstanceMonitoring을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [MonitorInstances](#).

## Stop-EC2ImportTask

다음 코드 예시에서는 Stop-EC2ImportTask을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 가져오기 작업 (스냅샷 또는 이미지 가져오기) 을 취소합니다. 필요한 경우 **-CancelReason** 매개변수를 사용하여 이유를 제공할 수 있습니다.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CancelImportTask](#).

## Stop-EC2Instance

다음 코드 예시에서는 Stop-EC2Instance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [StopInstances](#).

## Stop-EC2InstanceMonitoring

다음 코드 예시에서는 Stop-EC2InstanceMonitoring을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [UnmonitorInstances](#).

## Stop-EC2SpotFleetRequest

다음 코드 예시에서는 Stop-EC2SpotFleetRequest을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 스팟 플릿 요청을 취소하고 연결된 스팟 인스턴스를 종료합니다.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

예 2: 이 예제는 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 참조를 참조하십시오 [CancelSpotFleetRequests](#).

## Stop-EC2SpotInstanceRequest

다음 코드 예시에서는 Stop-EC2SpotInstanceRequest을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 스팟 인스턴스 요청을 취소합니다.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

출력:

SpotInstanceRequestId	State
-----	-----
sir-12345678	cancelled

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CancelSpotInstanceRequests](#).

## Unregister-EC2Address

다음 코드 예시에서는 Unregister-EC2Address을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 VPC의 지정된 인스턴스에서 지정된 엘라스틱 IP 주소를 분리합니다.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

예 2: 이 예제는 EC2-Classic의 지정된 인스턴스에서 지정된 엘라스틱 IP 주소를 분리합니다.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DisassociateAddress](#). AWS Tools for PowerShell

## Unregister-EC2Image

다음 코드 예시에서는 Unregister-EC2Image을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 AMI를 등록 취소합니다.

```
Unregister-EC2Image -ImageId ami-12345678
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeregisterImage](#).

## Unregister-EC2PrivateIpAddress

다음 코드 예시에서는 Unregister-EC2PrivateIpAddress을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 네트워크 인터페이스에서 지정된 사설 IP 주소 할당을 취소합니다.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress 10.0.0.82
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UnassignPrivateIpAddresses](#).

## Unregister-EC2RouteTable

다음 코드 예시에서는 Unregister-EC2RouteTable을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 라우팅 테이블과 서브넷 간의 지정된 연결을 제거합니다.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisassociateRouteTable](#).

## 다음 도구를 사용한 Amazon ECR 예제 PowerShell

다음 코드 예제는 Amazon ECR과 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### Get-ECRLoginCommand

다음 코드 예시에서는 Get-ECRLoginCommand을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: IAM 보안 주체가 액세스할 수 있는 Amazon ECR 레지스트리를 인증하는 데 사용할 수 있는 로그인 정보가 포함된 PSObject를 반환합니다. 권한 부여 토큰을 얻기 위한 호출에 필요한 자격 증명과 리전 엔드포인트는 셀 기본값 (또는 cmdlet으로 설정) 에서 가져옵니다. **Set-AWSCredential/Set-DefaultAWSRegion Initialize-AWSDefaultConfiguration** Command 속성을 Invoke-Expression과 함께 사용하여 지정된 레지스트리에 로그인하거나 로그인이 필요한 다른 도구에서 반환된 자격 증명을 사용할 수 있습니다.

```
Get-ECRLoginCommand
```

출력:

```
Username      : AWS
Password      : eyJwYX1sb2Fk...kRBVEffS0VZIn0=
```

```
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
eyJwYX1sb2Fk...kRBVEFFS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

예 2: docker 로그인 명령에 대한 입력으로 사용하는 로그인 정보가 들어 있는 PSObject를 검색합니다. IAM 보안 주체가 해당 레지스트리에 액세스할 수 있는 한 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 Cmdlet 참조의 [Get-ECR을 LoginCommand](#) 참조하십시오.AWS Tools for PowerShell

## 다음 도구를 사용한 Amazon ECS 예제 PowerShell

다음 코드 예제는 Amazon ECS와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-ECSClusterDetail

다음 코드 예시에서는 Get-ECSClusterDetail을 사용하는 방법을 보여 줍니다.

## 예 1: 이 cmdlet은 하나 이상의 ECS 클러스터를 설명합니다.

예 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 Cmdlet 참조를 참조하십시오 [DescribeClusters](#).AWS Tools for PowerShell

## Get-ECSClusterList

다음 코드 예시에서는 Get-ECSClusterList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 Cmdlet 참조를 참조하십시오 [ListClusters](#).AWS Tools for PowerShell

## Get-ECSClusterService

다음 코드 예시에서는 Get-ECSClusterService을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예는 기본 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
Get-ECSClusterService
```

예 2: 이 예제는 지정된 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
Get-ECSClusterService -Cluster myCluster
```

예 3: 이 예에서는 한 번에 최대 10개의 서비스 세부 정보를 가져오면서 지정된 클러스터에서 실행 중인 서비스를 나열합니다.

```
$nextToken = $null
do
{
    Get-ECSClusterService -Cluster myCluster -MaxResult 10 -NextToken $nextToken
    $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListServices](#).

## Get-ECSService

다음 코드 예시에서는 Get-ECSService을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 기본 클러스터에서 특정 서비스의 세부 정보를 검색하는 방법을 보여줍니다.

```
Get-ECSService -Service my-http-service
```

예 2: 이 예제는 명명된 클러스터에서 실행 중인 특정 서비스의 세부 정보를 검색하는 방법을 보여 줍니다.

```
Get-ECSService -Cluster myCluster -Service my-http-service
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeServices](#).

## New-ECSCluster

다음 코드 예시에서는 New-ECSCluster을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 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 Cmdlet 참조를 참조하십시오 [CreateCluster](#).AWS Tools for PowerShell

## New-ECSService

다음 코드 예시에서는 New-ECSService을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예제 1: 이 예제 명령은 기본 클러스터에서 ecs-simple-service ``라는 서비스를 생성합니다. 이 서비스는 `ecs-demo` 작업 정의를 사용하며 해당 작업의 인스턴스 10개를 유지 관리합니다.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

예제 2: 이 예제 명령은 기본 클러스터의 로드 밸런서 뒤에 ``라는 서비스를 생성합니다. `ecs-simple-service` 이 서비스는 `ecs-demo` 작업 정의를 사용하며 해당 작업의 인스턴스 10개를 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [CreateService](#) AWS Tools for PowerShell

## Remove-ECSCluster

다음 코드 예시에서는 `Remove-ECSCluster`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 cmdlet은 지정된 ECS 클러스터를 삭제합니다. 클러스터를 삭제하려면 먼저 이 클러스터에서 모든 컨테이너 인스턴스를 등록 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 참조를 참조하십시오 [DeleteCluster](#).

## Remove-ECSService

다음 코드 예시에서는 `Remove-ECSService`을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 기본 클러스터에서 이름이 my-http-service "인 서비스를 삭제합니다. 서비스를 삭제하려면 먼저 원하는 수와 실행 횟수가 0이어야 합니다. 명령이 진행되기 전에 확인 메시지가 표시됩니다. 확인 프롬프트를 건너뛰려면 -Force 스위치를 추가하십시오.

```
Remove-ECSService -Service my-http-service
```

예 2: 명명된 클러스터에서 이름이 my-http-service "인 서비스를 삭제합니다.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteService](#).

## Update-ECSClusterSetting

다음 코드 예시에서는 Update-ECSClusterSetting을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 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 Cmdlet 참조를 참조하십시오 [UpdateClusterSettings](#). AWS Tools for PowerShell

## Update-ECSService

다음 코드 예시에서는 Update-ECSService을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예제 1: 이 예제 명령은 my-http-service ``태스크 정의를 사용하도록 amazon-ecs-sample`` 서비스를 업데이트합니다.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

예 2: 이 예제 명령은 my-http-service ``서비스의 원하는 수를 10으로 업데이트합니다.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateService](#).

## 다음 도구를 사용한 Amazon EFS 예제 PowerShell

다음 코드 예제는 Amazon EFS와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

### 주제

- [작업](#)

## 작업

### Edit-EFSMountTargetSecurityGroup

다음 코드 예시에서는 Edit-EFSMountTargetSecurityGroup을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 탑재 대상에 유효한 보안 그룹을 업데이트합니다. "sg-xxxxxxx" 형식으로 최대 5개까 지 지정할 수 있습니다.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-group1,sg-group3
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.  
[ModifyMountTargetSecurityGroups](#) AWS Tools for PowerShell

### Get-EFSFileSystem

다음 코드 예시에서는 Get-EFSFileSystem을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 해당 지역의 발신자 계정이 소유한 모든 파일 시스템의 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
```

...

예 2: 지정된 파일 시스템의 세부 정보를 반환합니다.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

예 3: 파일 시스템 생성 시 지정된 동일성 생성 토큰을 사용하여 파일 시스템의 세부 정보를 반환합니다.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DescribeFileSystems](#).AWS Tools for PowerShell

## Get-EFSMountTarget

다음 코드 예시에서는 Get-EFSMountTarget을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 파일 시스템과 관련된 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 참조를 참조하십시오 [DescribeMountTargets](#).

## Get-EFSMountTargetSecurityGroup

다음 코드 예시에서는 Get-EFSMountTargetSecurityGroup을 사용하는 방법을 보여 줍니다.

## 예 1: Get-EFSMountTargetSecurityGroup

예 1: 탑재 대상과 관련된 네트워크 인터페이스에 현재 할당된 보안 그룹의 ID를 반환합니다.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

출력:

```
sg-1a2b3c4d
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMountTargetSecurityGroups](#).

## Get-EFSTag

다음 코드 예시에서는 Get-EFSTag을 사용하는 방법을 보여 줍니다.

## 예 1: Get-EFSTag

예 1: 지정된 파일 시스템과 현재 연결된 태그 컬렉션을 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 참조를 참조하십시오 [DescribeTags](#).

## New-EFSFileSystem

다음 코드 예시에서는 New-EFSFileSystem을 사용하는 방법을 보여 줍니다.

## 예 1: New-EFSFileSystem

예 1: 비어 있는 새 파일 시스템을 만듭니다. idempotent 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
```

예 2: 동일성 생성을 보장하기 위해 사용자 지정 토큰을 사용하여 비어 있는 새 파일 시스템을 만듭니다.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CreateFileSystem](#). AWS Tools for PowerShell

## New-EFSMountTarget

다음 코드 예시에서는 New-EFSMountTarget을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 파일 시스템의 새 탑재 대상을 만듭니다. 지정된 서브넷은 탑재 대상이 생성될 VPC (Virtual Private Cloud) 와 서브넷의 주소 범위에서 자동 할당될 IP 주소를 결정하는 데 사용됩니다. 할당된 IP 주소를 사용하여 이 파일 시스템을 Amazon EC2 인스턴스에 마운트할 수 있습니다. 보안 그룹이 지정되지 않았으므로 대상에 대해 생성된 네트워크 인터페이스는 서브넷 VPC의 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

```

예 2: 자동 할당된 IP 주소를 사용하여 지정된 파일 시스템에 대한 새 탑재 대상을 생성합니다. 탑재 대상에 대해 생성된 네트워크 인터페이스는 지정된 보안 그룹과 연결됩니다 (“sg-xxxxxxx” 형식으로 최대 5개까지 지정할 수 있음).

```

New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3

```

예 3: 지정된 IP 주소를 사용하여 지정된 파일 시스템에 대한 새 탑재 대상을 만듭니다.

```

New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
10.0.0.131

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateMountTarget](#).

## New-EFSTag

다음 코드 예시에서는 New-EFSTag을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 파일 시스템에 태그 컬렉션을 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"}

```

예 2: 지정된 파일 시스템의 이름 태그를 설정합니다. 이 값은 Get-EFS FileSystem 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 Cmdlet 참조를 참조하십시오 [CreateTags](#).AWS Tools for PowerShell

## Remove-EFSFileSystem

다음 코드 예시에서는 Remove-EFSFileSystem을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 더 이상 사용하지 않는 지정된 파일 시스템을 삭제합니다 (파일 시스템에 탑재 대상이 있는 경우 먼저 제거해야 함). cmdlet이 진행되기 전에 확인 메시지가 표시됩니다. 확인을 표시하지 않으려면 스위치를 사용하십시오. **-Force**

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteFileSystem](#).AWS Tools for PowerShell

## Remove-EFSMountTarget

다음 코드 예시에서는 Remove-EFSMountTarget을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 지정된 탑재 대상을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 프롬프트를 표시하지 않으려면 스위치를 사용하십시오. **-Force** 이 작업을 수행하면 타겟을 통한 파일 시스템 마운트가 강제로 중단된다는 점에 유의하십시오. 가능하면 이 명령을 실행하기 전에 파일 시스템을 마운트 해제하는 것이 좋습니다.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [DeleteMountTarget](#)AWS Tools for PowerShell

## Remove-EFSTag

다음 코드 예시에서는 Remove-EFSTag을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 파일 시스템에서 하나 이상의 태그 컬렉션을 삭제합니다. cmdlet이 진행되기 전에 확인을 요청하는 메시지가 표시됩니다. 확인을 표시하지 않으려면 스위치를 사용하십시오. **-Force**

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteTags](#).AWS Tools for PowerShell

## 다음 도구를 사용한 Amazon EKS 예제 PowerShell

다음 코드 예제는 Amazon EKS와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Add-EKSResourceTag

다음 코드 예시에서는 Add-EKSResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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"}
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오. [TagResource](#)AWS Tools for PowerShell

## Get-EKSCluster

다음 코드 예시에서는 Get-EKSCluster을 사용하는 방법을 보여 줍니다.

예 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 Cmdlet 참조를 참조하십시오 [DescribeCluster](#).AWS Tools for PowerShell

## Get-EKSClusterList

다음 코드 예시에서는 Get-EKSClusterList을 사용하는 방법을 보여 줍니다.

예 1: 이 cmdlet은 지정된 AWS 계정 지역의 Amazon EKS 클러스터를 나열합니다.

```
Get-EKSClusterList
```

출력:

```
PROD
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ListClusters](#).AWS Tools for PowerShell

## Get-EKSFargateProfile

다음 코드 예시에서는 Get-EKSFargateProfile을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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 Cmdlet 참조를 참조하십시오 [DescribeFargateProfile](#).AWS Tools for PowerShell

## Get-EKSFargateProfileList

다음 코드 예시에서는 Get-EKSFargateProfileList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 지정된 지역의 지정된 클러스터와 연결된 AWS Fargate 프로필을 나열합니다.  
AWS 계정

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

출력:

```
EKSFargate
EKSFargateProfile
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오 [ListFargateProfiles](#). AWS Tools for PowerShell

## Get-EKSNodegroup

다음 코드 예시에서는 Get-EKSNodegroup을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 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 Cmdlet 참조를 참조하십시오 [DescribeNodegroup](#).AWS Tools for PowerShell

## Get-EKSNodegroupList

다음 코드 예시에서는 Get-EKSNodegroupList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 지정된 지역의 지정된 클러스터와 연결된 Amazon EKS 노드 그룹을 나열합니다.  
AWS 계정

```
Get-EKSNodegroupList -ClusterName PROD
```

출력:

```
ProdEKSNodeGroup
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ListNodegroups](#).AWS Tools for PowerShell

## Get-EKSResourceTag

다음 코드 예시에서는 Get-EKSResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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 Cmdlet 참조를 참조하십시오 [ListTagsForResource](#).AWS Tools for PowerShell

## Get-EKSUpdate

다음 코드 예시에서는 Get-EKSUpdate을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 Amazon EKS 클러스터 또는 관련 관리형 노드 그룹에 대한 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 Cmdlet 참조를 참조하십시오 [DescribeUpdate](#).AWS Tools for PowerShell

## Get-EKSUpdateList

다음 코드 예시에서는 Get-EKSUpdateList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 지정된 지역의 Amazon AWS 계정 EKS 클러스터 또는 관리형 노드 그룹과 관련된 업데이트를 나열합니다.

```
Get-EKSUpdateList -Name "PROD"
```

출력:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [ListUpdates](#).AWS Tools for PowerShell

## New-EKSCluster

다음 코드 예시에서는 New-EKSCluster을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 '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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateCluster](#).

## New-EKSFargateProfile

다음 코드 예시에서는 New-EKSFargateProfile을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 Amazon EKS 클러스터를 위한 AWS Fargate 프로필을 생성합니다. Fargate 인프라에서 포드를 스케줄링하려면 클러스터에 Fargate 프로필이 하나 이상 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 Cmdlet 참조를 참조하십시오 [CreateFargateProfile](#).AWS Tools for PowerShell

**New-EKSNodeGroup**

다음 코드 예시에서는 New-EKSNodeGroup을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

예 1: 이 cmdlet은 Amazon EKS 클러스터를 위한 관리형 작업자 노드 그룹을 생성합니다. 클러스터에 대한 현재 Kubernetes 버전과 동일한 클러스터에 대해서만 노드 그룹을 생성할 수 있습니다. 모든 노드 그룹은 클러스터의 각 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

```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CreateNodegroup](#).AWS Tools for PowerShell

## Remove-EKSCluster

다음 코드 예시에서는 Remove-EKSCluster을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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.y14.us-west-2.eks.amazonaws.com

```

```

Identity      : Amazon.EKS.Model.Identity
Logging       : Amazon.EKS.Model.Logging
Name          : DEV-KUBE-CL
PlatformVersion : eks.7
ResourcesVpcConfig : Amazon.EKS.Model.VpcConfigResponse
RoleArn       : arn:aws:iam::012345678912:role/eks-iam-role
Status        : DELETING
Tags          : {}
Version       : 1.14

```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeleteCluster](#).AWS Tools for PowerShell

## Remove-EKSFargateProfile

다음 코드 예시에서는 Remove-EKSFargateProfile을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 cmdlet은 AWS Fargate 프로필을 삭제합니다. Fargate 프로필을 삭제하면 해당 프로필로 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 : {}
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DeleteFargateProfile](#).AWS Tools for PowerShell

## Remove-EKSNodegroup

다음 코드 예시에서는 Remove-EKSNodegroup을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 cmdlet은 클러스터의 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 Cmdlet 참조를 참조하십시오 [DeleteNodegroup](#).AWS Tools for PowerShell

## Remove-EKSResourceTag

다음 코드 예시에서는 Remove-EKSResourceTag을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 Cmdlet 참조를 참조하십시오 [UntagResource](#).AWS Tools for PowerShell

## Update-EKSClusterConfig

다음 코드 예시에서는 Update-EKSClusterConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: Amazon EKS 클러스터 구성을 업데이트합니다. 클러스터는 업데이트 중에도 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 참조를 참조하십시오 [UpdateClusterConfig](#).

## Update-EKSClusterVersion

다음 코드 예시에서는 Update-EKSClusterVersion을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 cmdlet은 Amazon EKS 클러스터를 지정된 쿠버네티스 버전으로 업데이트합니다. 클러스터는 업데이트 중에도 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 참조를 참조하십시오 [UpdateClusterVersion](#).

## Elastic Load Balancing - 다음 도구를 사용한 버전 1 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with Elastic Load Balancing - 버전 1을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

## 작업

### Add-ELBLoadBalancerToSubnet

다음 코드 예시에서는 Add-ELBLoadBalancerToSubnet을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서에 구성된 서브넷 세트에 지정된 서브넷을 추가합니다. 출력에는 서브넷의 전체 목록이 포함됩니다.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

출력:

```
subnet-12345678
subnet-87654321
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AttachLoadBalancerToSubnets](#).

### Add-ELBResourceTag

다음 코드 예시에서는 Add-ELBResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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" }
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 Tag 매개 변수에 대한 태그를 만들어야 합니다.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AddTags](#).

## Disable-ELBAvailabilityZoneForLoadBalancer

다음 코드 예시에서는 Disable-ELBAvailabilityZoneForLoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예시는 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 참조를 참조하십시오 [DisableAvailabilityZonesForLoadBalancer](#).

## Dismount-ELBLoadBalancerFromSubnet

다음 코드 예시에서는 Dismount-ELBLoadBalancerFromSubnet을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 로드 밸런서에 구성된 서브넷 세트에서 지정된 서브넷을 제거합니다. 출력에는 나머지 서브넷이 포함됩니다.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
subnet-12345678
```

출력:

```
subnet-87654321
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachLoadBalancerFromSubnets](#).

## Edit-ELBLoadBalancerAttribute

다음 코드 예시에서는 Edit-ELBLoadBalancerAttribute을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예시에서는 지정된 로드 밸런서에 대해 영역 간 부하 분산을 활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

예 2: 이 예시에서는 지정된 로드 밸런서의 연결 드레이닝을 비활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

예 3: 이 예시에서는 지정된 로드 밸런서에 대한 액세스 로깅을 활성화합니다.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName my-logs-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyLoadBalancerAttributes](#).

## Enable-ELBAvailabilityZoneForLoadBalancer

다음 코드 예시에서는 Enable-ELBAvailabilityZoneForLoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제는 지정된 가용 영역을 지정된 로드 밸런서에 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 참조를 참조하십시오 [EnableAvailabilityZonesForLoadBalancer](#).

## Get-ELBInstanceHealth

다음 코드 예시에서는 Get-ELBInstanceHealth을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제는 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 등록된 지정된 인스턴스의 상태를 설명합니다.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

예 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 참조를 참조하십시오 [DescribeInstanceHealth](#).

## Get-ELBLoadBalancer

다음 코드 예시에서는 Get-ELBLoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예시는 로드 밸런서의 이름을 나열합니다.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

출력:

```
LoadBalancerName
-----
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

예 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

```

예 3: 이 예시에서는 현재 AWS 지역의 모든 로드 밸런서를 설명합니다.

```
Get-ELBLoadBalancer
```

예 4: 이 예시에서는 사용 가능한 모든 로드 밸런서를 모두 설명합니다. AWS 리전

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeLoadBalancers](#).

## Get-ELBLoadBalancerAttribute

다음 코드 예시에서는 Get-ELBLoadBalancerAttribute을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 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 참조를 참조하십시오  
[DescribeLoadBalancerAttributes](#).

## Get-ELBLoadBalancerPolicy

다음 코드 예시에서는 Get-ELBLoadBalancerPolicy을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 로드 밸런서와 관련된 정책을 설명합니다.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

출력:

```

PolicyAttributeDescriptions      PolicyName
PolicyTypeName
-----
-----
{ProxyProtocol}                  my-ProxyProtocol-policy
ProxyProtocolPolicyType
{CookieName}                      my-app-cookie-policy
AppCookieStickinessPolicyType

```

예 2: 이 예제에서는 지정된 정책의 속성을 설명합니다.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-ProxyProtocol-policy).PolicyAttributeDescriptions
```

출력:

```

AttributeName      AttributeValue
-----
ProxyProtocol       true

```

예 3: 이 예에서는 샘플 정책을 포함하여 사전 정의된 정책을 설명합니다. 샘플 정책 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [DescribeLoadBalancerPolicies](#).AWS Tools for PowerShell

## Get-ELBLoadBalancerPolicyType

다음 코드 예시에서는 Get-ELBLoadBalancerPolicyType을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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

```

예 2: 이 예제에서는 지정된 정책 유형을 설명합니다.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

**출력:**

```

Description                                PolicyAttributeTypeDescriptions
PolicyTypeName
-----
-----
Policy that controls whether to incl... {ProxyProtocol}
  ProxyProtocolPolicyType

```

예 3: 이 예에서는 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 참조를 참조하십시오 [DescribeLoadBalancerPolicyTypes](#).

## Get-ELBResourceTag

다음 코드 예시에서는 Get-ELBResourceTag을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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}

예 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 참조를 참조하십시오 [DescribeTags](#).

## Join-ELBSecurityGroupToLoadBalancer

다음 코드 예시에서는 Join-ELBSecurityGroupToLoadBalancer을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 로드 밸런서의 현재 보안 그룹을 지정된 보안 그룹으로 바꿉니다.

예 1: 이 예제는 지정된 로드 밸런서의 현재 보안 그룹을 지정된 보안 그룹으로 바꿉니다.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

출력:

```
sg-87654321
```

예 2: 현재 보안 그룹을 유지하고 추가 보안 그룹을 지정하려면 기존 보안 그룹과 새 보안 그룹을 모두 지정하십시오.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

출력:

```
sg-12345678
sg-87654321
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ApplySecurityGroupsToLoadBalancer](#).

## New-ELBAppCookieStickinessPolicy

다음 코드 예시에서는 New-ELBAppCookieStickinessPolicy을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 애플리케이션 생성 쿠키의 고정 세션 수명을 따르는 고정성 정책을 생성합니다.

예 1: 이 예제는 지정된 애플리케이션 생성 쿠키의 고정 세션 수명을 따르는 고정성 정책을 생성합니다.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [CreateAppCookieStickinessPolicy](#) AWS Tools for PowerShell

## New-ELBLCookieStickinessPolicy

다음 코드 예시에서는 New-ELBLCookieStickinessPolicy을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 지정된 만료 기간 (초) 으로 제어되는 고정 세션 수명을 포함하는 고정성 정책을 생성합니다.

```
New-ELBLCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy -CookieExpirationPeriod 60
```

예 2: 이 예제는 고정 세션 수명을 브라우저 (사용자 에이전트) 의 수명에 의해 제어되는 고정성 정책을 생성합니다.

```
New-ELBLCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-duration-cookie-policy
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오. [CreateLbCookieStickinessPolicy](#) AWS Tools for PowerShell

## New-ELBLoadBalancer

다음 코드 예시에서는 New-ELBLoadBalancer을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예제 1: 이 예제는 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 예제는 EC2-Classic에서 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
```

예 3: 이 예시에서는 HTTPS 리스너를 사용하여 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 Cmdlet 참조를 참조하십시오 [CreateLoadBalancer](#).AWS Tools for PowerShell

## New-ELBLoadBalancerListener

다음 코드 예시에서는 New-ELBLoadBalancerListener를 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제는 지정된 로드 밸런서에 HTTPS 리스너를 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 Cmdlet 참조를 참조하십시오 [CreateLoadBalancerListeners](#). AWS Tools for PowerShell

## New-ELBLoadBalancerPolicy

다음 코드 예시에서는 New-ELBLoadBalancerPolicy을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [CreateLoadBalancerPolicy](#).

## Register-ELBInstanceWithLoadBalancer

다음 코드 예시에서는 Register-ELBInstanceWithLoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [RegisterInstancesWithLoadBalancer](#). AWS Tools for PowerShell

## Remove-ELBInstanceFromLoadBalancer

다음 코드 예시에서는 Remove-ELBInstanceFromLoadBalancer을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서에서 지정된 EC2 인스턴스를 제거합니다. Force 파라미터도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeregisterInstancesFromLoadBalancer](#).

## Remove-ELBLoadBalancer

다음 코드 예시에서는 Remove-ELBLoadBalancer을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서를 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteLoadBalancer](#).

## Remove-ELBLoadBalancerListener

다음 코드 예시에서는 Remove-ELBLoadBalancerListener을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서의 포트 80에서 리스너를 삭제합니다. Force 매개 변수도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteLoadBalancerListeners](#).

## Remove-ELBLoadBalancerPolicy

다음 코드 예시에서는 Remove-ELBLoadBalancerPolicy을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예시는 지정된 로드 밸런서에서 지정된 정책을 삭제합니다. Force 매개 변수도 함께 지정하지 않는 한 작업이 진행되기 전에 확인 메시지가 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 참조를 참조하십시오 [DeleteLoadBalancerPolicy](#).

## Remove-ELBResourceTag

다음 코드 예시에서는 Remove-ELBResourceTag을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 로드 밸런서에서 지정된 태그를 제거합니다. Force 매개 변수도 지정하지 않은 경우 작업이 진행되기 전에 확인 메시지가 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"):
```

예 2: Powershell 버전 2에서는 New-Object를 사용하여 태그 매개 변수에 대한 태그를 만들어야 합니다.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [RemoveTags](#).AWS Tools for PowerShell

## Set-ELBHealthCheck

다음 코드 예시에서는 Set-ELBHealthCheck을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 로드 밸런서의 상태 점검 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 Cmdlet 참조를 참조하십시오 [ConfigureHealthCheck](#).AWS Tools for PowerShell

## Set-ELBLoadBalancerListenerSSLCertificate

다음 코드 예시에서는 Set-ELBLoadBalancerListenerSSLCertificate을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 리스너의 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 Cmdlet 참조를 참조하십시오 [SetLoadBalancerListenerSslCertificate](#).AWS Tools for PowerShell

## Set-ELBLoadBalancerPolicyForBackendServer

다음 코드 예시에서는 Set-ELBLoadBalancerPolicyForBackendServer을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 포트의 정책을 지정된 정책으로 대체합니다.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

예 2: 이 예에서는 지정된 포트와 관련된 모든 정책을 제거합니다.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SetLoadBalancerPoliciesForBackendServer](#).

## Set-ELBLoadBalancerPolicyOfListener

다음 코드 예시에서는 Set-ELBLoadBalancerPolicyOfListener을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 지정된 리스너의 정책을 지정된 정책으로 대체합니다.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

예 2: 이 예제는 지정된 리스너와 관련된 모든 정책을 제거합니다.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SetLoadBalancerPoliciesOfListener](#).

## Elastic Load Balancing - 다음 도구를 사용한 버전 2 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with Elastic Load Balancing - 버전 2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Add-ELB2ListenerCertificate

다음 코드 예시에서는 Add-ELB2ListenerCertificate을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [AddListenerCertificates](#).

## Add-ELB2Tag

다음 코드 예시에서는 Add-ELB2Tag을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [AddTags](#).

## Edit-ELB2Listener

다음 코드 예시에서는 Edit-ELB2Listener을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제는 지정된 리스너의 기본 동작을 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 Cmdlet 참조를 참조하십시오. [ModifyListener](#) AWS Tools for PowerShell

**Edit-ELB2LoadBalancerAttribute**

다음 코드 예시에서는 Edit-ELB2LoadBalancerAttribute을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [ModifyLoadBalancerAttributes](#).AWS Tools for PowerShell

## Edit-ELB2Rule

다음 코드 예시에서는 Edit-ELB2Rule을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [ModifyRule](#).AWS Tools for PowerShell

## Edit-ELB2TargetGroup

다음 코드 예시에서는 Edit-ELB2TargetGroup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [ModifyTargetGroup](#).

## Edit-ELB2TargetGroupAttribute

다음 코드 예시에서는 Edit-ELB2TargetGroupAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 대상 그룹의 등록 취소\_지연 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 Cmdlet 참조를 참조하십시오. [ModifyTargetGroupAttributes](#) AWS Tools for PowerShell

## Get-ELB2AccountLimit

다음 코드 예시에서는 Get-ELB2AccountLimit을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 명령은 특정 지역의 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 참조를 참조하십시오 [DescribeAccountLimits](#).

## Get-ELB2Listener

다음 코드 예시에서는 Get-ELB2Listener을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 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 Cmdlet 참조를 참조하십시오. [DescribeListeners](#) AWS Tools for PowerShell

## Get-ELB2ListenerCertificate

다음 코드 예시에서는 Get-ELB2ListenerCertificate을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeListenerCertificates](#).

## Get-ELB2LoadBalancer

다음 코드 예시에서는 Get-ELB2LoadBalancer을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 해당 지역의 모든 로드 밸런서를 표시합니다.

```
Get-ELB2LoadBalancer
```

출력:

```
AvailabilityZones      : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFT0TI
```

```

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 참조를 참조하십시오 [DescribeLoadBalancers](#).

## Get-ELB2LoadBalancerAttribute

다음 코드 예시에서는 Get-ELB2LoadBalancerAttribute을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 명령은 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 참조를 참조하십시오 [DescribeLoadBalancerAttributes](#).

## Get-ELB2Rule

다음 코드 예시에서는 Get-ELB2Rule을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제에서는 지정된 리스너 ARN에 대한 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 Cmdlet 참조를 참조하십시오 [DescribeRules](#).AWS Tools for PowerShell

## Get-ELB2SSLPolicy

다음 코드 예시에서는 Get-ELB2SSLPolicy을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 ElasticLoadBalancing V2에 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 참조를 참조하십시오 [DescribeSslPolicies](#).

**Get-ELB2Tag**

다음 코드 예시에서는 Get-ELB2Tag을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 리소스의 태그를 나열합니다.

예 1: 이 예제는 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 참조를 참조하십시오 [DescribeTags](#).

## Get-ELB2TargetGroup

다음 코드 예시에서는 Get-ELB2TargetGroup을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 대상 그룹을 설명합니다.

예 1: 이 예에서는 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 참조를 참조하십시오 [DescribeTargetGroups](#).

## Get-ELB2TargetGroupAttribute

다음 코드 예시에서는 Get-ELB2TargetGroupAttribute을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 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 참조를 참조하십시오 [DescribeTargetGroupAttributes](#).

## Get-ELB2TargetHealth

다음 코드 예시에서는 Get-ELB2TargetHealth을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 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 참조를 참조하십시오 [DescribeTargetHealth](#).

## New-ELB2Listener

다음 코드 예시에서는 New-ELB2Listener을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 기본 작업 'Forward'를 사용하여 새 ALB 리스너를 생성하여 지정된 대상 그룹으로 트래픽을 전송합니다.

```
$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 참조를 참조하십시오 [CreateListener](#).

**New-ELB2LoadBalancer**

다음 코드 예시에서는 New-ELB2LoadBalancer을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

- 예 1: 이 예에서는 서브넷 2개가 있는 새로운 인터넷 연결 애플리케이션 로드 밸런서를 만듭니다.

```

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 : Z35SXD0TRQ7X7K
CreatedTime       : 12/28/19 2:58:03 PM
DNSName           : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType     : ipv4
LoadBalancerArn   : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName  : New-Test-ALB
Scheme            : internet-facing
SecurityGroups    : {sg-07c3414abb8811cbd}
State             : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type              : application
VpcId             : vpc-2cfd7000

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateLoadBalancer](#).

## New-ELB2Rule

다음 코드 예시에서는 New-ELB2Rule을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 리스너의 고객 헤더 값을 기반으로 고정 응답 조치가 포함된 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 Cmdlet 참조를 참조하십시오 [CreateRule](#).AWS Tools for PowerShell

## New-ELB2TargetGroup

다음 코드 예시에서는 New-ELB2TargetGroup을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 제공된 파라미터를 사용하여 새 Target 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 참조를 참조하십시오 [CreateTargetGroup](#).

## Register-ELB2Target

다음 코드 예시에서는 Register-ELB2Target을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예에서는 'i-0672a4c4cdeae3111' 인스턴스를 지정된 대상 그룹에 등록합니다.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id = 'i-0672a4c4cdeae3111'}
```

- API에 [RegisterTargets](#) 대한 AWS Tools for PowerShell 세부 정보는 Cmdlet 참조를 참조하십시오.

## Remove-ELB2Listener

다음 코드 예시에서는 Remove-ELB2Listener을 사용하는 방법을 보여 줍니다.

예 1: 이 예제는 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
```

예 2: 이 예제는 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 참조를 참조하십시오 [DeleteListener](#).

## Remove-ELB2ListenerCertificate

다음 코드 예시에서는 Remove-ELB2ListenerCertificate을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [RemoveListenerCertificates](#).

## Remove-ELB2LoadBalancer

다음 코드 예시에서는 Remove-ELB2LoadBalancer을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [DeleteLoadBalancer](#).

**Remove-ELB2Rule**

다음 코드 예시에서는 Remove-ELB2Rule을 사용하는 방법을 보여 줍니다.

**예 대한 도구 PowerShell**

예 1: 이 예제는 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 참조를 참조하십시오 [DeleteRule](#).

**Remove-ELB2Tag**

다음 코드 예시에서는 Remove-ELB2Tag을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 지정된 키의 태그를 제거합니다.

예 1: 이 예제는 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 참조를 참조하십시오 [RemoveTags](#).

## Remove-ELB2TargetGroup

다음 코드 예시에서는 Remove-ELB2TargetGroup을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 대상 그룹을 제거합니다.

예 1: 이 예에서는 지정된 대상 그룹을 제거합니다.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/testsssss/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/
testsssss/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 참조를 참조하십시오 [DeleteTargetGroup](#).

## Set-ELB2IpAddressType

다음 코드 예시에서는 Set-ELB2IpAddressType을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 로드 밸런서 IP 주소 유형을 'IPv4'에서 "로 변경합니다. DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType dualstack
```

출력:

```
Value
-----
dualstack
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [SetIpAddressType](#).AWS Tools for PowerShell

## Set-ELB2RulePriority

다음 코드 예시에서는 Set-ELB2RulePriority을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [SetRulePriorities](#).

## Set-ELB2SecurityGroup

다음 코드 예시에서는 Set-ELB2SecurityGroup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예에서는 보안 그룹 'sg-07c3414abb8811cbd'를 지정된 로드 밸런서에 추가합니다.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
'sg-07c3414abb8811cbd'
```

출력:

```
sg-07c3414abb8811cbd
```

- API에 [SetSecurityGroups AWS Tools for PowerShell](#) 대한 세부 정보는 Cmdlet 참조를 참조하십시오.

## Set-ELB2Subnet

다음 코드 예시에서는 Set-ELB2Subnet을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [SetSubnets](#).AWS Tools for PowerShell

## Unregister-ELB2Target

다음 코드 예시에서는 Unregister-ELB2Target을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제는 지정된 타겟 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'
```

- API에 [DeregisterTargets](#)대한 AWS Tools for PowerShell 세부 정보는 Cmdlet 참조를 참조하십시오.

## 다음과 같은 도구를 사용하는 Amazon FSx 예제 PowerShell

다음 코드 예제는 Amazon FSx와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 GitHub 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다.

주제

- [작업](#)

## 작업

### Add-FSXResourceTag

다음 코드 예시에서는 Add-FSXResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [TagResource](#).

### Get-FSXBackup

다음 코드 예시에서는 Get-FSXBackup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [DescribeBackups](#).

## Get-FSXFileSystem

다음 코드 예시에서는 Get-FSXFileSystem를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [DescribeFileSystems](#).AWS Tools for PowerShell

## Get-FSXResourceTagList

다음 코드 예시에서는 Get-FSXResourceTagList를 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 도구 PowerShell

예 1: 이 예제는 제공된 리소스 arn의 태그를 나열합니다.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

출력:

Key	Value
---	-----
FSx-Service	Windows
Users	Dev

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTagsForResource](#).

## New-FSXBackup

다음 코드 예시에서는 New-FSXBackup을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 도구 PowerShell

예 1: 이 예제는 지정된 파일 시스템의 백업을 만듭니다.

```
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 참조를 참조하십시오 [CreateBackup](#).

## New-FSXFileSystem

다음 코드 예시에서는 New-FSXFileSystem를 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 서브넷에서의 액세스를 허용하고 초당 최대 8MB의 처리량을 지원하는 새로운 300GB Windows 파일 시스템을 생성합니다. 새 파일 시스템은 지정된 Microsoft Active Directory에 자동으로 가입됩니다.

```
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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateFileSystem](#).

## New-FSXFileSystemFromBackup

다음 코드 예시에서는 New-FSXFileSystemFromBackup를 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 Windows File Server용 기존 Amazon FSx 백업에서 새 Amazon FSx 파일 시스템을 생성합니다.

예 1: 이 예제는 Windows File Server용 기존 Amazon FSx 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 Cmdlet 참조를 참조하십시오. [CreateFileSystemFromBackup](#) AWS Tools for PowerShell

## Remove-FSXBackup

다음 코드 예시에서는 Remove-FSXBackup을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 지정된 백업 ID를 제거합니다.

예 1: 이 예에서는 지정된 백업 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 Cmdlet 참조를 참조하십시오 [DeleteBackup](#).AWS Tools for PowerShell

## Remove-FSXFileSystem

다음 코드 예시에서는 Remove-FSXFileSystem을 사용하는 방법을 보여 줍니다.

예 1: 이 예에서는 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteFileSystem](#).

## Remove-FSXResourceTag

다음 코드 예시에서는 Remove-FSXResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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 Cmdlet 참조를 참조하십시오 [UntagResource](#).AWS Tools for PowerShell

## Update-FSXFileSystem

다음 코드 예시에서는 Update-FSXFileSystem을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 를 통해 FSX 파일 시스템 자동 백업 보존 날짜를 업데이트합니다.

UpdateFileSystemWindowsConfiguration

```
$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 참조를 참조하십시오 [UpdateFileSystem](#).

## AWS Glue 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS Glue. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스  
함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼  
수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코  
드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되  
어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### New-GLUEJob

다음 코드 예시에서는 New-GLUEJob을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제는 AWS Glue에서 새 작업을 생성합니다. 명령 이름 값은 항상 **glueet1**. AWS Glue는 Python 또는 Scala로 작성된 작업 스크립트 실행을 지원합니다. 이 예제에서 작업 스크립트 (MyTestGlueJob.py) 는 Python으로 작성되었습니다. Python 매개 변수는 **\$DefArgs** 변수에 지정된 다음 해시 테이블을 허용하는 **DefaultArguments** 매개 변수의 PowerShell 명령에 전달됩니다. **\$JobParams** 변수의 매개 변수는 API에서 가져온 것으로, CreateJob AWS Glue API 참조의 작업 (<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html>) 주제에 설명되어 있습니다.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueet1'
$Command.ScriptLocation = 's3://aws-glue-scripts-000000000000-us-west-2/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
    '--TempDir' = 's3://aws-glue-temporary-000000000000-us-west-2/admin'
    '--job-bookmark-option' = 'job-bookmark-disable'
    '--job-language' = 'python'
}
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity" = "5"
    "Command" = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments" = $DefArgs
```

```

"Description"      = "This is a test"
"ExecutionProperty" = $ExecutionProp
"MaxRetries"      = "1"
"Name"            = "MyOregonTestGlueJob"
"Role"            = "Amazon-GlueServiceRoleForSSM"
"Timeout"         = "20"
}

```

```
New-GlueJob @JobParams
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateJob](#).

## AWS Health 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Health. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-HLTHEvent

다음 코드 예시에서는 Get-HLTHEvent을 사용하는 방법을 보여 줍니다.

예 1: 이 명령은 AWS Personal Health Dashboard에서 이벤트를 반환합니다. 사용자는 -Region 매개 변수를 추가하여 미국 동부 (버지니아 북부) 지역의 서비스에 사용할 수 있는 이벤트를 볼 수 있습니다.

지만 `-Filter_Region` 매개 변수는 EU (런던) 및 미국 서부 (오레곤) 지역 (`eu-west-2` 및 `us-west-2`) 에서 기록되는 이벤트를 필터링합니다. `-Filter_StartTime` 매개 변수는 이벤트가 시작될 수 있는 시간 범위를 필터링하는 반면, `-Filter_EndTime` 매개 변수는 이벤트가 종료될 수 있는 시간 범위를 필터링합니다. 그 결과 지정된 `-Filter_` 범위 내에서 시작하여 예약된 `-Filter_StartTime` 범위 내에서 종료 되는 RDS에 대한 예약된 유지 관리 이벤트가 생성됩니다. `EndTime`

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [DescribeEvents](#) AWS Tools for PowerShell

## 다음과 같은 도구를 사용한 IAM 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with IAM을 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Add-IAMClientIDToOpenIDConnectProvider

다음 코드 예시에서는 Add-IAMClientIDToOpenIDConnectProvider를 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AddClientIdToOpenIdConnectProvider](#).

### Add-IAMRoleTag

다음 코드 예시에서는 Add-IAMRoleTag를 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제는 ID 관리 서비스의 역할에 태그를 추가합니다.

```
Add-IAMRoleTag -RoleName AdminRoleaccess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [TagRole](#).

### Add-IAMRoleToInstanceProfile

다음 코드 예시에서는 Add-IAMRoleToInstanceProfile를 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AddRoleToInstanceProfile](#).

## Add-IAMUserTag

다음 코드 예시에서는 Add-IAMUserTag을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예제 1: 이 예제는 ID 관리 서비스의 사용자에게 태그를 추가합니다.

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing' }
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [TagUser](#).

## Add-IAMUserToGroup

다음 코드 예시에서는 Add-IAMUserToGroup을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예제 1: 이 명령은 **Bob**이라는 사용자를 **Admins**라는 그룹에 추가합니다.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AddUserToGroup](#).

## Disable-IAMMFADevice

다음 코드 예시에서는 Disable-IAMMFADevice을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeactivateMfaDevice](#).

## Edit-IAMPassword

다음 코드 예시에서는 Edit-IAMPassword을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 명령은 명령을 실행하는 사용자의 암호를 변경합니다. 이 명령은 IAM 사용자만 호출할 수 있습니다. AWS 계정 (루트) 자격 증명으로 로그인할 때 이 명령을 호출하면 명령이 오류를 반환합니다. **InvalidUserType**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [ChangePassword](#).AWS Tools for PowerShell

## Enable-IAMMFADevice

다음 코드 예시에서는 Enable-IAMMFADevice을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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 이 명령은 가상 MFA 디바이스의 처음 두 코드를 순서대로 포함하여 디바이스를 동기화합니다. AWS

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber $MFADevice.SerialNumber
-AuthenticationCode1 "24681357" -AuthenticationCode2
"13572468"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [EnableMfaDevice](#).

## Get-IAMAccessKey

다음 코드 예시에서는 Get-IAMAccessKey을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 명령은 **Bob**이라는 IAM 사용자의 액세스 키를 나열합니다. IAM 사용자의 비밀 액세스 키는 나열할 수 없습니다. 비밀 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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAccessKeys](#).

## Get-IAMAccessKeyLastUsed

다음 코드 예시에서는 Get-IAMAccessKeyLastUsed을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 제공된 액세스 키의 소유 사용자 이름과 마지막 사용 정보를 반환합니다.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAccessKeyLastUsed](#).

## Get-IAMAccountAlias

다음 코드 예시에서는 Get-IAMAccountAlias을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 명령은 AWS 계정에 대한 계정 별칭을 반환합니다.

```
Get-IAMAccountAlias
```

출력:

```
ExampleCo
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAccountAliases](#).

## Get-IAMAccountAuthorizationDetail

다음 코드 예시에서는 Get-IAMAccountAuthorizationDetail을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 AWS 계정의 ID에 대한 권한 부여 세부 정보를 가져오고 사용자, 그룹 및 역할을 포함하여 반환된 개체의 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 참조를 참조하십시오 [GetAccountAuthorizationDetails](#).

## Get-IAMAccountPasswordPolicy

다음 코드 예시에서는 Get-IAMAccountPasswordPolicy을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAccountPasswordPolicy](#).

## Get-IAMAccountSummary

다음 코드 예시에서는 Get-IAMAccountSummary을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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

...

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAccountSummary](#).

## Get-IAMAttachedGroupPolicyList

다음 코드 예시에서는 Get-IAMAttachedGroupPolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 계정에 이름이 지정된 IAM 그룹에 연결된 관리형 정책의 이름과 ARN을 AWS 반환합니다. **Admins** 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 참조를 참조하십시오 [ListAttachedGroupPolicies](#).

## Get-IAMAttachedRolePolicyList

다음 코드 예시에서는 Get-IAMAttachedRolePolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAttachedRolePolicies](#).

## Get-IAMAttachedUserPolicyList

다음 코드 예시에서는 Get-IAMAttachedUserPolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 계정에 이름이 지정된 IAM 사용자의 관리형 정책 이름과 ARN을 AWS 반환합니다. **Bob** 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 참조를 참조하십시오 [ListAttachedUserPolicies](#).

## Get-IAMContextKeysForCustomPolicy

다음 코드 예시에서는 Get-IAMContextKeysForCustomPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키를 가져옵니다. 여러 정책을 제공하려면 심볼로 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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetContextKeysForCustomPolicy](#).

## Get-IAMContextKeysForPrincipalPolicy

다음 코드 예시에서는 Get-IAMContextKeysForPrincipalPolicy를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키와 IAM 엔티티(사용자/역할 등)에 연결된 정책을 가져옵니다. 대상 - PolicyInputList 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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetContextKeysForPrincipalPolicy](#).

## Get-IAMCredentialReport

다음 코드 예시에서는 Get-IAMCredentialReport를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 사용자에게 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,password_last_changed,password_last_used,password_last_changed,password_last_used,root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04-15T16:31:25+00:00,A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+00:00,A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/A,A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/A,A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetCredentialReport](#).

## Get-IAMEntitiesForPolicy

다음 코드 예시에서는 `Get-IAMEntitiesForPolicy`을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListEntitiesForPolicy](#).

## Get-IAMGroup

다음 코드 예시에서는 `Get-IAMGroup`을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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   : 40SVDDJJTF4XEEXAMPLE2
UserName : Theresa

Arn      : arn:aws:iam::123456789012:user/David
CreateDate : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path     : /
UserId   : Y4FKWQCXTA52QEXAMPLE3
UserName : David

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetGroup](#).

**Get-IAMGroupForUser**

다음 코드 예시에서는 Get-IAMGroupForUser을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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    : ZU2E0WMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListGroupForUser](#).

## Get-IAMGroupList

다음 코드 예시에서는 Get-IAMGroupList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

- 예 1: 이 예제는 AWS 계정현재에 정의된 모든 IAM 그룹의 컬렉션을 반환합니다.

```
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    : ZU2E0WMK6WBZOEXAMPLE2
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 참조를 참조하십시오 [ListGroups](#).

## Get-IAMGroupPolicy

다음 코드 예시에서는 Get-IAMGroupPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}

```

```
}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetGroupPolicy](#).

## Get-IAMGroupPolicyList

다음 코드 예시에서는 Get-IAMGroupPolicyList를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListGroupPolicies](#).

## Get-IAMInstanceProfile

다음 코드 예시에서는 Get-IAMInstanceProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 현재 AWS 계정에 정의된 이름이 지정된 **ec2instancerole** 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 참조를 참조하십시오 [GetInstanceProfile](#).

## Get-IAMInstanceProfileForRole

다음 코드 예시에서는 Get-IAMInstanceProfileForRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListInstanceProfilesForRole](#).

## Get-IAMInstanceProfileList

다음 코드 예시에서는 Get-IAMInstanceProfileList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 현재에 정의된 인스턴스 프로파일 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 참조를 참조하십시오 [ListInstanceProfiles](#).

**Get-IAMLoginProfile**

다음 코드 예시에서는 Get-IAMLoginProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetLoginProfile](#).

**Get-IAMMFADevice**

다음 코드 예시에서는 Get-IAMMFADevice을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListMfaDevices](#).

## Get-IAMOpenIDConnectProvider

다음 코드 예시에서는 Get-IAMOpenIDConnectProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 Url	CreateDate	ThumbprintList
-----	-----	-----
---		
{MyOIDCApp}	2/3/2015 3:00:30 PM	
{12345abcdefghijk67890lmnopqrst98765uvwxyz}		oidc.example.com

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetOpenIdConnectProvider](#).

## Get-IAMOpenIDConnectProviderList

다음 코드 예시에서는 Get-IAMOpenIDConnectProviderList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListOpenIdConnectProviders](#).

## Get-IAMPolicy

다음 코드 예시에서는 Get-IAMPolicy을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPolicy](#).

## Get-IAMPolicyList

다음 코드 예시에서는 Get-IAMPolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 현재 AWS 계정에서 사용할 수 있는 처음 세 개의 관리형 정책 컬렉션을 반환합니다. **-scope**이 지정되지 않았으므로 기본적으로 관리형 **all** 정책과 고객 AWS 관리형 정책이 모두 포함됩니다.

```
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    : Z27SI6FQMG2EXAMPLE1
PolicyName  : AWSDirectConnectReadOnlyAccess
UpdateDate  : 2/6/2015 10:40:08 AM

Arn          : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount : 0
CreateDate   : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description  :
IsAttachable : True
Path        : /
PolicyId    : NJKMU274MET4EEXAMPLE2
PolicyName  : AmazonGlacierReadOnlyAccess
UpdateDate  : 2/6/2015 10:40:27 AM

Arn          : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount : 0
CreateDate   : 2/11/2015 9:21:45 AM
DefaultVersionId : v1

```

```

Description      :
IsAttachable    : True
Path            : /
PolicyId        : 5ULJS02FYVPYGEXAMPLE3
PolicyName      : AWSMarketplaceFullAccess
UpdateDate     : 2/11/2015 9:21:45 AM

```

예 2: 이 예에서는 현재 AWS 계정에서 사용할 수 있는 처음 두 고객 관리형 정책의 컬렉션을 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        : SQVCBLC4VA0UCEXAMPLE4
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        : X5JPBLJH2Z2S0EXAMPLE5
PolicyName      : policyforec2instancerole
UpdateDate     : 2/18/2015 8:52:31 AM

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListPolicies](#).

## Get-IAMPolicyVersion

다음 코드 예시에서는 Get-IAMPolicyVersion을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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": "*",
    "Resource": "*"
  }
}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPolicyVersion](#).

## Get-IAMPolicyVersionList

다음 코드 예시에서는 Get-IAMPolicyVersionList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 VersionId	Document	IsDefaultVersion
----- -----	-----	-----
2/12/2015 9:39:53 AM v2		True
2/12/2015 9:39:09 AM v1		False

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListPolicyVersions](#).

## Get-IAMRole

다음 코드 예시에서는 **Get-IAMRole**을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제는 **lambda\_exec\_role**의 세부 정보를 반환합니다. 여기에는 이 역할을 수입할 수 있는 사용자를 지정하는 신뢰 정책 문서가 포함됩니다. 정책 문서는 URL로 인코딩되며 **.NET UriDecode** 메서드를 사용하여 디코딩할 수 있습니다. 이 예제에서는 원래 정책에 업로드되기 전에 모든 공백이 제거됩니다. 역할을 수입한 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"}]}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetRole](#).

## Get-IAMRoleList

다음 코드 예시에서는 Get-IAMRoleList를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 AWS 계정의 모든 IAM 역할 목록을 검색합니다.

```
Get-IAMRoleList
```

예제 2: 이 예제 코드 스니펫은 AWS 계정의 IAM 역할 목록을 검색하여 한 번에 세 개씩 표시하고 각 그룹 사이에서 Enter 키를 누를 때까지 기다립니다. 다음 그룹이 시작되어야 하는 위치를 지정하기 위해 이전 직접 호출의 **Marker** 값을 전달합니다.

```

$nextMarker = $null
Do

```

```
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [ListRoles](#) AWS Tools for PowerShell

## Get-IAMRolePolicy

다음 코드 예시에서는 Get-IAMRolePolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 IAM 역할 **lambda\_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
<pre>           UserName -----           ----- %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%... oneClick_lambda_exec_role_policy    lambda_exec_role</pre>	

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

출력:

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:*"
    ],
    "Resource": "arn:aws:logs:*:*:*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject"
    ],
    "Resource": [
      "arn:aws:s3:*:*:*"
    ]
  }
]
}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetRolePolicy](#).

## Get-IAMRolePolicyList

다음 코드 예시에서는 Get-IAMRolePolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListRolePolicies](#).

## Get-IAMRoleTagList

다음 코드 예시에서는 Get-IAMRoleTagList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 역할과 연결된 태그를 가져옵니다.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListRoleTags](#).

## Get-IAMSAMLProvider

다음 코드 예시에서는 Get-IAMSAMLProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 ARM이 arn:aws:iam::123456789012:saml-provider/SAMLADFS인 SAML 2.0 제공업체에 대한 세부 정보를 검색합니다. 응답에는 AWS SAML 제공자 엔티티를 생성하기 위해 ID 제공자로부터 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 참조를 참조하십시오 [GetSamlProvider](#).

## Get-IAMSAMLProviderList

다음 코드 예시에서는 Get-IAMSAMLProviderList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 참조의 [ListSAMLProviders](#)를 참조하세요.

## Get-IAMServerCertificate

다음 코드 예시에서는 Get-IAMServerCertificate을 사용하는 방법을 보여 줍니다.

를 위한 도구 PowerShell

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-----

MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC

VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6

b24xFDASBgNVBAsTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0eW11YWMxHzAd

```

```

BgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAdDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xZDASBgNVBAsTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0e21sYWx1e21sYWx1e21sYWx1e21sYWx1e21sYWx1e21s
YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/
f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/
MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
CertificateChain      :
ServerCertificateMetadata :
Amazon.IdentityManagement.Model.ServerCertificateMetadata

```

```
$result.ServerCertificateMetadata
```

#### 출력:

```

Arn          : arn:aws:iam::123456789012:server-certificate/0rg1/0rg2/
MyServerCertificate
Expiration    : 1/14/2018 9:52:36 AM
Path         : /0rg1/0rg2/
ServerCertificateId : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate   : 4/21/2015 11:14:16 AM

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetServerCertificate](#).

## Get-IAMServerCertificateList

다음 코드 예시에서는 Get-IAMServerCertificateList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListServerCertificates](#).

## Get-IAMServiceLastAccessedDetail

다음 코드 예시에서는 Get-IAMServiceLastAccessedDetail을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetServiceLastAccessedDetails](#).

## Get-IAMServiceLastAccessedDetailWithEntity

다음 코드 예시에서는 Get-IAMServiceLastAccessedDetailWithEntity을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetServiceLastAccessedDetailsWithEntities](#).

## Get-IAMSigningCertificate

다음 코드 예시에서는 Get-IAMSigningCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **Bob**이라는 사용자와 연결된 서명 인증서에 대한 세부 정보를 검색합니다.

```
Get-IAMSigningCertificate -UserName Bob
```

출력:

```

CertificateBody : -----BEGIN CERTIFICATE-----
MIICiTCCAfICCD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAdDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAwTC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAd
BgkqhkiG9w0BCQEWEG5vb251QGFTYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI1MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAdDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYW11ZAdBgkqhkiG9w0BCQEWEG5vb251QGFT
YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
CertificateId : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status : Active
UploadDate : 4/20/2015 1:26:01 PM
UserName : Bob

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListSigningCertificates](#).

## Get-IAMUser

다음 코드 예시에서는 Get-IAMUser을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetUser](#).

## Get-IAMUserList

다음 코드 예시에서는 Get-IAMUserList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 현재 AWS 계정사용자 컬렉션을 검색합니다.

```
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 참조를 참조하십시오 [ListUsers](#).

## Get-IAMUserPolicy

다음 코드 예시에서는 Get-IAMUserPolicy을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
-----
-----
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...  Davids_IAM_Admin_Policy
  David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetUserPolicy](#).

## Get-IAMUserPolicyList

다음 코드 예시에서는 Get-IAMUserPolicyList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **David**라는 IAM 사용자에게 포함된 인라인 정책의 이름 목록을 검색합니다.

```
Get-IAMUserPolicyList -UserName David
```

출력:

```
 Davids_IAM_Admin_Policy
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListUserPolicies](#).

## Get-IAMUserTagList

다음 코드 예시에서는 Get-IAMUserTagList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 사용자와 연결된 태그를 가져옵니다.

```
Get-IAMUserTagList -UserName joe
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListUserTags](#).

## Get-IAMVirtualMFADevice

다음 코드 예시에서는 Get-IAMVirtualMFADevice을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 계정의 사용자에게 할당된 가상 MFA 디바이스 컬렉션을 검색합니다. AWS 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 참조를 참조하십시오 [ListVirtualMfaDevices](#).

## New-IAMAccessKey

다음 코드 예시에서는 New-IAMAccessKey을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 새 액세스 키와 비밀 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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateAccessKey](#).

## New-IAMAccountAlias

다음 코드 예시에서는 New-IAMAccountAlias을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예시에서는 계정의 계정 별칭을 로 변경합니다. AWS **mycompanyaws** 사용자 로그인 페이지의 주소가 <https://mycompanyaws.signin.aws.amazon.com/console>로 변경됩니다. 별칭 대신 계정 ID 번호를 사용하는 원래 URL(<accountidnumber><https://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 참조를 참조하십시오 [CreateAccountAlias](#).

## New-IAMGroup

다음 코드 예시에서는 `New-IAMGroup`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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         : /
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateGroup](#).

## New-IAMInstanceProfile

다음 코드 예시에서는 `New-IAMInstanceProfile`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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              : {}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateInstanceProfile](#).

## New-IAMLoginProfile

다음 코드 예시에서는 New-IAMLoginProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 Bob이라는 IAM 사용자에게 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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateLoginProfile](#).

## New-IAMOpenIDConnectProvider

다음 코드 예시에서는 New-IAMOpenIDConnectProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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.html> 단계를 따르십시오.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

출력:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateOpenIdConnectProvider](#).

## New-IAMPolicy

다음 코드 예시에서는 `New-IAMPolicy`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 현재 AWS 계정에 새 IAM 정책을 생성합니다. 파일은 정책 콘텐츠를 **MySamplePolicy.json** 제공합니다. **MySamplePolicy** 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 참조를 참조하십시오 [CreatePolicy](#).

## New-IAMPolicyVersion

다음 코드 예시에서는 `New-IAMPolicyVersion`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 VersionId	Document	IsDefaultVersion
----- -----	-----	-----
4/15/2015 10:54:54 AM v2		True

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreatePolicyVersion](#).

## New-IAMRole

다음 코드 예시에서는 `New-IAMRole`을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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%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%22AWS%22%3A%20%22arn%3Aaws%3Aiam%3A%3A123456789012%3ADavid%22%0D%0A%20%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"
    }
  ]
}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateRole](#).

**New-IAMSAMLProvider**

다음 코드 예시에서는 New-IAMSAMLProvider를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 참조의 [CreateSAMLProvider](#)를 참조하세요.

## New-IAMServiceLinkedRole

다음 코드 예시에서는 New-IAMServiceLinkedRole을 사용하는 방법을 보여 줍니다.

를 위한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateServiceLinkedRole](#).

## New-IAMUser

다음 코드 예시에서는 New-IAMUser을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **Bob**이라는 IAM 사용자를 생성합니다. Bob이 AWS 콘솔에 로그인해야 하는 경우 명령을 별도로 **New-IAMLoginProfile** 실행하여 비밀번호가 포함된 로그인 프로필을 만들어야 합니다. Bob이 플랫폼 간 CLI 명령을 AWS PowerShell 실행하거나 AWS API를 호출해야 하는 경우 별도로 명령을 **New-IAMAccessKey** 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 참조를 참조하십시오 [CreateUser](#).

**New-IAMVirtualMFADevice**

다음 코드 예시에서는 New-IAMVirtualMFADevice을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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. 가상 MFA 소프트웨어 프로그램에서 이 이미지를 스캔하여 계정을 생성할 수 있습니다 (StringSeed Base32 값을 수동으로 입력하는 대신 사용). 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CreateVirtualMfaDevice](#). AWS Tools for PowerShell

## Publish-IAMServerCertificate

다음 코드 예시에서는 Publish-IAMServerCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UploadServerCertificate](#).

## Publish-IAMSigningCertificate

다음 코드 예시에서는 Publish-IAMSigningCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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-----
MIICiTCCAFICCCQD6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMAKGA1UEBhMC
VVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBA5TC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAd
BgkqhkiG9w0BCQEWEG5vb251QGFTYXpvbi5jb20wHhcNMTEwNDI0MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWx1eHAdBgkqhkiG9w0BCQEWEG5vb251QGFT
YXpvbi5jb20wZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySwTC2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvcQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUHVxYUntneD9+h8Mg9q6q+auNKyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDmFJ10ZxBHjJnyp3780D8uTs7fLvJx79LjStB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----

CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHJMJLU
Status         : Active
UploadDate     : 4/20/2015 1:26:01 PM
```

```
UserName      : Bob
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UploadSigningCertificate](#).

## Register-IAMGroupPolicy

다음 코드 예시에서는 Register-IAMGroupPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

예 2: 이 예제는 이름이 지정된 AWS **AdministratorAccess** 관리형 정책을 IAM 그룹에 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 참조를 참조하십시오 [AttachGroupPolicy](#).

## Register-IAMRolePolicy

다음 코드 예시에서는 Register-IAMRolePolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 이름이 지정된 AWS **SecurityAudit** 관리형 정책을 IAM 역할에 연결합니다. **CoSecurityAuditors** 해당 역할을 수입하는 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 참조를 참조하십시오 [AttachRolePolicy](#).

## Register-IAMUserPolicy

다음 코드 예시에서는 Register-IAMUserPolicy를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 이름이 지정된 AWS **AmazonCognitoPowerUser** 관리형 정책을 IAM 사용자에게 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 참조를 참조하십시오 [AttachUserPolicy](#).

## Remove-IAMAccessKey

다음 코드 예시에서는 Remove-IAMAccessKey를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 이름이 지정된 **Bob** 사용자의 키 ID가 있는 AWS 액세스 키 쌍을 **AKIAIOSFODNN7EXAMPLE** 삭제합니다.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteAccessKey](#).

## Remove-IAMAccountAlias

다음 코드 예시에서는 Remove-IAMAccountAlias를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 사용자 AWS 계정계정에서 계정 별칭을 제거합니다. <https://mycompanyaws.signin.aws.amazon.com/console>에서 별칭을 사용하는 사용자 로그인 페이지가 더 이상 작동하지 않습니다. 대신 AWS 계정 <https://signin.aws.amazon.com/console>에서 ID 번호와 함께 원래 URL을 사용해야 합니다. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API에 [DeleteAccountAlias AWS Tools for PowerShell](#) 대한 세부 정보는 Cmdlet 참조를 참조하십시오.

## Remove-IAMAccountPasswordPolicy

다음 코드 예시에서는 Remove-IAMAccountPasswordPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 의 암호 정책을 AWS 계정 삭제하고 모든 값을 원래 기본값으로 재설정합니다. 암호 정책이 현재 존재하지 않는 경우 다음과 같은 오류 메시지가 나타납니다. 이름이 있는 계정 정책을 찾을 PasswordPolicy 수 없습니다.

```
Remove-IAMAccountPasswordPolicy
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteAccountPasswordPolicy](#).

## Remove-IAMClientIDFromOpenIDConnectProvider

다음 코드 예시에서는 Remove-IAMClientIDFromOpenIDConnectProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RemoveClientIDFromOpenIDConnectProvider](#).

## Remove-IAMGroup

다음 코드 예시에서는 Remove-IAMGroup을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 메시지 없이 작동합니다.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName MyTestGroup -Force  
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteGroup](#).

## Remove-IAMGroupPolicy

다음 코드 예시에서는 Remove-IAMGroupPolicy을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteGroupPolicy](#).

## Remove-IAMInstanceProfile

다음 코드 예시에서는 Remove-IAMInstanceProfile을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteInstanceProfile](#).

## Remove-IAMLoginProfile

다음 코드 예시에서는 Remove-IAMLoginProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **Bob**이라는 IAM 사용자로부터 로그인 프로파일을 삭제합니다. 이렇게 하면 사용자가 콘솔에 로그인할 수 없습니다. AWS 사용자가 사용자 계정에 연결되어 있을 수 있는 AWS 액세스 키를 사용하여 AWS CLI 또는 API 호출을 실행하는 것을 막지는 않습니다. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteLoginProfile](#).

## Remove-IAMOpenIDConnectProvider

다음 코드 예시에서는 Remove-IAMOpenIDConnectProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteOpenIdConnectProvider](#).

## Remove-IAMPolicy

다음 코드 예시에서는 Remove-IAMPolicy을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 엔티티에서 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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePolicy](#).

## Remove-IAMPolicyVersion

다음 코드 예시에서는 **Remove-IAMPolicyVersion**을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePolicyVersion](#).

## Remove-IAMRole

다음 코드 예시에서는 Remove-IAMRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteRole](#).

## Remove-IAMRoleFromInstanceProfile

다음 코드 예시에서는 Remove-IAMRoleFromInstanceProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RemoveRoleFromInstanceProfile](#).

## Remove-IAMRolePermissionsBoundary

다음 코드 예시에서는 Remove-IAMRolePermissionsBoundary을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 IAM 역할에 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteRolePermissionsBoundary](#).

## Remove-IAMRolePolicy

다음 코드 예시에서는 Remove-IAMRolePolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 IAM 역할 **S3BackupRole**에 포함된 인라인 정책 **S3AccessPolicy**를 삭제합니다.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteRolePolicy](#).

## Remove-IAMRoleTag

다음 코드 예시에서는 Remove-IAMRoleTag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 태그 키가 "abac"인 MyRoleName "역할에서 태그를 제거합니다. 여러 태그를 제거하려면 쉼표로 구분된 태그 키 목록을 제공합니다.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UntagRole](#).

## Remove-IAMSAMLProvider

다음 코드 예시에서는 Remove-IAMSAMLProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 참조의 [DeleteSAMLProvider](#)를 참조하세요.

## Remove-IAMServerCertificate

다음 코드 예시에서는 Remove-IAMServerCertificate을 사용하는 방법을 보여 줍니다.

를 위한 도구 PowerShell

예제 1: 이 예제는 **MyServerCert**라는 서버 인증서를 삭제합니다.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteServerCertificate](#).

## Remove-IAMServiceLinkedRole

다음 코드 예시에서는 Remove-IAMServiceLinkedRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 서비스 연결 역할을 삭제했습니다. 서비스에서 여전히 이 역할을 사용하고 있는 경우 이 명령을 실행하면 실패합니다.

```
Remove-IAMServiceLinkedRole -RoleName  
AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteServiceLinkedRole](#).

## Remove-IAMSigningCertificate

다음 코드 예시에서는 Remove-IAMSigningCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **Bob**이라는 IAM 사용자로부터 ID가 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**인 서명 인증서를 삭제합니다.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId  
Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteSigningCertificate](#).

## Remove-IAMUser

다음 코드 예시에서는 Remove-IAMUser을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteUser](#).

## Remove-IAMUserFromGroup

다음 코드 예시에서는 Remove-IAMUserFromGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RemoveUserFromGroup](#).

## Remove-IAMUserPermissionsBoundary

다음 코드 예시에서는 Remove-IAMUserPermissionsBoundary을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 IAM 사용자에게 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteUserPermissionsBoundary](#).

## Remove-IAMUserPolicy

다음 코드 예시에서는 Remove-IAMUserPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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.

```
$inlinenpols = Get-IAMUserPolicies -UserName Theresa  
foreach ($pol in $inlinenpols) { Remove-IAMUserPolicy -PolicyName $pol -UserName  
Theresa -Force}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteUserPolicy](#).

## Remove-IAMUserTag

다음 코드 예시에서는 Remove-IAMUserTag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UntagUser](#).

## Remove-IAMVirtualMFADevice

다음 코드 예시에서는 Remove-IAMVirtualMFADevice을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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 사용자에게 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 }
}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteVirtualMfaDevice](#).

## Request-IAMCredentialReport

다음 코드 예시에서는 Request-IAMCredentialReport을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GenerateCredentialReport](#).

## Request-IAMServiceLastAccessedDetail

다음 코드 예시에서는 Request-IAMServiceLastAccessedDetail을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 API의 cmdlet과 GenerateServiceLastAccessedDetails 동일합니다. 여기에는 Get-IAM 및 ServiceLastAccessedDetail Get-IAM에서 사용할 수 있는 작업 ID가 제공됩니다. ServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.  
[GenerateServiceLastAccessedDetails](#) AWS Tools for PowerShell

## Set-IAMDefaultPolicyVersion

다음 코드 예시에서는 Set-IAMDefaultPolicyVersion을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오  
[SetDefaultPolicyVersion](#).

## Set-IAMRolePermissionsBoundary

다음 코드 예시에서는 Set-IAMRolePermissionsBoundary을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 참조를 참조하십시오  
[PutRolePermissionsBoundary](#).

## Set-IAMUserPermissionsBoundary

다음 코드 예시에서는 Set-IAMUserPermissionsBoundary을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 참조를 참조하십시오  
[PutUserPermissionsBoundary](#).

## Sync-IAMMFADevice

다음 코드 예시에서는 Sync-IAMMFADevice을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ResyncMfaDevice](#).

## Unregister-IAMGroupPolicy

다음 코드 예시에서는 Unregister-IAMGroupPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachGroupPolicy](#).

## Unregister-IAMRolePolicy

다음 코드 예시에서는 Unregister-IAMRolePolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachRolePolicy](#).

## Unregister-IAMUserPolicy

다음 코드 예시에서는 Unregister-IAMUserPolicy을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DetachUserPolicy](#).

## Update-IAMAccessKey

다음 코드 예시에서는 Update-IAMAccessKey을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **Bob**이라는 IAM 사용자에게 대한 액세스 키 **AKIAIOSFODNN7EXAMPLE**의 상태를 **Inactive**로 변경합니다.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAccessKey](#).

## Update-IAMAccountPasswordPolicy

다음 코드 예시에서는 Update-IAMAccountPasswordPolicy를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAccountPasswordPolicy](#).

## Update-IAMAssumeRolePolicy

다음 코드 예시에서는 Update-IAMAssumeRolePolicy를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAssumeRolePolicy](#).

## Update-IAMGroup

다음 코드 예시에서는 Update-IAMGroup를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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/
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateGroup](#).

## Update-IAMLoginProfile

다음 코드 예시에서는 Update-IAMLoginProfile을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateLoginProfile](#).

## Update-IAMOpenIDConnectProviderThumbprint

다음 코드 예시에서는 Update-IAMOpenIDConnectProviderThumbprint을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 7359755EXAMPLEeabc3060bce3EXAMPLEeec4542a3
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateOpenIdConnectProviderThumbprint](#).

## Update-IAMRole

다음 코드 예시에서는 Update-IAMRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateRole](#).

## Update-IAMRoleDescription

다음 코드 예시에서는 Update-IAMRoleDescription을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 계정의 IAM 역할 설명을 업데이트합니다.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateRoleDescription](#).

## Update-IAMSAMLProvider

다음 코드 예시에서는 Update-IAMSAMLProvider을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateSamlProvider](#).

## Update-IAMServerCertificate

다음 코드 예시에서는 Update-IAMServerCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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/
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateServerCertificate](#).

## Update-IAMSigningCertificate

다음 코드 예시에서는 Update-IAMSigningCertificate을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateSigningCertificate](#).

## Update-IAMUser

다음 코드 예시에서는 Update-IAMUser를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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/
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateUser](#).

## Write-IAMGroupPolicy

다음 코드 예시에서는 Write-IAMGroupPolicy를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 **AppTesterPolicy**라는 인라인 정책을 생성하고 이를 IAM 그룹 **AppTesters**에 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. 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)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutGroupPolicy](#).

## Write-IAMRolePolicy

다음 코드 예시에서는 Write-IAMRolePolicy를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutRolePolicy](#).

## Write-IAMUserPolicy

다음 코드 예시에서는 Write-IAMUserPolicy을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

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)
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutUserPolicy](#).

## 다음 툴을 사용한 Kinesis 예제 PowerShell

다음 코드 예제는 Kinesis와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Get-KINRecord

다음 코드 예시에서는 Get-KINRecord를 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제에서는 일련의 하나 이상의 레코드에서 데이터를 반환하고 추출하는 방법을 보여줍니다. Get-KinRecord에 제공된 반복자는 반환할 레코드의 시작 위치를 결정합니다. 이 예제에서는 \$records라는 변수로 캡처됩니다. 그러면 \$records 컬렉션을 인덱싱하여 각 개별 레코드에 액세스할 수 있습니다. 레코드의 데이터가 UTF-8 인코딩 텍스트라고 가정하면 마지막 명령은 개체의 데이터를 추출하여 MemoryStream 콘솔에 텍스트로 반환하는 방법을 보여줍니다.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNpP"
```

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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetRecords](#).

## Get-KINShardIterator

다음 코드 예시에서는 Get-KINShardIterator을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 지정된 샤드 및 시작 위치에 대한 샤드 이터레이터를 반환합니다. 샤드 식별자와 시퀀스 번호에 대한 세부 정보는 반환된 스트림 객체의 샤드 컬렉션을 참조하여 Get-KinStream cmdlet의 출력에서 얻을 수 있습니다. 반환된 이터레이터는 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [GetShardIterator](#) AWS Tools for PowerShell

## Get-KINStream

다음 코드 예시에서는 Get-KINStream을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 참조를 참조하십시오 [DescribeStream](#).

## New-KINStream

다음 코드 예시에서는 New-KINStream을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 새 스트림을 만듭니다. 기본적으로 이 cmdlet은 출력을 반환하지 않으므로 -PassThru 스위치가 추가되어 나중에 사용할 수 있도록 -StreamName 매개 변수에 제공된 값을 반환합니다.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateStream](#).

## Remove-KINStream

다음 코드 예시에서는 Remove-KINStream을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 지정된 스트림을 삭제합니다. 명령이 실행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 표시하지 않으려면 -Force 스위치를 사용합니다.

```
Remove-KINStream -StreamName "mystream"
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeleteStream](#).AWS Tools for PowerShell

## Write-KINRecord

다음 코드 예시에서는 Write-KINRecord을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: -Text 매개 변수에 제공된 문자열을 포함하는 레코드를 씁니다.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey "Key1"
```

예 2: 지정된 파일에 포함된 데이터를 포함하는 레코드를 씁니다. 파일은 바이트 시퀀스로 처리되므로 텍스트가 포함된 경우 이 cmdlet에서 사용하기 전에 필요한 인코딩으로 작성해야 합니다.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey "Key2"
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [PutRecord](#).AWS Tools for PowerShell

## 예에 대한 도구를 사용한 Lambda 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with Lambda를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 GitHub 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다.

주제

- [작업](#)

작업

### Add-LMResourceTag

다음 코드 예시에서는 Add-LMResourceTag를 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

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" }
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [TagResource](#).

## Get-LMAccountSetting

다음 코드 예시에서는 Get-LMAccountSetting을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAccountSettings](#).

## Get-LMAlias

다음 코드 예시에서는 Get-LMAlias을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

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]}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAlias](#).

## Get-LMFunctionConcurrency

다음 코드 예시에서는 Get-LMFunctionConcurrency을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제에서는 Lambda 함수에 대한 예약된 동시성을 가져옵니다.

```
Get-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -Select *
```

출력:

```
ReservedConcurrentExecutions
-----
100
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetFunctionConcurrency](#).

## Get-LMFunctionConfiguration

다음 코드 예시에서는 Get-LMFunctionConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제에서는 Lambda 함수의 버전별 구성을 반환합니다.

```
Get-LMFunctionConfiguration -FunctionName "MyLambdaFunction123" -Qualifier
"PowershellAlias"
```

**출력:**

```

CodeSha256           : uW0W0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize             : 1426
DeadLetterConfig     : Amazon.Lambda.Model.DeadLetterConfig
Description          : Verson 3 to test Aliases
Environment          : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn         : arn:aws:lambda:us-
east-1:123456789012:function:MyLambdaFunction123
                    : PowershellAlias
FunctionName        : MyLambdaFunction123
Handler              : lambda_function.launch_instance
KMSKeyArn           :
LastModified        : 2019-12-25T09:52:59.872+0000
LastUpdateStatus    : Successful
LastUpdateStatusReason :
LastUpdateStatusReasonCode :
Layers              : {}
MasterArn           :
MemorySize          : 128
RevisionId          : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                : arn:aws:iam::123456789012:role/service-role/lambda
Runtime             : python3.8
State               : Active
StateReason         :
StateReasonCode     :
Timeout             : 600
TracingConfig       : Amazon.Lambda.Model.TracingConfigResponse
Version             : 4
VpcConfig           : Amazon.Lambda.Model.VpcConfigDetail

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetFunctionConfiguration](#).

**Get-LMFunctionList**

다음 코드 예시에서는 Get-LMFunctionList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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 CodeSize	Runtime	Timeout
test 243	python2.7	3
MylambdaFunction123 659	python3.8	600
myfuncpython1 675	python3.8	303

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListFunctions](#).

## Get-LMPolicy

다음 코드 예시에서는 Get-LMPolicy을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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"]}]}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPolicy](#).

## Get-LMProvisionedConcurrencyConfig

다음 코드 예시에서는 Get-LMProvisionedConcurrencyConfig을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

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                              :
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetProvisionedConcurrencyConfig](#).

## Get-LMProvisionedConcurrencyConfigList

다음 코드 예시에서는 Get-LMProvisionedConcurrencyConfigList를 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예제 1: 이 예제에서는 Lambda 함수에 대해 프로비저닝된 동시성 구성 목록을 검색합니다.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MyLambdaFunction123"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListProvisionedConcurrencyConfigs](#).

## Get-LMResourceTag

다음 코드 예시에서는 Get-LMResourceTag를 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTags](#).

## Get-LMVersionsByFunction

다음 코드 예시에서는 Get-LMVersionsByFunction을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제에서는 Lambda 함수의 각 버전에 대한 버전별 구성 목록을 반환합니다.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

출력:

FunctionName RoleName	Runtime	MemorySize	Timeout	CodeSize	LastModified
----- -----	-----	-----	-----	-----	-----
MylambdaFunction123 2020-01-10T03:20:56.390+0000	python3.8	128	600	659	lambda
MylambdaFunction123 2019-12-25T09:19:02.238+0000	python3.8	128	5	1426	lambda
MylambdaFunction123 2019-12-25T09:39:36.779+0000	python3.8	128	5	1426	lambda
MylambdaFunction123 2019-12-25T09:52:59.872+0000	python3.8	128	600	1426	lambda

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListVersionsByFunction](#).

## New-LMAlias

다음 코드 예시에서는 New-LMAlias을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateAlias](#).

## Publish-LMFunction

다음 코드 예시에서는 Publish-LMFunction을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제는 MyFunction AWS Lambda로 명명된 새 C# (dotnetcore1.0 런타임) 함수를 생성하여 로컬 파일 시스템의 zip 파일에서 함수에 대한 컴파일된 바이너리를 제공합니다 (상대 경로 또는 절대 경로를 사용할 수 있음). C# Lambda 함수는 :Namespace라는 지정을 사용하여 함수의 핸들러를 지정합니다. AssemblyName ClassName:: MethodName 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 mybucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
    -FunctionName MyFunction `
    -BucketName mybucket `
    -Key MyFunctionBinaries.zip `
    -Handler "AssemblyName::Namespace.ClassName::MethodName" `
    -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
    -Runtime dotnetcore1.0 `
    -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }

```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateFunction](#).

## Publish-LMVersion

다음 코드 예시에서는 Publish-LMVersion을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PublishVersion](#).

## Remove-LMAlias

다음 코드 예시에서는 Remove-LMAlias을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제에서는 명령에 언급된 Lambda 함수 별칭을 삭제합니다.

```
Remove-LMAlias -FunctionName "MyLambdaFunction123" -Name "NewAlias"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteAlias](#).

## Remove-LMFunction

다음 코드 예시에서는 Remove-LMFunction을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제에서는 특정 버전의 Lambda 함수를 삭제합니다.

```
Remove-LMFunction -FunctionName "MyLambdaFunction123" -Qualifier '3'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteFunction](#).

## Remove-LMFunctionConcurrency

다음 코드 예시에서는 Remove-LMFunctionConcurrency을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제에서는 Lambda 함수의 함수 동시성을 제거합니다.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteFunctionConcurrency](#).

## Remove-LMPermission

다음 코드 예시에서는 Remove-LMPermission을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제는 지정된 Lambda StatementId 함수에 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 참조를 참조하십시오 [RemovePermission](#).

## Remove-LMProvisionedConcurrencyConfig

다음 코드 예시에서는 Remove-LMProvisionedConcurrencyConfig을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제에서는 특정 별칭에 대한 프로비저닝된 동시성 구성을 제거합니다.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier  
"NewAlias1"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteProvisionedConcurrencyConfig](#).

## Remove-LMResourceTag

다음 코드 예시에서는 Remove-LMResourceTag을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예제 1: 함수에서 제공된 태그를 제거합니다. -Force 스위치가 지정되지 않은 경우 cmdlet은 진행하기 전에 확인 메시지를 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 메시지를 표시합니다. 제공된 태그별로 서비스에 대한 호출은 한 번만 이루어집니다.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UntagResource](#).

## Update-LMAlias

다음 코드 예시에서는 Update-LMAlias을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

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"}
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAlias](#).

## Update-LMFunctionCode

다음 코드 예시에서는 Update-LMFunctionCode을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이름이 MyFunction "인 함수를 지정된 zip 파일에 포함된 새 콘텐츠로 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 mybucket -Key UpdatedCode.zip
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateFunctionCode](#).

## Update-LMFunctionConfiguration

다음 코드 예시에서는 Update-LMFunctionConfiguration을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateFunctionConfiguration](#).

## Write-LMFunctionConcurrency

다음 코드 예시에서는 Write-LMFunctionConcurrency을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예제 1: 이 예제에서는 함수 전체에 대해 동시성 설정을 적용합니다.

```
Write-LMFunctionConcurrency -FunctionName "MyLambdaFunction123" -
ReservedConcurrentExecution 100
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutFunctionConcurrency](#).

## Write-LMProvisionedConcurrencyConfig

다음 코드 예시에서는 Write-LMProvisionedConcurrencyConfig을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제에서는 함수의 별칭에 프로비저닝된 동시성 구성을 추가합니다.

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MyLambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutProvisionedConcurrencyConfig](#).

## 다음 도구를 사용한 Amazon ML 예제 PowerShell

다음 코드 예제는 Amazon ML과 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Get-MLBatchPrediction

다음 코드 예시에서는 Get-MLBatchPrediction을 사용하는 방법을 보여 줍니다.

예 1: id ID가 있는 배치 예측의 세부 메타데이터를 반환합니다.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBatchPrediction](#).

### Get-MLBatchPredictionList

다음 코드 예시에서는 Get-MLBatchPredictionList을 사용하는 방법을 보여 줍니다.

예 1: 요청에 제공된 검색 기준과 일치하는 모든 데이터 레코드 BatchPredictions 및 관련 데이터 레코드의 목록을 반환합니다.

```
Get-MLBatchPredictionList
```

예 2: COMPLETED BatchPredictions 상태인 모든 목록을 반환합니다.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeBatchPredictions](#).

### Get-MLDataSource

다음 코드 예시에서는 Get-MLDataSource을 사용하는 방법을 보여 줍니다.

예 1: id ID를 DataSource 가진 a의 메타데이터, 상태 및 데이터 파일 정보를 반환합니다.

```
Get-MLDataSource -DataSourceId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDataSource](#).

## Get-MLDataSourceList

다음 코드 예시에서는 Get-MLDataSourceList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 전체 DataSources 및 관련 데이터 레코드 목록을 반환합니다.

```
Get-MLDataSourceList
```

- 예 2: COMPLETED 상태인 모든 DataSources 목록을 반환합니다.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeDataSources](#).

## Get-MLEvaluation

다음 코드 예시에서는 Get-MLEvaluation을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: id ID를 사용한 평가의 메타데이터 및 상태를 반환합니다.

```
Get-MLEvaluation -EvaluationId ID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetEvaluation](#).

## Get-MLEvaluationList

다음 코드 예시에서는 Get-MLEvaluationList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 모든 평가 리소스의 목록을 반환합니다.

```
Get-MLEvaluationList
```

예 2: 완료 상태인 모든 평가 목록을 반환합니다.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DescribeEvaluations](#).AWS Tools for PowerShell

## Get-MLModel

다음 코드 예시에서는 Get-MLModel을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: id ID가 있는 MLModel의 세부 메타데이터, 상태, 스키마 및 데이터 파일 정보를 반환합니다.

```
Get-MLModel -ModelId ID
```

- API에 대한 세부 정보는 Cmdlet 참조의 [AWS Tools for PowerShell GetMLModel](#)을 참조하십시오.

## Get-MLModelList

다음 코드 예시에서는 Get-MLModelList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 모든 모델 및 관련 데이터 레코드 목록을 반환합니다.

```
Get-MLModelList
```

예 2: COMPLETED 상태인 모든 모델의 목록을 반환합니다.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- API에 대한 세부 정보는 Cmdlet 참조의 [AWS Tools for PowerShell DescribeMLModels](#)를 참조하십시오.

## Get-MLPrediction

다음 코드 예시에서는 Get-MLPrediction을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예제 1: id ID가 있는 Model의 실시간 예측 엔드포인트 URL에 레코드를 전송합니다.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @"{"A" = "B"; "C" = "D"};"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [예측](#)을 참조하십시오.

## New-MLBatchPrediction

다음 코드 예시에서는 New-MLBatchPrediction을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: id ID가 있는 모델에 대한 새 배치 예측 요청을 생성하고 출력을 지정된 S3 위치에 배치합니다.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateBatchPrediction](#).

## New-MLDataSourceFromS3

다음 코드 예시에서는 New-MLDataSourceFromS3을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이름이 NAME이고 스키마가 SCHEMA인 S3 위치에 대한 데이터가 포함된 데이터 소스를 생성합니다.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [CreateDataSourceFromS3](#)를 참조하십시오.

## New-MLEvaluation

다음 코드 예시에서는 New-MLEvaluation을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예제 1: 주어진 데이터 소스 ID 및 모델 ID에 대한 평가 생성

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateEvaluation](#).

## New-MLModel

다음 코드 예시에서는 New-MLModel을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예 1: 학습 데이터를 사용하여 새 모델 만들기.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- API에 대한 세부 정보는 Cmdlet 참조의 [MLModel](#) 만들기를 참조하십시오. AWS Tools for PowerShell

## New-MLRealtimeEndpoint

다음 코드 예시에서는 New-MLRealtimeEndpoint을 사용하는 방법을 보여 줍니다.

## 예에 대한 도구 PowerShell

예 1: 지정된 모델 ID에 대한 새 실시간 예측 엔드포인트를 생성합니다.

```
New-MLRealtimeEndpoint -ModelId ID
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateRealtimeEndpoint](#).

## 다음과 같은 도구를 사용한 Macie 예제 PowerShell

다음 코드 예제는 Macie와 AWS Tools for PowerShell 함께 를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-MAC2FindingList

다음 코드 예시에서는 Get-MAC2FindingList을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: “CREDIT\_CARD\_NUMBER” 또는 “US\_SOCIAL\_SECURITY\_NUMBER” 유형의 민감한 데이터 탐지가 포함된 결과 목록을 반환합니다. FindingIds

```
$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 [ListFindings](#) 정보는 Cmdlet 참조를 참조하십시오. AWS Tools for PowerShell

## AWS OpsWorks 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 `with`를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS OpsWorks. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### New-OPSDeployment

다음 코드 예시에서는 `New-OPSDeployment`을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 명령은 스택의 레이어에 있는 모든 Linux 기반 인스턴스에 새 앱 배포를 생성합니다. AWS OpsWorks 레이어 ID를 지정하더라도 스택 ID도 지정해야 합니다. 필요한 경우 이 명령을 사용하면 배포에서 인스턴스를 다시 시작할 수 있습니다.

```
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z" -LayerId
"511b99c5-ec78-4caa-8a9d-1440116ffd1b" -AppId "0f7a109c-bf68-4336-8cb9-
d37fe0b8c61d" -Command_Name deploy -Command_Arg @{Name="allow_reboot";Value="true"}
```

예 2: 이 명령은 쿡북의 **appsetup** 레시피와 **phpapp** 쿡북의 **secbaseline** 레시피를 배포합니다 **testcookbook**. 배포 대상은 인스턴스 1개이지만 스택 ID와 레이어 ID도 필요합니다. `Command_Arg` 파라미터 **allow\_reboot** 속성이 로 설정되어 있어 필요한 **true** 경우 배포에서 인스턴스를 다시 시작할 수 있습니다.

```
$commandArgs = '{ "Name":"execute_recipes", "Args":{"recipes":
["phpapp::appsetup","testcookbook::secbaseline"]} } }'
New-OPSDeployment -StackID "724z93zz-zz78-4zzz-8z9z-1290123zzz1z"
-LayerId "511b99c5-ec78-4caa-8a9d-1440116ffd1b" -InstanceId
"d89a6118-0007-4ccf-a51e-59f844127021" -Command_Name $commandArgs -Command_Arg
@{Name="allow_reboot";Value="true"
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오 [CreateDeployment](#). AWS Tools for PowerShell

## AWS 가격표 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS 가격표. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-PLSAttributeValue

다음 코드 예시에서는 Get-PLSAttributeValue을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: us-east-1 리전에 있는 Amazon EC2의 '볼륨 유형' 속성 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 Cmdlet 참조를 참조하십시오. [GetAttributeValues](#) AWS Tools for PowerShell

**Get-PLSProduct**

다음 코드 예시에서는 Get-PLSProduct을 사용하는 방법을 보여 줍니다.

**예 1: Amazon EC2의 모든 제품에 대한 세부 정보를 반환합니다.**

예 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"},...
```

예 2: SSD 지원 '범용' 볼륨 유형을 기준으로 필터링된 us-east-1 지역의 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 AWS Tools for PowerShell 세부 정보는 Cmdlet 참조를 참조하십시오. [GetProducts](#)

## Get-PLSService

다음 코드 예시에서는 Get-PLSService을 사용하는 방법을 보여 줍니다.

예 1: us-east-1 지역에 대한 도구 PowerShell

예 1: us-east-1 지역에서 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
...	

예 2: us-east-1 지역의 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 Cmdlet 참조를 참조하십시오 [DescribeServices](#).AWS Tools for PowerShell

## 에 대한 도구를 사용한 Resource Groups 예제 PowerShell

다음 코드 예제는 with Resource Groups를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Add-RGResourceTag

다음 코드 예시에서는 Add-RGResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제에서는 값이 '워크박스'인 태그 키 '인스턴스'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 참조의 [태그](#)를 참조하십시오.

## Find-RGResource

다음 코드 예시에서는 Find-RGResource을 사용하는 방법을 보여 줍니다.

에 대한 도구: PowerShell

예 1: 이 예제에서는 태그 필터를 사용하여 ResourceQuery for Instance 리소스 유형을 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 참조를 참조하십시오 [SearchResources](#).

## Get-RGGroup

다음 코드 예시에서는 Get-RGGroup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [GetGroup](#).

## Get-RGGroupList

다음 코드 예시에서는 Get-RGGroupList을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [ListGroups](#).

## Get-RGGroupQuery

다음 코드 예시에서는 Get-RGGroupQuery을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [GetGroupQuery](#).

## Get-RGGroupResourceList

다음 코드 예시에서는 Get-RGGroupResourceList을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 리소스 유형별로 필터링된 기준에 따라 그룹 리소스를 나열합니다.

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
-GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

출력:

```
ResourceArn
-----
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 참조를 참조하십시오 [ListGroupResources](#).

## Get-RGResourceTag

다음 코드 예시에서는 Get-RGResourceTag을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 지정된 리소스 그룹 arn의 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 참조를 참조하십시오 [GetTags](#).

## New-RGGroup

다음 코드 예시에서는 New-RGGroup을 사용하는 방법을 보여 줍니다.

예 1에 대한 도구 PowerShell

예 1: 이 예제에서는 라는 이름의 TestPowerShellGroup 태그 기반 AWS Resource Groups 자원 그룹을 새로 만듭니다. 이 그룹에는 태그 키 "Name"과 태그 값 "test2"로 태그가 지정된 현재 지역의 Amazon EC2 인스턴스가 포함됩니다. 이 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

```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateGroup](#).

## Remove-RGGroup

다음 코드 예시에서는 Remove-RGGroup을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예에서는 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 참조를 참조하십시오 [DeleteGroup](#).

## Remove-RGResourceTag

다음 코드 예시에서는 Remove-RGResourceTag을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예에서는 리소스 그룹에서 언급된 태그를 제거합니다.

예 1: 이 예에서는 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 참조의 [태그 해제를](#) 참조하십시오.

## Update-RGGroup

다음 코드 예시에서는 Update-RGGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 그룹 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 참조를 참조하십시오 [UpdateGroup](#).

## Update-RGGroupQuery

다음 코드 예시에서는 Update-RGGroupQuery을 사용하는 방법을 보여 줍니다.

### 예 대한 도구 PowerShell

예 1: 이 예제는 쿼리 개체를 만들고 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 참조를 참조하십시오 [UpdateGroupQuery](#).

## 도구를 사용한 리소스 그룹 태깅 API 예제 PowerShell

다음 코드 예제는 with Resource Groups Tagging API를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

## Add-RGTResourceTag

다음 코드 예시에서는 Add-RGTResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예제 1: 이 예제는 값이 “베타”와 “preprod\_test”인 태그 키 “스테이지”와 “버전”을 Amazon S3 버킷과 Amazon DynamoDB 테이블에 추가합니다. 서비스를 한 번만 호출하면 태그를 적용할 수 있습니다.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
"version"="preprod_test" }
```

예제 2: 이 예제는 Amazon S3 버킷과 Amazon DynamoDB 테이블에 지정된 태그와 값을 추가합니다. cmdlet으로 파이프된 각 리소스 ARN에 대해 하나씩, 서비스에 대해 두 번의 호출이 이루어집니다.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [TagResources](#).AWS Tools for PowerShell

## Get-RGTResource

다음 코드 예시에서는 Get-RGTResource을 사용하는 방법을 보여 줍니다.

## 예 1: 한 지역에서 태그가 지정된 모든 리소스와 해당 리소스에 연결된 태그 키를 반환합니다.

예 1: 한 지역에서 태그가 지정된 모든 리소스와 해당 리소스에 연결된 태그 키를 반환합니다. cmdlet에 -Region 매개 변수를 제공하지 않으면 셸 또는 EC2 인스턴스 메타데이터에서 지역 유추를 시도합니다.

```
Get-RGTResource
```

출력:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version,
othertag}	

예 2: 리전에서 지정된 유형의 태그가 지정된 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:::mybucket	{stage, version,
othertag}	

예 3: 리전에서 지정된 유형의 태그가 지정된 모든 리소스를 반환합니다. 참고로 리소스 유형을 cmdlet으로 파이프하면 제공된 각 리소스 유형에 대해 서비스가 한 번 호출됩니다.

```
"dynamodb","s3" | Get-RGTResource
```

출력:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}
arn:aws:s3:::mybucket	{stage, version,
othertag}	

예 4: 지정된 필터와 일치하는 태그가 지정된 리소스를 모두 반환합니다.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

출력:

ResourceARN	Tags
-----	----
arn:aws:s3:::mybucket othertag}	{stage, version,

예 5: 지정된 필터 및 리소스 유형과 일치하는 태그가 지정된 리소스를 모두 반환합니다.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

출력:

ResourceARN	Tags
-----	----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable	{stage, version}

예 6: 지정된 필터와 일치하는 태그가 지정된 리소스를 모두 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 참조를 참조하십시오 [GetResources](#).

## Get-RGTTagKey

다음 코드 예시에서는 Get-RGTTagKey을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 지정된 지역의 모든 태그 키를 반환합니다. -Region 매개 변수가 지정되지 않은 경우 cmdlet은 기본 셀 리전 또는 EC2 인스턴스 메타데이터에서 리전을 유추하려고 시도합니다. 단, 태그 키는 특정 순서로 반환되지 않습니다.

```
Get-RGTagKey -region us-west-2
```

출력:

```
version  
stage
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetTagKeys](#).

## Get-RGTagValue

다음 코드 예시에서는 Get-RGTagValue을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 지역 내 지정된 태그의 값을 반환합니다. -Region 매개 변수가 지정되지 않은 경우 cmdlet은 기본 셀 리전 또는 EC2 인스턴스 메타데이터에서 리전을 유추하려고 시도합니다.

```
Get-RGTagValue -Key "stage" -Region us-west-2
```

출력:

```
beta
```

- API에 대한 세부 정보는 Cmdlet 참조를 참조하십시오. [GetTagValues](#) AWS Tools for PowerShell

## Remove-RGResourceTag

다음 코드 예시에서는 Remove-RGResourceTag을 사용하는 방법을 보여 줍니다.

## 예 1: Amazon S3 버킷과 Amazon DynamoDB 테이블에서 태그 키 “스테이지”, “버전”과 관련 값을 제거합니다

예 1: Amazon S3 버킷과 Amazon DynamoDB 테이블에서 태그 키 “스테이지”, “버전”과 관련 값을 제거합니다. 태그를 제거하려면 서비스를 한 번만 직접적으로 호출하면 됩니다. 태그가 제거되기 전에 cmdlet에 확인 메시지가 표시됩니다. 확인을 생략하려면 -Force 매개 변수를 추가하십시오.

```
$arn1 = "arn:aws:s3:::mybucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

예 2: Amazon S3 버킷과 Amazon DynamoDB 테이블에서 태그 키 “스테이지”, “버전”과 관련 값을 제거합니다. cmdlet으로 파이프된 각 리소스 ARN에 대해 하나씩, 서비스에 대해 두 번의 호출이 이루어집니다. 각 호출 전에 cmdlet에 확인 메시지가 표시됩니다. 확인을 생략하려면 -Force 매개 변수를 추가하십시오.

```
$arn1 = "arn:aws:s3:::mybucket"
$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 참조를 참조하십시오 [UntagResources](#).

## 다음과 같은 도구를 사용한 Route 53 예제 PowerShell

다음 코드 예제는 Route 53과 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

### 주제

- [작업](#)

## 작업

### Edit-R53ResourceRecordSet

다음 코드 예시에서는 Edit-R53ResourceRecordSet을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예제 1: 이 예제에서는 www.example.com에 대한 A 레코드를 생성하고 test.example.com의 A 레코드를 192.0.2.3에서 192.0.2.1로 변경합니다. 단, TXT 유형 레코드의 변경 값은 큰따옴표로 묶어야 합니다. 자세한 내용은 Amazon Route 53 설명서를 참조하십시오. Get-R53Changecmdlet을 사용하여 폴링하여 변경 완료 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
```

예 2: 이 예에서는 별칭 리소스 레코드 세트를 만드는 방법을 보여줍니다. 'Z222222222'는 별칭 리소스 레코드 세트를 생성하고 있는 Amazon Route 53 호스팅 영역의 ID입니다. 'example.com'은 별칭을 생성하려는 영역 정점이고, 'www.example.com'은 별칭을 생성하려는 하위 도메인입니다. 'Z1111111111111'은 로드 밸런서의 호스팅 영역 ID의 예이며, example-load-balancer-1111111111.us-east-1.elb.amazonaws.com'은 Amazon Route 53이 example.com 및 www.example.com에 대한 쿼리에 응답하는 로드 밸런서 도메인 이름의 예입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하십시오. Get-R53Changecmdlet을 사용하여 폴링하여 변경 완료 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
```

예 3: 이 예에서는 `www.example.com`에 대해 두 개의 A 레코드를 생성합니다. 4분의 1 ( $1/(1+3)$ ), Amazon Route 53은 첫 번째 리소스 레코드 세트 (192.0.2.9 및 192.0.2.10) 의 두 값을 사용하여 `www.example.com`에 대한 쿼리에 응답합니다. 4분의 3 ( $3/(1+3)$ ) Amazon Route 53은 두 번째 리소스 레코드 세트 (192.0.2.11 및 192.0.2.12) 에 대한 두 개의 값을 사용하여 `www.example.com`에 대한 쿼리에 응답합니다. 자세한 내용은 Amazon Route 53 설명서를 참조하십시오. `Get-R53Changecmdlet`을 사용하여 폴링하여 변경 완료 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
```

예 4: 이 예에서는 `example.com`이 가중치 기반 별칭 리소스 레코드 세트를 만들려는 도메인이라고 가정하고 가중치 기반 별칭 리소스 레코드 세트를 만드는 방법을 보여줍니다. `SetIdentifier` 두

가중치 기반 별칭 리소스 레코드 세트를 서로 구분합니다. Name 및 Type 요소의 값이 두 리소스 레코드 세트 모두에 동일하기 때문에 이 요소가 필요합니다. DNSName 값으로 지정된 ELB 로드 밸런서의 호스팅 영역 ID의 예는 Z1111111111111111 및 Z3333333333333333입니다. example-load-balancer-222222222.us-east-1.elb.amazonaws.com 및 example-load-balancer-444444444.us-east-1.elb.amazonaws.com은 Amazon Route 53이 example.com에 대한 쿼리에 응답하는 Elastic Load Balancing 도메인의 예입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하십시오. Get-R53Changecmdlet을 사용하여 폴링하여 변경 완료 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 = "Z1111111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z555555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
record sets. Amazon Route 53 responds to queries for example.com with the first ELB
domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}
```

```
Edit-R53ResourceRecordSet @params
```

예 5: 이 예제에서는 지연 시간 별칭 리소스 레코드 세트 두 개를 생성합니다. 하나는 미국 서부 (오레곤) 지역 (us-west-2) 의 ELB 로드 밸런서용이고 다른 하나는 아시아 태평양 (싱가포르) 지역의 로드 밸런서용 (ap-southeast-1) 입니다. 자세한 내용은 Amazon Route 53 설명서를 참조하십시오. Get-R53Changecmdlet을 사용하여 폴링하여 변경 완료 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 = "Z11111111111111"
$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 = "Z22222222222222"
$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 참조를 참조하십시오 [ChangeResourceRecordSets](#).

## Get-R53AccountLimit

다음 코드 예시에서는 Get-R53AccountLimit을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 참조를 참조하십시오 [GetAccountLimit](#).

## Get-R53CheckerIpRanges

다음 코드 예시에서는 Get-R53CheckerIpRanges을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 Route53 상태 확인 프로그램의 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 Cmdlet 참조를 참조하십시오. [GetCheckerIpRanges](#) AWS Tools for PowerShell

## Get-R53HostedZone

다음 코드 예시에서는 Get-R53HostedZone을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: ID가 Z1D633PJN98FT9 인 호스팅 영역의 세부 정보를 반환합니다.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetHostedZone](#).

## Get-R53HostedZoneCount

다음 코드 예시에서는 Get-R53HostedZoneCount을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 현재의 퍼블릭 및 프라이빗 호스팅 영역의 총 수를 반환합니다 AWS 계정.

```
Get-R53HostedZoneCount
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetHostedZoneCount](#).

## Get-R53HostedZoneLimit

다음 코드 예시에서는 Get-R53HostedZoneLimit을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 이 예제는 지정된 호스팅 영역에서 생성할 수 있는 최대 레코드 수에 대한 제한을 반환합니다.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAAF -Type MAX_RRSETS_BY_ZONE
```

출력:

```
5
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetHostedZoneLimit](#).

## Get-R53HostedZoneList

다음 코드 예시에서는 Get-R53HostedZoneList을 사용하는 방법을 보여 줍니다.

## 에 대한 도구 PowerShell

예 1: 모든 퍼블릭 및 프라이빗 호스팅 영역을 출력합니다.

```
Get-R53HostedZoneList
```

예 2: ID가 NZ8X2CISAMPLE 인 재사용 가능한 위임 세트와 연결된 모든 호스팅 영역을 출력합니다.

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListHostedZones](#).

## Get-R53HostedZonesByName

다음 코드 예시에서는 Get-R53HostedZonesByName을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 모든 퍼블릭 및 프라이빗 호스팅 영역을 도메인 이름을 기준으로 ASCII 순서로 반환합니다.

```
Get-R53HostedZonesByName
```

예 2: 퍼블릭 및 프라이빗 호스팅 영역을 지정된 DNS 이름에서 시작하여 도메인 이름을 기준으로 ASCII 순서로 반환합니다.

```
Get-R53HostedZonesByName -DnsName example2.com
```

예 3: 이 예제에서는 각 호출 후 스택의 서비스 응답에 첨부된 마커 속성을 사용하여 먼저 단일 항목을 검색한 다음 모든 영역이 반환될 때까지 한 번에 두 항목씩 반복하여 호스팅 영역을 수동으로 열거하는 방법을 보여줍니다. **\$AWSHistory**

```
Get-R53HostedZonesByName -MaxItem 1
while ($LastServiceResponse.IsTruncated)
{
    $nextPageParams = @{
        DnsName=$LastServiceResponse.NextDNSName
        HostedZoneId=$LastServiceResponse.NextHostedZoneId
    }
    Get-R53HostedZonesByName -MaxItem 2 @nextPageParams
}
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [ListHostedZonesByName](#) AWS Tools for PowerShell

## Get-R53QueryLoggingConfigList

다음 코드 예시에서는 Get-R53QueryLoggingConfigList을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 현재와 관련된 DNS 쿼리 로깅의 모든 구성을 반환합니다 AWS 계정.

예 1: 이 예제는 현재와 관련된 DNS 쿼리 로깅의 모든 구성을 반환합니다 AWS 계정.

```
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-f0ae56aaa2b	Z3MEQ8T7AAA1BF	arn:aws:logs:us-east-1:111111111112:log-group:/aws/route53/example3.com:*

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListQueryLoggingConfigs](#).

## Get-R53ReusableDelegationSet

다음 코드 예시에서는 Get-R53ReusableDelegationSet을 사용하는 방법을 보여 줍니다.

## 예 1: 이 예제는 위임 세트에 할당된 네 개의 이름 서버를 포함하여 지정된 위임 세트에 대한 정보를 검색합니다.

예 1: 이 예제는 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 Cmdlet 참조를 참조하십시오 [GetReusableDelegationSet](#).AWS Tools for PowerShell

## New-R53HostedZone

다음 코드 예시에서는 New-R53HostedZone을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 재사용 가능한 위임 세트와 연결된 'example.com'이라는 새 호스팅 영역을 생성합니다. 단, 작업을 두 번 실행할 위험 없이 필요한 경우 요청을 재시도할 수 있도록 CallerReference 파라미터 값을 제공해야 합니다. 호스팅 영역은 VPC에서 생성되므로 자동으로 비공개가 되므로 -HostedZoneConfig\_PrivateZone 매개변수를 설정하지 않아야 합니다.

```
$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
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateHostedZone](#).

## New-R53QueryLoggingConfig

다음 코드 예시에서는 New-R53QueryLoggingConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예제는 지정된 호스팅 영역에 대한 새 Route53 DNS 쿼리 로깅 구성을 생성합니다. Amazon Route53은 DNS 쿼리 로그를 지정된 클라우드워치 로그 그룹에 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 Cmdlet 참조를 참조하십시오 [CreateQueryLoggingConfig](#).AWS Tools for PowerShell

## New-R53ReusableDelegationSet

다음 코드 예시에서는 New-R53ReusableDelegationSet을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 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 Cmdlet 참조를 참조하십시오 [CreateReusableDelegationSet](#).AWS Tools for PowerShell

## Register-R53VPCWithHostedZone

다음 코드 예시에서는 Register-R53VPCWithHostedZone을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 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 Cmdlet 참조의 [VPC 연결을 WithHostedZone](#) 참조하십시오. AWS Tools for PowerShell

## Remove-R53HostedZone

다음 코드 예시에서는 Remove-R53HostedZone을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 ID의 호스팅 영역을 삭제합니다. -Force 스위치 파라미터를 추가하지 않는 한 명령이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteHostedZone](#).

## Remove-R53QueryLoggingConfig

다음 코드 예시에서는 Remove-R53QueryLoggingConfig을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 DNS 쿼리 로깅을 위해 지정된 구성을 제거합니다.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteQueryLoggingConfig](#).

## Remove-R53ReusableDelegationSet

다음 코드 예시에서는 Remove-R53ReusableDelegationSet을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 예에서는 지정된 재사용 가능한 위임 세트를 삭제합니다.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DeleteReusableDelegationSet](#). AWS Tools for PowerShell

## Unregister-R53VPCFromHostedZone

다음 코드 예시에서는 Unregister-R53VPCFromHostedZone을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제는 지정된 VPC를 프라이빗 호스팅 영역에서 분리합니다.

```
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 Cmdlet 참조의 VPC [연결 해제를 FromHostedZone](#) 참조하십시오. AWS Tools for PowerShell

## Update-R53HostedZoneComment

다음 코드 예시에서는 Update-R53HostedZoneComment을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 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 참조를 참조하십시오 [UpdateHostedZoneComment](#).

## 다음 도구를 사용한 Amazon S3 예제 PowerShell

다음 코드 예제는 Amazon S3와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Copy-S3Object

다음 코드 예시에서는 Copy-S3Object를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예시 1: 이 명령은 “test-files” 버킷의 “sample.txt” 객체를 동일한 버킷으로 복사하지만 새로운 키인 “sample-copy.txt”를 사용합니다.

```
Copy-S3Object -BucketName test-files -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 test-files -Key sample.txt -DestinationKey sample-copy.txt
-DestinationBucket backup-files
```

예시 3: 이 명령은 “test-files” 버킷에서 "local-sample.txt"라는 이름의 로컬 파일로 "sample.txt" 객체를 다운로드합니다.

```
Copy-S3Object -BucketName test-files -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 test-files -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 test-files -KeyPrefix data -LocalFolder c:\downloads
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CopyObject](#).

## Get-S3ACL

다음 코드 예시에서는 Get-S3ACL을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 명령은 S3 객체의 객체 소유자 세부 정보를 가져옵니다.

```
Get-S3ACL -BucketName 's3casetestbucket' -key 'initialize.ps1' -Select
AccessControlList.Owner
```

출력:

```
DisplayName Id
----- --
```

```
testusername          9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [GetACL](#)을 참조하십시오.

## Get-S3Bucket

다음 코드 예시에서는 Get-S3Bucket을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 모든 S3 버킷을 반환합니다.

```
Get-S3Bucket
```

예시 2: 이 명령은 이름이 “test-files”인 버킷을 반환합니다.

```
Get-S3Bucket -BucketName test-files
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListBuckets](#).

## Get-S3BucketAccelerateConfiguration

다음 코드 예시에서는 Get-S3BucketAccelerateConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 버킷에 대해 전송 가속화 설정이 활성화된 경우 Enabled 값을 반환합니다.

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

출력:

```
Value
-----
Enabled
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketAccelerateConfiguration](#).

## Get-S3BucketAnalyticsConfiguration

다음 코드 예시에서는 Get-S3BucketAnalyticsConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터의 세부 정보를 반환합니다.

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketAnalyticsConfiguration](#).

## Get-S3BucketAnalyticsConfigurationList

다음 코드 예시에서는 Get-S3BucketAnalyticsConfigurationList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 분석 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListBucketAnalyticsConfigurations](#).

## Get-S3BucketEncryption

다음 코드 예시에서는 Get-S3BucketEncryption을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 버킷과 연결된 모든 서버 측 암호화 규칙을 반환합니다.

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketEncryption](#).

## Get-S3BucketInventoryConfiguration

다음 코드 예시에서는 Get-S3BucketInventoryConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷에 대해 이름이 'testinventory'인 인벤토리의 세부 정보를 반환합니다.

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testinventory'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketInventoryConfiguration](#).

## Get-S3BucketInventoryConfigurationList

다음 코드 예시에서는 Get-S3BucketInventoryConfigurationList을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 인벤토리 구성 중 처음 100개를 반환합니다.

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListBucketInventoryConfigurations](#).

## Get-S3BucketLocation

다음 코드 예시에서는 Get-S3BucketLocation을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 제약 조건이 있는 경우 's3testbucket' 버킷에 대한 위치 제약 조건을 반환합니다.

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

출력:

```
Value
-----
ap-south-1
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketLocation](#).

## Get-S3BucketLogging

다음 코드 예시에서는 Get-S3BucketLogging을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 버킷의 로깅 상태를 반환합니다.

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

출력:

```
TargetBucketName  Grants TargetPrefix
-----
testbucket1      {}      testprefix
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketLogging](#).

## Get-S3BucketMetricsConfiguration

다음 코드 예시에서는 Get-S3BucketMetricsConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 'testfilter'라는 지표 필터에 대한 세부 정보를 반환합니다.

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId 'testfilter'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketMetricsConfiguration](#).

## Get-S3BucketNotification

다음 코드 예시에서는 Get-S3BucketNotification을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 예시는 지정된 버킷의 알림 구성을 검색합니다.

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
  TopicConfigurations
```

출력:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketNotification](#).

## Get-S3BucketPolicy

다음 코드 예시에서는 Get-S3BucketPolicy을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 출력합니다.

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketPolicy](#).

## Get-S3BucketPolicyStatus

다음 코드 예시에서는 Get-S3BucketPolicyStatus을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 버킷이 퍼블릭 버킷인지 여부를 나타내는 지정된 S3 버킷에 대한 정책 상태를 반환합니다.

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketPolicyStatus](#).

## Get-S3BucketReplication

다음 코드 예시에서는 Get-S3BucketReplication을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이름이 'mybucket'인 버킷에 설정된 복제 구성 정보를 반환합니다.

```
Get-S3BucketReplication -BucketName mybucket
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketReplication](#).

## Get-S3BucketRequestPayment

다음 코드 예시에서는 Get-S3BucketRequestPayment을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이름이 'mybucket'인 버킷에 대한 지불 요청 구성을 반환합니다. 기본적으로 버킷에서 다운로드하는 비용은 버킷 소유자가 지불합니다.

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketRequestPayment](#).

## Get-S3BucketTagging

다음 코드 예시에서는 Get-S3BucketTagging을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 버킷과 관련된 모든 태그를 반환합니다.

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketTagging](#).

## Get-S3BucketVersioning

다음 코드 예시에서는 Get-S3BucketVersioning을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 버킷과 관련된 버전 관리 상태를 반환합니다.

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketVersioning](#).

## Get-S3BucketWebsite

다음 코드 예시에서는 Get-S3BucketWebsite을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 구성 세부 정보를 반환합니다.

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetBucketWebsite](#).

## Get-S3CORSConfiguration

다음 코드 예시에서는 Get-S3CORSConfiguration을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 지정된 S3 버킷에 해당하는 모든 CORS 구성 규칙을 포함하는 객체를 반환합니다.

```
Get-S3CORSConfiguration -BucketName 's3testbucket' -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 : {}

```

- API에 대한 세부 정보는 Cmdlet 참조의 [GetCorsConfiguration](#)을 참조하십시오.AWS Tools for PowerShell

**Get-S3LifecycleConfiguration**

다음 코드 예시에서는 Get-S3LifecycleConfiguration을 사용하는 방법을 보여 줍니다.

예에 대한 도구 PowerShell

예 1: 이 예제는 버킷의 수명 주기 구성을 검색합니다.

```
Get-S3LifecycleConfiguration -BucketName test-bla
```

**출력:**

```

Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetLifecycleConfiguration](#).

## Get-S3Object

다음 코드 예시에서는 Get-S3Object를 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예시 1: 이 명령은 “test-files” 버킷에 있는 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName test-files
```

예시 2: 이 명령은 "test-files" 버킷에서 "sample.txt" 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName test-files -Key sample.txt
```

예시 3: 이 명령은 “test-files” 버킷에서 접두사가 “sample”인 모든 항목에 대한 정보를 검색합니다.

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListObjects](#).

## Get-S3ObjectLockConfiguration

다음 코드 예시에서는 Get-S3ObjectLockConfiguration을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷에 대해 객체 잠금 구성이 활성화된 경우 'Enabled' 값을 반환합니다.

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

출력:

```
Value
-----
Enabled
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetObjectLockConfiguration](#).

## Get-S3ObjectMetadata

다음 코드 예시에서는 Get-S3ObjectMetadata을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 지정된 S3 버킷에서 키가 ListTrusts '.txt'인 객체의 메타데이터를 반환합니다.

```
Get-S3ObjectMetadata -BucketName 's3testbucket' -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 Cmdlet 참조를 참조하십시오 [GetObjectMetadata](#).AWS Tools for PowerShell

## Get-S3ObjectRetention

다음 코드 예시에서는 Get-S3ObjectRetention을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 객체 보존의 모드와 종료 날짜를 반환합니다.

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetObjectRetention](#).

## Get-S3ObjectTagSet

다음 코드 예시에서는 Get-S3ObjectTagSet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 샘플은 지정된 S3 버킷에 있는 객체와 연결된 태그를 반환합니다.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

출력:

```
Key Value
--- -----
test value
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetObjectTagging](#).

## Get-S3PreSignedURL

다음 코드 예시에서는 Get-S3PreSignedURL을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 지정된 키와 만료 날짜에 대해 미리 서명된 URL을 반환합니다.

```
Get-S3PreSignedURL -BucketName 's3testbucket' -Key 'testkey' -Expires '2023-11-16'
```

예 2: 이 명령은 지정된 키와 만료 날짜가 있는 디렉터리 버킷에 대해 미리 서명된 URL을 반환합니다.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
    Get-S3PreSignedURL -BucketName sampledirectorybucket--use1-az5--x-s3 -Key
    'testkey' -Expire '2023-11-17'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 [GetPreSignedURL](#)을 참조하십시오.

## Get-S3PublicAccessBlock

다음 코드 예시에서는 Get-S3PublicAccessBlock을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 퍼블릭 액세스 차단 구성을 반환합니다.

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPublicAccessBlock](#).

## Get-S3Version

다음 코드 예시에서는 Get-S3Version을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 지정된 S3 버킷의 모든 객체 버전에 대한 메타데이터를 반환합니다.

```
Get-S3Version -BucketName 's3testbucket'
```

출력:

```
IsTruncated           : False
KeyMarker             :
VersionIdMarker       :
NextKeyMarker         :
NextVersionIdMarker   :
Versions               : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
    ListTrusts.json}
```

```
Name           : s3testbucket
Prefix         :
MaxKeys        : 1000
CommonPrefixes : {}
Delimiter      :
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListVersions](#).

## New-S3Bucket

다음 코드 예시에서는 New-S3Bucket을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 명령은 “sample-bucket”이라는 새 프라이빗 버킷을 생성합니다.

```
New-S3Bucket -BucketName sample-bucket
```

예 2: 이 명령은 읽기-쓰기 권한이 있는 “sample-bucket”이라는 새 버킷을 생성합니다.

```
New-S3Bucket -BucketName sample-bucket -PublicReadWrite
```

예 3: 이 명령은 읽기 전용 권한이 있는 “sample-bucket”이라는 새 버킷을 생성합니다.

```
New-S3Bucket -BucketName sample-bucket -PublicReadOnly
```

예제 4: 이 명령은 를 사용하여 “samplebucket--use1-az5--x-s3”이라는 새 디렉터리 버킷을 생성합니다. PutBucketConfiguration

```
$bucketConfiguration = @{
    BucketInfo = @{
        DataRedundancy = 'SingleAvailabilityZone'
        Type = 'Directory'
    }
    Location = @{
        Name = 'use1-az5'
        Type = 'AvailabilityZone'
    }
}
New-S3Bucket -BucketName samplebucket--use1-az5--x-s3 -BucketConfiguration
$bucketConfiguration -Region us-east-1
```

- API에 [PutBucket](#) 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. AWS Tools for PowerShell

## Read-S3Object

다음 코드 예시에서는 Read-S3Object를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 "test-files" 버킷에서 "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 test-files -Key sample.txt -File local-sample.txt
```

예시 2: 이 명령은 "test-files" 버킷에서 가상 디렉터리 "DIR"을 검색하여 현재 위치의 "Local-DIR"이라는 폴더에 저장합니다. 이 명령을 직접 호출하기 전에 "Local-DIR" 폴더가 없어도 됩니다.

```
Read-S3Object -BucketName test-files -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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetObject](#).

## Remove-S3Bucket

다음 코드 예시에서는 Remove-S3Bucket을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 'test-files' 버킷에서 모든 객체 및 객체 버전을 제거한 다음 버킷을 삭제합니다. 명령을 실행하면 계속 진행하기 전에 확인하라는 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 추가합니다. 비어 있지 않은 버킷은 삭제할 수 없다는 점에 유의하세요.

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucket](#).

## Remove-S3BucketAnalyticsConfiguration

다음 코드 예시에서는 Remove-S3BucketAnalyticsConfiguration을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testfilter'인 분석 필터를 제거합니다.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId  
'testfilter'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketAnalyticsConfiguration](#).

## Remove-S3BucketEncryption

다음 코드 예시에서는 Remove-S3BucketEncryption을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이렇게 하면 제공된 S3 버킷에 대해 활성화된 암호화가 비활성화됩니다.

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketEncryption](#).

## Remove-S3BucketInventoryConfiguration

다음 코드 예시에서는 Remove-S3BucketInventoryConfiguration을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 명령은 지정된 S3 버킷에 해당하는 이름이 testInventoryName "인 인벤토리를 제거합니다.

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId  
'testInventoryName'
```

출력:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "s3testbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketInventoryConfiguration](#).

## Remove-S3BucketMetricsConfiguration

다음 코드 예시에서는 Remove-S3BucketMetricsConfiguration을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷에서 이름이 'testmetrics'인 지표 필터를 제거합니다.

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId  
'testmetrics'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketMetricsConfiguration](#).

## Remove-S3BucketPolicy

다음 코드 예시에서는 Remove-S3BucketPolicy을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷과 연결된 버킷 정책을 제거합니다.

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketPolicy](#).

## Remove-S3BucketReplication

다음 코드 예시에서는 Remove-S3BucketReplication을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이름이 'mybucket'인 버킷과 연결된 복제 구성을 삭제합니다. 참고로 이 작업을 수행하려면 s3: DeleteReplicationConfiguration 작업에 대한 권한이 필요합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3BucketReplication -BucketName mybucket
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketReplication](#).

## Remove-S3BucketTagging

다음 코드 예시에서는 Remove-S3BucketTagging을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷과 연결된 모든 태그를 제거합니다.

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketTagging](#).

## Remove-S3BucketWebsite

다음 코드 예시에서는 Remove-S3BucketWebsite을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 정적 웹 사이트 호스팅 속성을 비활성화합니다.

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteBucketWebsite](#).

## Remove-S3CORSConfiguration

다음 코드 예시에서는 Remove-S3CORSConfiguration을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 지정된 S3 버킷의 CORS 구성을 제거합니다.

```
Remove-S3CORSConfiguration -BucketName 's3testbucket'
```

출력:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
target "s3testbucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API 세부 정보는 Cmdlet 참조의 [CORS구성 삭제](#)를 참조하십시오. AWS Tools for PowerShell

## Remove-S3LifecycleConfiguration

다음 코드 예시에서는 Remove-S3LifecycleConfiguration을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 명령은 해당 S3 버킷의 모든 수명 주기 규칙을 제거합니다.

```
Remove-S3LifecycleConfiguration -BucketName 's3testbucket'
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteLifecycleConfiguration](#).

## Remove-S3MultipartUpload

다음 코드 예시에서는 Remove-S3MultipartUpload을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 5일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

예시 2: 이 명령은 2014년 1월 2일 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
02, 2014"
```

예시 3: 이 명령은 2014년 1월 2일 10시 45분 37초 이전에 생성된 멀티파트 업로드를 중단합니다.

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02 10:45:37"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AbortMultipartUpload](#).

## Remove-S3Object

다음 코드 예시에서는 Remove-S3Object를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 "test-files" 버킷에서 "sample.txt" 객체를 제거합니다. 명령이 실행되기 전에 확인 메시지가 표시됩니다. 확인 메시지를 차단하려면 -Force 스위치를 사용하세요.

```
Remove-S3Object -BucketName test-files -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 test-files -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 처리할 수 없는 키의 오류만 가져오려면 -ReportErrorsOnly 매개변수를 추가합니다. 이 매개변수는 -Quiet라는 별칭으로 지정할 수도 있습니다.

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

예 4: 이 예제에서는 인라인 표현식과 -KeyCollection 매개 변수를 사용하여 삭제할 개체의 키를 가져옵니다. Get-S3ObjectAmazon.S3.Model.S3Object 인스턴스의 컬렉션을 반환합니다. 각 인스턴스에는 객체를 식별하는 문자열 유형의 키 멤버가 있습니다.

```
Remove-S3Object -bucketname "test-files" -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

(alias -S3ObjectCollection) 매개 변수로 전달하여 서비스를 한 번 호출하여 일괄 삭제를 수행할 수 있도록 하는 것이 좋습니다.

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-S3Object -Force
```

예 6: 이 예제에서는 삭제 마커를 나타내는 Amazon.S3.Model.S3 ObjectVersion 인스턴스 컬렉션을 cmdlet으로 파이프하여 삭제합니다. 들어오는 객체는 한 번에 하나씩 처리됩니다. 대규모 컬렉션의 경우 해당 컬렉션을 cmdlet의 -InputObject (alias -S3ObjectCollection) 매개 변수로 전달하여 서비스를 한 번 호출하여 일괄 삭제를 수행할 수 있도록 하는 것이 좋습니다.

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

예제 7: 이 스크립트는 - 매개 변수와 함께 사용할 개체 배열을 구성하여 개체 집합을 일괄 삭제 (이 경우 삭제 마커) 하는 방법을 보여줍니다. KeyAndVersionCollection

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteObjects](#).

## Remove-S3ObjectTagSet

다음 코드 예시에서는 Remove-S3ObjectTagSet을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷에서 'testfile.txt' 키가 있는 객체와 연결된 모든 태그를 제거합니다.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteObjectTagging](#).

## Remove-S3PublicAccessBlock

다음 코드 예시에서는 Remove-S3PublicAccessBlock을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 버킷의 퍼블릭 액세스 차단 설정을 끕니다.

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select '^BucketName'
```

출력:

```
s3testbucket
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePublicAccessBlock](#).

## Set-S3BucketEncryption

다음 코드 예시에서는 Set-S3BucketEncryption을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 's3testbucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketEncryption](#).

## Test-S3Bucket

다음 코드 예시에서는 Test-S3Bucket을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령은 버킷이 있으면 True를 반환하고, 그렇지 않으면 False를 반환합니다. 이 명령은 버킷이 사용자 소유가 아닌 경우에도 True를 반환합니다.

```
Test-S3Bucket -BucketName test-files
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [Test-S3Bucket](#).

## Write-S3BucketAccelerateConfiguration

다음 코드 예시에서는 Write-S3BucketAccelerateConfiguration을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 전송 가속화를 활성화합니다.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -  
AccelerateConfiguration_Status $statusVal
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketAccelerateConfiguration](#).

## Write-S3BucketNotification

다음 코드 예시에서는 Write-S3BucketNotification을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 이 예제는 S3 이벤트에 ObjectRemovedDelete 대한 SNS 주제 구성을 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 kt-tools -TopicConfiguration $topic
```

예제 2: 이 예제는 지정된 ObjectCreatedAll 버킷에 대해 Lambda 함수로 전송하는 알림을 활성화합니다.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
$lambdaConfig
```

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 ssm-editor -LambdaFunctionConfiguration
$firstLambdaConfig,$secondLambdaConfig

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketNotification](#).

## Write-S3BucketReplication

다음 코드 예시에서는 Write-S3BucketReplication을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제는 버킷 'examplebucket'에서 키 이름 접두사 TaxDocs ""로 생성된 모든 새 객체를 'exampletargetbucket' 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:::exampltargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

예 2: 이 예제에서는 키 이름 접두사 "" 또는 ""로 생성된 새 객체를 'exampltargetbucket' 버킷에 복제할 수 있도록 하는 여러 규칙을 사용하여 복제 구성을 설정합니다. TaxDocs OtherDocs 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:::exampltargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampltargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

예제 3: 이 예제는 지정된 버킷의 복제 구성을 업데이트하여 키 이름 접두사가 ""TaxDocs인 객체를 버킷 'exampltargetbucket'으로 복제하는 것을 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:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [PutBucketReplication](#) AWS Tools for PowerShell

## Write-S3BucketRequestPayment

다음 코드 예시에서는 Write-S3BucketRequestPayment을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이름이 'mybucket'인 버킷의 지불 요청 구성을 업데이트하여 버킷에서 다운로드를 요청하는 사람에게 다운로드 요금이 부과되도록 합니다. 기본적으로 다운로드 비용은 버킷 소유자가 지불합니다. 요청 결제를 기본값으로 다시 설정하려면 RequestPaymentConfiguration\_Payer BucketOwner 매개변수에 "을 사용하십시오.

```
Write-S3BucketRequestPayment -BucketName mybucket -RequestPaymentConfiguration_Payer
Requester
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketRequestPayment](#).

## Write-S3BucketTagging

다음 코드 예시에서는 Write-S3BucketTagging을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 태

그입니다. 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 관리 콘솔의 태그 편집기를 사용하여 S3 버킷 태그를 관리할 수 있습니다.

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -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 cloudtrail-test-2018 | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketTagging](#).

## Write-S3BucketVersioning

다음 코드 예시에서는 Write-S3BucketVersioning을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 지정된 S3 버킷의 버전 관리를 활성화합니다.

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status Enabled
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketVersioning](#).

## Write-S3BucketWebsite

다음 코드 예시에서는 Write-S3BucketWebsite을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예시 1: 이 명령은 인덱스 문서가 'index.html', 오류 문서가 'error.html'인 지정된 버킷에 대한 웹 사이트 호스팅을 활성화합니다.

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutBucketWebsite](#).

## Write-S3LifecycleConfiguration

다음 코드 예시에서는 Write-S3LifecycleConfiguration을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 NewRule \$에 제공된 구성을 쓰거나 대체합니다. 이 구성에서는 지정된 접두사와 태그 값으로 범위 개체를 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 my-review-scrap -Configuration_Rule
$NewRule
```

예 2: 이 예제에서는 필터링을 사용하여 여러 규칙을 설정합니다. \$는 30일 후에 보관할 객체를 Glacier에, 120일 후에 보관할 객체로 ArchiveRule 설정합니다. DeepArchive \$는 'py' 접두사와 태그:key 'archive'가 'yes'로 설정된 객체의 경우 현재 버전과 이전 버전 모두 150일 후에 ExpireRule 만료됩니다.

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = @{
        Days= 150
    }
    Id = "Remove-in-150-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
                [Amazon.S3.Model.LifecycleTagPredicate] @{
                    "Tag"= @{
                        "Key" = "archived"
                        "Value" = "yes"
                    }
                }
            )
        }
    }
    Status= 'Enabled'
    NoncurrentVersionExpiration = @{
        NoncurrentDays = 150
    }
}

$ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
    Expiration = $null
    Id = "Archive-to-Glacier-in-30-days"
    Filter= @{
        LifecycleFilterPredicate = [Amazon.S3.Model.LifecycleAndOperator]@{
            Operands= @(
                [Amazon.S3.Model.LifecyclePrefixPredicate] @{
                    "Prefix" = "py"
                },
            ),
        }
    }
}
```

```
[Amazon.S3.Model.LifecycleTagPredicate] @{
    "Tag"= @{
        "Key" = "reviewed"
        "Value" = "yes"
    }
}
)
}
}
Status = 'Enabled'
NoncurrentVersionExpiration = @{
    NoncurrentDays = 75
}
Transitions = @(
    @{
        Days = 30
        "StorageClass"= 'Glacier'
    },
    @{
        Days = 120
        "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
    }
)
}

Write-S3LifecycleConfiguration -BucketName my-review-scrap -Configuration_Rule
$ExpireRule,$ArchiveRule
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [PutLifecycleConfiguration](#) AWS Tools for PowerShell

## Write-S3Object

다음 코드 예시에서는 Write-S3Object을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 test-files -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 test-files -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 test-files -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

예 4: 이 명령은 하위 디렉토리 "Scripts"의 모든 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 ""를 각 객체에 적용합니다. SampleScripts 업로드된 각 파일의 키는 "SampleScripts/filename"이며, 여기서 '파일 이름'은 다릅니다.

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

예 5: 이 명령은 로컬 디렉터 "스크립트"에 있는 모든 \*.ps1 파일을 "test-files" 버킷에 업로드하고 공통 키 접두사 ""를 각 객체에 적용합니다. SampleScripts 업로드된 각 파일의 키는 "SampleScripts/filename.ps1"이며, 여기서 '파일 이름'은 다양합니다.

```
Write-S3Object -BucketName test-files -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 test-files -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 test-files -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 test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse -
TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [PutObject](#) AWS Tools for PowerShell

## Write-S3ObjectRetention

다음 코드 예시에서는 Write-S3ObjectRetention을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutObjectRetention](#).

## 다음과 같은 도구를 사용한 S3 Glacier 예제 PowerShell

다음 코드 예제는 S3 Glacier와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 GitHub 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다.

주제

- [작업](#)

## 작업

### Get-GLCJob

다음 코드 예시에서는 Get-GLCJob을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 지정된 작업의 세부 정보를 반환합니다. 작업이 성공적으로 완료되면 Read-gc JobOutput cmdlet을 사용하여 작업의 내용 (아카이브 또는 인벤토리 목록) 을 로컬 파일 시스템으로 검색할 수 있습니다.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

출력:

```
Action                : ArchiveRetrieval
ArchiveId              : o909j...X-TpIhQJw
ArchiveSHA256TreeHash : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes    : 38034480
Completed              : False
CompletionDate         : 1/1/0001 12:00:00 AM
CreationDate           : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes   : 0
JobDescription         :
JobId                  : op1x...JSbthM
JobOutputPath          :
OutputLocation         :
RetrievalByteRange    : 0-38034479
SelectParameters      :
SHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
SNSTopic               :
StatusCode             : InProgress
StatusMessage         :
Tier                   : Standard
VaultARN               : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [DescribeJob](#) AWS Tools for PowerShell

## New-GLCVault

다음 코드 예시에서는 New-GLCVault를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 사용자 계정을 위한 새 볼트를 만듭니다. - AccountId 매개 변수에 값이 제공되지 않았으므로 cmdlet은 현재 계정을 나타내는 기본값인 "-"를 사용합니다.

```
New-GLCVault -VaultName myvault
```

출력:

```
/01234567812/vaults/myvault
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateVault](#).

## Read-GLCJobOutput

다음 코드 예시에서는 Read-GLCJobOutput을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 지정된 작업에서 검색하도록 예약된 아카이브 콘텐츠를 다운로드하고 그 내용을 디스크의 파일에 저장합니다. 다운로드를 통해 체크섬을 확인할 수 있습니다 (사용 가능한 경우). 필요한 경우 다음과 같이 서비스 응답 기록에서 체크섬을 가져올 수 있습니다 (이 cmdlet이 마지막으로 실행된 것으로 가정). **\$AWSHistory.LastServiceResponse** cmdlet이 가장 최근에 실행되지 않은 경우 **\$AWSHistory.Commands** 컬렉션을 검사하여 관련 서비스 응답을 얻으십시오.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp\nblue.bin"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetJobOutput](#).

## Start-GLCJob

다음 코드 예시에서는 Start-GLCJob을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 사용자가 소유한 지정된 보관소에서 아카이브를 검색하는 작업을 시작합니다. Get-GLCJob cmdlet을 사용하여 작업 상태를 확인할 수 있습니다. 작업이 성공적으로 완료되면 Read-GC JobOutput cmdlet을 사용하여 아카이브의 내용을 로컬 파일 시스템으로 검색할 수 있습니다.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
"archive retrieval" -ArchiveId "o909j...TX-TpIhQJw"
```

출력:

JobId	JobOutputPath	Location
-----	-----	-----
op1x...JSbthM		/012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [InitiateJob](#) AWS Tools for PowerShell

## Write-GLCArchive

다음 코드 예시에서는 Write-GLCArchive을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 지정된 보관소에 단일 파일을 업로드하고 아카이브 ID와 계산된 체크섬을 반환합니다.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

출력:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b

예 2: 폴더 계층 구조의 콘텐츠를 사용자 계정의 지정된 보관소에 업로드합니다. 업로드된 각 파일에 대해 cmdlet은 파일 이름, 해당 아카이브 ID 및 아카이브의 계산된 체크섬을 내보냅니다.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

출력:

FilePath	ArchiveId	Checksum
-----	-----	-----
C:\temp\blue.bin	o909jUUs...TTX-TpIhQJw	79f3e...f4395b
C:\temp\green.bin	qXAf0dSG...czo729UHXrw	d50a1...9184b9
C:\temp\lum.bin	39aNifP3...q9nb8nZkFIg	28886...5c3e27
C:\temp\red.bin	vp7E6rU_...Ejk_HhjAxKA	e05f7...4e34f5
C:\temp\Folder1\file1.txt	_eRINlip...5Sxy7dD2BaA	d0d2a...c8a3ba
C:\temp\Folder2\file2.iso	-Ix3jlmU...iXiDh-Xf0PA	7469e...3e86f1

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [UploadArchive](#) AWS Tools for PowerShell

## 다음 도구를 사용한 Amazon SES 예제 PowerShell

다음 코드 예제는 Amazon SES와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

작업

### Get-SESIentity

다음 코드 예시에서는 Get-SESIentity을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 확인 상태와 상관없이 특정 AWS 계정의 모든 ID (이메일 주소 및 도메인) 가 포함된 목록을 반환합니다.

```
Get-SESIIdentity
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListIdentities](#).

## Get-SESSendQuota

다음 코드 예시에서는 Get-SESSendQuota을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령어는 사용자의 현재 전송 한도를 반환합니다.

```
Get-SESSendQuota
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetSendQuota](#).

## Get-SESSendStatistic

다음 코드 예시에서는 Get-SESSendStatistic을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 명령어는 사용자의 전송 통계를 반환합니다. 결과는 지난 2주간의 전송 활동을 나타내는 데이터 요소 목록입니다. 목록의 각 데이터 요소에는 15분 간격의 통계가 포함됩니다.

```
Get-SESSendStatistic
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetSendStatistics](#).

## 다음 도구를 사용한 Amazon SNS 예제 PowerShell

다음 코드 예제는 Amazon SNS와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Publish-SNSMessage

다음 코드 예시에서는 Publish-SNSMessage을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 MessageAttribute 선언된 단일 인라인으로 메시지를 게시하는 방법을 보여 줍니다.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
"Hello" -MessageAttribute
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
StringValue='AnyCity'}}
```

예 2: 이 예에서는 여러 개를 미리 MessageAttributes 선언하여 메시지를 게시하는 방법을 보여 줍니다.

```
$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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet에 [게시](#) 참조를 참조하십시오.

## 다음 도구를 사용한 Amazon SQS 예제 PowerShell

다음 코드 예제는 Amazon SQS와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### Add-SQSPermission

다음 코드 예시에서는 Add-SQSPermission을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예에서는 지정된 사용자가 지정된 대기열에서 메시지를 보낼 수 AWS 계정 있도록 합니다.

```
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 참조를 참조하십시오 [AddPermission](#).

## Clear-SQSQueue

다음 코드 예시에서는 Clear-SQSQueue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 참조를 참조하십시오 [PurgeQueue](#).

## Edit-SQSMessageVisibility

다음 코드 예시에서는 Edit-SQSMessageVisibility을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 대기열에 지정된 수신 핸들이 있는 메시지의 가시성 제한 시간을 10시간 (10시간\* 60분\* 60초 = 36000초) 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 참조를 참조하십시오 [ChangeMessageVisibility](#).

## Edit-SQSMessageVisibilityBatch

다음 코드 예시에서는 Edit-SQSMessageVisibilityBatch을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 대기열에 지정된 수신 핸들이 있는 메시지 2개에 대한 가시성 제한 시간을 변경합니다. 첫 번째 메시지의 가시성 제한 시간이 10시간 (10시간\* 60분\* 60초 = 36000초) 으로 변경됩니다. 두 번째 메시지의 가시성 제한 시간이 5시간 (5시간\* 60분\* 60초 = 18000초) 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-SQSMessagesVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
    $changeVisibilityRequest2

```

**출력:**

```

Failed      Successful
-----
{}          {Request2, Request1}

```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ChangeMessageVisibilityBatch](#).

**Get-SQSDeadLetterSourceQueue**

다음 코드 예시에서는 Get-SQSDeadLetterSourceQueue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 대기열을 데드레터 대기열로 사용하는 모든 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 Cmdlet 참조를 참조하십시오 [ListDeadLetterSourceQueues](#). AWS Tools for PowerShell

## Get-SQSQueue

다음 코드 예시에서는 Get-SQSQueue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 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
```

예 2: 이 예에서는 지정된 이름으로 시작하는 모든 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 참조를 참조하십시오 [ListQueues](#).

## Get-SQSQueueAttribute

다음 코드 예시에서는 Get-SQSQueueAttribute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                     495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                     398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":
  "SendMessageFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                     arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                     [ApproximateNumberOfMessagesNotVisible, 0],
                                     [ApproximateNumberOfMessagesDelayed, 0]...}
```

예 2: 이 예제에서는 지정된 대기열의 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":"2008-10-17","Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/
SQSDefaultPolicy","Statement":[{"Sid":"Sid14
                                495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}},
{"Sid":
  "SendMessageFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                  : {[MaximumMessageSize, 262144],
[VisibilityTimeout, 30]}

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetQueueAttributes](#).

## Get-SQSQueueUrl

다음 코드 예시에서는 Get-SQSQueueUrl을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 이름을 가진 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 참조를 참조하십시오 [GetQueueUrl](#).

## New-SQSQueue

다음 코드 예시에서는 New-SQSQueue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 이름의 대기열을 만듭니다.

```
New-SQSQueue -QueueName MyQueue
```

출력:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateQueue](#).

## Receive-SQSMessage

다음 코드 예시에서는 Receive-SQSMessage을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 대기열에 대해 수신될 다음 최대 10개의 메시지에 대한 정보를 나열합니다. 이 정보에는 지정된 메시지 속성 값이 있는 경우 해당 값이 포함됩니다.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
  StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

출력:

```
Attributes          : {[SenderId, AIDAIAZKMSNQ7EXAMPLE], [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 참조를 참조하십시오 [ReceiveMessage](#).

## Remove-SQSMessage

다음 코드 예시에서는 Remove-SQSMessage를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 참조를 참조하십시오 [DeleteMessage](#).

## Remove-SQSMessageBatch

다음 코드 예시에서는 Remove-SQSMessageBatch를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 = "AQEBq0VY...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 참조를 참조하십시오 [DeleteMessageBatch](#).

## Remove-SQSPermission

다음 코드 예시에서는 Remove-SQSPermission을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 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 참조를 참조하십시오 [RemovePermission](#).

## Remove-SQSQueue

다음 코드 예시에서는 Remove-SQSQueue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 지정된 대기열을 삭제합니다.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteQueue](#).

## Send-SQSMessage

다음 코드 예시에서는 Send-SQSMessage을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예에서는 10초 동안 메시지 전달이 지연된 상태로 지정된 속성과 메시지 본문이 포함된 메시지를 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 c739-4d0c-818b-1820eEXAMPLE	51b0a3256d59467f973009b73EXAMPLE	c35fed8f-

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SendMessage](#).

**Send-SQSMessageBatch**

다음 코드 예시에서는 Send-SQSMessageBatch를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 지정된 속성 및 메시지 본문이 있는 메시지 2개를 지정된 대기열로 보냅니다. 첫 번째 메시지는 15초, 두 번째 메시지는 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 참조를 참조하십시오 [SendMessageBatch](#).

## Set-SQSQueueAttribute

다음 코드 예시에서는 Set-SQSQueueAttribute을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에서는 대기열을 SNS 주제에 구독하는 정책을 설정하는 방법을 보여줍니다. 메시지가 주제에 게시되면 메시지가 구독된 대기열로 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
$policy = @"
{
  "Version": "2008-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
        }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

예 2: 이 예에서는 지정된 대기열의 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 참조를 참조하십시오 [SetQueueAttributes](#).

## AWS STS 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS STS. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Convert-STSAuthorizationMessage

다음 코드 예시에서는 Convert-STSAuthorizationMessage을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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..."
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DecodeAuthorizationMessage](#).

### Get-STSFederationToken

다음 코드 예시에서는 Get-STSFederationToken을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 사용자에게 대한 최종 권한은 전달된 정책과 IAM 사용자 정책의 교차점을 기준으로 가장 제한적인 세트입니다.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
3600
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetFederationToken](#).

## Get-STSSessionToken

다음 코드 예시에서는 Get-STSSessionToken을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 설정된 기간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 임시 자격 증명을 요청하는 데 사용되는 자격 증명은 현재 셸 기본값에서 유추됩니다. 다른 자격 증명을 지정하려면 - ProfileName 또는 - AccessKey /- SecretKey 매개변수를 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...  SamPleTokenN.....

```

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...  SamPleTokenN.....

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetSessionToken](#).

## Use-STSRole

다음 코드 예시에서는 Use-STSRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 요청한 사용자가 일반적으로 액세스할 수 없는 AWS 리소스에 액세스하는 데 1시간 동안 사용할 수 있는 임시 자격 증명 세트 (액세스 키, 비밀 키, 세션 토큰) 를 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 위임할 수 있는 각 역할에 대해 고객 계정은 해당 역할이 수임될 때마다 - ExternalId 매개변수로 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 참조를 참조하십시오 [AssumeRole](#).

## Use-STSWebIdentityRole

다음 코드 예시에서는 Use-STSWebIdentityRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: Login with Amazon ID 제공업체를 통해 인증된 사용자에게 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. WebIdentityToken -에 제공된 값은 ID 제공자가 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 참조를 참조하십시오 [AssumeRoleWithWebIdentity](#).

## AWS Support 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다  
AWS Support. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### Add-ASACommunicationToCase

다음 코드 예시에서는 Add-ASACommunicationToCase을 사용하는 방법을 보여 줍니다.

예 1: 이메일 통신 본문을 지정된 대/소문자에 추가합니다.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

예 2: 이메일 통신 본문을 지정된 대/소문자에 추가하고 이메일의 CC 라인에 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 참조를 참조하십시오](#)  
[AddCommunicationToCase](#).

## Get-ASACase

다음 코드 예시에서는 Get-ASACase을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 모든 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase
```

예 2: 지정된 날짜 및 시간 이후 모든 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

예 3: 해결된 지원 사례를 포함하여 처음 10개 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

예 4: 지정된 단일 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

예 5: 지정된 지원 사례의 세부 정보를 반환합니다.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",  
"case-18929034710-2011-c4fdeabf33c5cf47")
```

예 6: 수동 페이징을 사용하여 모든 지원 사례를 반환합니다. 케이스는 20개씩 배치로 검색됩니다.

```
$nextToken = $null  
do {  
    Get-ASACase -NextToken $nextToken -MaxResult 20  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeCases](#).

## Get-ASACommunication

다음 코드 예시에서는 Get-ASACommunication을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 지정된 케이스에 대한 모든 통신을 반환합니다.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

예 2: 지정된 케이스에 대해 2012년 1월 1일 자정 UTC 이후의 모든 통신을 반환합니다.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime  
"2012-01-10T00:00Z"
```

예 3: 수동 페이징을 사용하여 지정된 케이스에 대해 2012년 1월 1일 자정 UTC 이후 모든 통신을 반환합니다. 통신은 20개씩 일괄 검색됩니다.

```
$nextToken = $null  
do {  
    Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -NextToken  
    $nextToken -MaxResult 20  
    $nextToken = $AWSHistory.LastServiceResponse.NextToken  
} while ($nextToken -ne $null)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오  
[DescribeCommunications](#).

## Get-ASAService

다음 코드 예시에서는 Get-ASAService을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 사용 가능한 모든 서비스 코드, 이름 및 카테고리를 반환합니다.

```
Get-ASAService
```

예 2: 지정된 코드를 사용하여 서비스의 이름 및 범주를 반환합니다.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

예 3: 지정된 서비스 코드의 이름 및 범주를 반환합니다.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

예 4: 지정된 서비스 코드의 이름 및 범주 (일본어) 를 반환합니다. 현재 영어 (“en”) 및 일본어 (“ja”) 언어 코드가 지원됩니다.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -  
Language "ja"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeServices](#).

## Get-ASASeverityLevel

다음 코드 예시에서는 Get-ASASeverityLevel을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: AWS Support 사례에 할당할 수 있는 심각도 수준 목록을 반환합니다.

```
Get-ASASeverityLevel
```

예 2: AWS Support 사례에 할당할 수 있는 심각도 수준 목록을 반환합니다. 레벨 이름은 일본어로 반환됩니다.

```
Get-ASASeverityLevel -Language "ja"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeSeverityLevels](#).

## Get-ASATrustedAdvisorCheck

다음 코드 예시에서는 Get-ASATrustedAdvisorCheck을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: Trusted Advisor 검사 컬렉션을 반환합니다. 영어 출력의 경우 “en”, 일본어 출력의 경우 “ja”를 받아들일 수 있는 언어 매개 변수를 지정해야 합니다.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTrustedAdvisorChecks](#).

## Get-ASATrustedAdvisorCheckRefreshStatus

다음 코드 예시에서는 Get-ASATrustedAdvisorCheckRefreshStatus을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 지정된 검사에 대한 새로 고침 요청의 현재 상태를 반환합니다. Request-ASA는 검사의 상태 정보를 새로 고치도록 요청하는 데 사용할 TrustedAdvisorCheckRefresh 수 있습니다.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오 [DescribeTrustedAdvisorCheckRefreshStatuses](#).AWS Tools for PowerShell

## Get-ASATrustedAdvisorCheckResult

다음 코드 예시에서는 Get-ASATrustedAdvisorCheckResult을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: Trusted Advisor 검사 결과를 반환합니다. 사용 가능한 Trusted Advisor 검사 목록은 TrustedAdvisorChecks Get-ASA를 사용하여 확인할 수 있습니다. 출력은 검사의 전체 상태, 검사가 마지막으로 실행된 타임스탬프, 특정 검사의 고유한 검사 ID입니다. 결과를 일본어로 출력하려면 -Language "ja" 매개 변수를 추가하십시오.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTrustedAdvisorCheckResult](#).

## Get-ASATrustedAdvisorCheckSummary

다음 코드 예시에서는 Get-ASATrustedAdvisorCheckSummary을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예 1: 지정된 Trusted Advisor 검사에 대한 최신 요약을 반환합니다.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

예 2: 지정된 Trusted Advisor 검사에 대한 최신 요약을 반환합니다.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeTrustedAdvisorCheckSummaries](#).

## New-ASACase

다음 코드 예시에서는 New-ASACase을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: AWS Support 센터에서 새 케이스를 생성합니다. - ServiceCode 및 - CategoryCode 매개 변수의 값은 Get-asaService cmdlet을 사용하여 가져올 수 있습니다. - SeverityCode 매개 변수의 값은 Get-ASA cmdlet을 사용하여 가져올 수 있습니다. SeverityLevel - IssueType 매개 변수 값은 “고객 서비스” 또는 “기술”일 수 있습니다. 성공하면 AWS Support 케이스 번호가 출력됩니다. 기본적으로 케이스는 영어로 처리되며, 일본어를 사용하려면 -Language “ja” 매개변수를 추가하십시오. - ServiceCode, -CategoryCode, -제목 및 - CommunicationBody 매개변수는 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 참조를 참조하십시오 [CreateCase](#).

## Request-ASATrustedAdvisorCheckRefresh

다음 코드 예시에서는 Request-ASATrustedAdvisorCheckRefresh을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

예 1: 지정된 Trusted Advisor 검사에 대한 새로 고침을 요청합니다.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RefreshTrustedAdvisorCheck](#).

## Resolve-ASACase

다음 코드 예시에서는 Resolve-ASACase을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 지정된 케이스의 초기 상태와 해결 호출이 완료된 후의 현재 상태를 반환합니다.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ResolveCase](#).

## 다음 툴을 사용한 Systems Manager 예제 PowerShell

다음 코드 예제는 AWS Tools for PowerShell with Systems Manager를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### Add-SSMResourceTag

다음 코드 예시에서는 Add-SSMResourceTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

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
```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 각 태그를 생성해야 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 참조를 참조하십시오 [AddTagsToResource](#).

### Edit-SSMDocumentPermission

다음 코드 예시에서는 Edit-SSMDocumentPermission을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyDocumentPermission](#).

## Get-SSMActivation

다음 코드 예시에서는 Get-SSMActivation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeActivations](#).

## Get-SSMAssociation

다음 코드 예시에서는 Get-SSMAssociation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAssociation](#).

**Get-SSMAssociationExecution**

다음 코드 예시에서는 Get-SSMAssociationExecution을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAssociationExecutions](#).

**Get-SSMAssociationExecutionTarget**

다음 코드 예시에서는 Get-SSMAssociationExecutionTarget을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAssociationExecutionTargets](#).

## Get-SSMAssociationList

다음 코드 예시에서는 Get-SSMAssociationList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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}
```

예 3: PowerShell 버전 2에서는 New-Object를 사용하여 각 필터를 만들어야 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 참조를 참조하십시오 [ListAssociations](#).

## Get-SSMAssociationVersionList

다음 코드 예시에서는 Get-SSMAssociationVersionList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListAssociationVersions](#).

## Get-SSMAutomationExecution

다음 코드 예시에서는 Get-SSMAutomationExecution을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
                             Yb6TI5b0GQeeNrpwNvpDzm0-
                             PSR1swlAbg9fdM9BcNjyrznsUpkWpuKu9EC10u6v30XU1KC9nZ7mPlWMFZNkSioQqpWwEvMw-
                             GZktsQzm67q0hUhBN0LWYhbS
                             pkfiqzY-5nw3S0obx30fhd3EJa50_-
                             GjV_a0nFXQJa70ik40bF0rEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84X0F5WmA60f1pi5GI08i-
                             nRfZS6oDeU
                             gELBjjoFKD8s3L2aI0B6umWVxnQ0jqhQRxwJ53b54sZJ2PW3v_mtg9-
                             q0CK0ezS3xfh_y0ilaUG0AZG-xjQFuvU_JZedWp1a3xi-MZsmb1AifBI
                             (Service: AmazonEC2; Status Code: 403; Error Code:
                             UnauthorizedOperation; Request ID:
                             6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                     : {[createImage.ImageId,
                             Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                  : {[AutomationAssumeRole,
                             Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,
                             Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,
                             Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions              : {launchInstance, updateOSSoftware, stopInstance,
                             createImage...}
```

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	{}

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetAutomationExecution](#).

## Get-SSMAutomationExecutionList

다음 코드 예시에서는 Get-SSMAutomationExecutionList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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]]}

```

예 2: 이 예제에서는 '성공' 이외의 실행에 대한 ExecutionID, 문서, 실행 시작/종료 타임스탬프를 표시합니다. AutomationExecutionStatus

```

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 Cmdlet 참조를 참조하십시오. [DescribeAutomationExecutions](#) AWS Tools for PowerShell

## Get-SSMAutomationStepExecution

다음 코드 예시에서는 Get-SSMAutomationStepExecution을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeAutomationStepExecutions](#).

## Get-SSMAvailablePatch

다음 코드 예시에서는 Get-SSMAvailablePatch을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
...

```

예 2: PowerShell 버전 2에서는 New-Object를 사용하여 각 필터를 생성해야 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

```

예 3: 이 예제는 지난 20일 동안 릴리스되고 2019와 일치하는 제품에 적용할 수 있는 모든 업데이트를 가져옵니다. WindowsServer

```

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 참조를 참조하십시오  
[DescribeAvailablePatches](#).

## Get-SSMCommand

다음 코드 예시에서는 Get-SSMCommand을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListCommands](#).

## Get-SSMCommandInvocation

다음 코드 예시에서는 Get-SSMCommandInvocation을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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    :
```

예 2: 이 예제에서는 명령 ID CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f 명령을 호출하는 방법을 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/KUM0.png to ..\..\programdata\KUM0.png
                kumo available

OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     : eu-west-1
ResponseCode       : 0
```

```
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- AWS Tools for PowerShell API에 [ListCommandInvocations](#) 대한 자세한 내용은 Cmdlet 참조를 참조하십시오.

## Get-SSMCommandInvocationDetail

다음 코드 예시에서는 Get-SSMCommandInvocationDetail을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetCommandInvocation](#).

## Get-SSMComplianceItemList

다음 코드 예시에서는 Get-SSMComplianceItemList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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              :
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListComplianceItems](#).

## Get-SSMComplianceSummaryList

다음 코드 예시에서는 Get-SSMComplianceSummaryList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListComplianceSummaries](#).

## Get-SSMConnectionStatus

다음 코드 예시에서는 Get-SSMConnectionStatus을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetConnectionStatus](#).

## Get-SSMDefaultPatchBaseline

다음 코드 예시에서는 Get-SSMDefaultPatchBaseline을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 기본 패치 기준을 표시합니다.

```
Get-SSMDefaultPatchBaseline
```

출력:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDefaultPatchBaseline](#).

## Get-SSMDeployablePatchSnapshotForInstance

다음 코드 예시에서는 Get-SSMDeployablePatchSnapshotForInstance을 사용하는 방법을 보여 줍니다.

### 도구: PowerShell

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...
```

예 2: 이 예제에서는 전체를 가져오는 방법을 보여줍니다 SnapshotDownloadUrl. 이 명령은 인스턴스 자격 증명을 사용하여 인스턴스에서 실행해야 합니다. 이 예제에서는 인스턴스 자격 증명을 사

용하는지 확인하기 위해 객체를 **Amazon.Runtime.InstanceProfileAWSCredentials** 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 참조를 참조하십시오 [GetDeployablePatchSnapshotForInstance](#).

## Get-SSMDocument

다음 코드 예시에서는 Get-SSMDocument을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 }}"
      }
    }
  ]
}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetDocument](#).

## Get-SSMDocumentDescription

다음 코드 예시에서는 Get-SSMDocumentDescription을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeDocument](#).

## Get-SSMDocumentList

다음 코드 예시에서는 Get-SSMDocumentList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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       :

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDocuments](#).

## Get-SSMDocumentPermission

다음 코드 예시에서는 Get-SSMDocumentPermission을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeDocumentPermission](#).

## Get-SSMDocumentVersionList

다음 코드 예시에서는 Get-SSMDocumentVersionList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 문서에 대한 권한 목록을 반환합니다.

```
Get-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share"
```

출력:

```
all
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListDocumentVersions](#).

## Get-SSMEffectiveInstanceAssociationList

다음 코드 예시에서는 Get-SSMEffectiveInstanceAssociationList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 }}"
      }
    ]
  }
}

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeEffectiveInstanceAssociations](#).

## Get-SSMEffectivePatchesForPatchBaseline

다음 코드 예시에서는 Get-SSMEffectivePatchesForPatchBaseline을 사용하는 방법을 보여줍니다.

도구: PowerShell

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

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeEffectivePatchesForPatchBaseline](#).

## Get-SSMInstanceAssociationsStatus

다음 코드 예시에서는 Get-SSMInstanceAssociationsStatus을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeInstanceAssociationsStatus](#).

## Get-SSMInstanceInformation

다음 코드 예시에서는 Get-SSMInstanceInformation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

예 2: 이 예제에서는 -Filter 매개 변수를 사용하여 a가 **us-east-1** 있는 지역의 AWS Systems Manager 인스턴스로만 결과를 필터링하는 방법을 보여줍니다. **AgentVersion 2.2.800.0** 유효한 -Filter 키 값 목록은 InstanceInformation API 참조 항목 (<https://docs.aws.amazon.com/systems->

manager/latest/APIReference/API\_InstanceInformation .html #systemsmanager -Type- -) 에서 찾을 수 있습니다. InstanceInformation ActivationId

```
$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 예에서는 - InstanceInformationFilterList 매개변수를 사용하여 **Windows** 또는 가 있는 지역의 AWS **us-east-1** Systems Manager 인스턴스로만 결과를 필터링하는 방법을 보여줍니다. **Linux**. **PlatformTypes** 유효한 - InstanceInformationFilterList 키 값 목록은 InstanceInformationFilter API 참조 항목 ([https://docs.aws.amazon.com/systems-manager/latest/APIReference/API\\_InstanceInformationFilter.html](https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html)) 에서 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

```

예 4: 이 예제에서는 ssm 관리형 인스턴스와 내보내기 InstanceId PingStatus, LastPingDateTime csv 파일로의 PlatformName 내보내기를 나열합니다.

```

Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
PlatformName | Export-Csv Instance-details.csv -NoTypeInfoInformation

```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeInstanceInformation](#).

## Get-SSMInstancePatch

다음 코드 예시에서는 Get-SSMInstancePatch를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 인스턴스에 대한 패치 규정 준수 세부 정보를 가져옵니다.

```

Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeInstancePatches](#).

## Get-SSMInstancePatchState

다음 코드 예시에서는 Get-SSMInstancePatchState을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeInstancePatchStates](#).

## Get-SSMInstancePatchStatesForPatchGroup

다음 코드 예시에서는 Get-SSMInstancePatchStatesForPatchGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 패치 그룹의 인스턴스당 패치 요약 상태를 가져옵니다.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeInstancePatchStatesForPatchGroup](#).

## Get-SSMInventory

다음 코드 예시에서는 Get-SSMInventory을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 인벤토리의 사용자 지정 메타데이터를 가져옵니다.

```
Get-SSMInventory
```

출력:

```
Data
  Id
----
--
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetInventory](#).

## Get-SSMInventoryEntriesList

다음 코드 예시에서는 Get-SSMInventoryEntriesList를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListInventoryEntries](#).

## Get-SSMInventoryEntryList

다음 코드 예시에서는 Get-SSMInventoryEntryList를 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제는 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 Cmdlet 참조의 [Get-SSM을 InventoryEntryList](#) 참조하십시오. AWS Tools for PowerShell

## Get-SSMInventorySchema

다음 코드 예시에서는 Get-SSMInventorySchema을 사용하는 방법을 보여 줍니다.

예 1: 이 예제에서는 계정의 인벤토리 유형 이름 목록을 반환합니다.

```
Get-SSMInventorySchema
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetInventorySchema](#).

## Get-SSMLatestEC2Image

다음 코드 예시에서는 Get-SSMLatestEC2Image을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예 1: 이 예제에는 최신 Windows AMI가 모두 나열되어 있습니다.

```
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	
...	

예 2: 이 예제는 us-west-2 지역에 대한 특정 아마존 리눅스 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
```

예 3: 이 예제는 지정된 와일드카드 표현식과 일치하는 최신 Windows AMI를 모두 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 Cmdlet 참조의 [Get-SSM 최신 EC2 이미지를](#) 참조하십시오.AWS Tools for PowerShell

## Get-SSMMaintenanceWindow

다음 코드 예시에서는 Get-SSMMaintenanceWindow을 사용하는 방법을 보여 줍니다.

예에 대한 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetMaintenanceWindow](#).

## Get-SSMMaintenanceWindowExecution

다음 코드 예시에서는 Get-SSMMaintenanceWindowExecution을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetMaintenanceWindowExecution](#).

## Get-SSMMaintenanceWindowExecutionList

다음 코드 예시에서는 Get-SSMMaintenanceWindowExecutionList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindowExecutions](#).

## Get-SSMMaintenanceWindowExecutionTask

다음 코드 예시에서는 Get-SSMMaintenanceWindowExecutionTask을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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.SimpleSystemsManag
    meterValueExpression]}
Type            : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetMaintenanceWindowExecutionTask](#).

## Get-SSMMaintenanceWindowExecutionTaskInvocationList

다음 코드 예시에서는 Get-SSMMaintenanceWindowExecutionTaskInvocationList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

- 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 :
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindowExecutionTaskInvocations](#).

## Get-SSMMaintenanceWindowExecutionTaskList

다음 코드 예시에서는 Get-SSMMaintenanceWindowExecutionTaskList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindowExecutionTasks](#).

## Get-SSMMaintenanceWindowList

다음 코드 예시에서는 Get-SSMMaintenanceWindowList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindows](#).

**Get-SSMMaintenanceWindowTarget**

다음 코드 예시에서는 Get-SSMMaintenanceWindowTarget을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindowTargets](#).

**Get-SSMMaintenanceWindowTaskList**

다음 코드 예시에서는 Get-SSMMaintenanceWindowTaskList을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeMaintenanceWindowTasks](#).

## Get-SSMParameterHistory

다음 코드 예시에서는 Get-SSMParameterHistory를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetParameterHistory](#).

## Get-SSMParameterList

다음 코드 예시에서는 Get-SSMParameterList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeParameters](#).

## Get-SSMParameterValue

다음 코드 예시에서는 Get-SSMParameterValue을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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!
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetParameters](#).

## Get-SSMPatchBaseline

다음 코드 예시에서는 Get-SSMPatchBaseline을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 모든 패치 기준을 나열합니다.

```
Get-SSMPatchBaseline
```

출력:

```
BaselineDescription                                     BaselineId
-----
Default Patch Baseline Provided by AWS.                arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
Production-B...
```

예 2: 이 예제에서는 에서 제공하는 모든 패치 기준을 나열합니다. AWS이 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
```

예 4: PowerShell 버전 2에서는 New-Object를 사용하여 각 태그를 생성해야 합니다.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"
```

```
Get-SSMPatchBaseline -Filter $filter1
```

출력:

BaselineDescription	BaselineName	BaselineId	DefaultBaseline
-----	-----	-----	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 참조를 참조하십시오 [DescribePatchBaselines](#).

## Get-SSMPatchBaselineDetail

다음 코드 예시에서는 Get-SSMPatchBaselineDetail을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

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    : {}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPatchBaseline](#).

## Get-SSMPatchBaselineForPatchGroup

다음 코드 예시에서는 Get-SSMPatchBaselineForPatchGroup을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 패치 그룹의 패치 기준을 표시합니다.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

출력:

```
BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [GetPatchBaselineForPatchGroup](#).

## Get-SSMPatchGroup

다음 코드 예시에서는 Get-SSMPatchGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 패치 그룹 등록을 나열합니다.

```
Get-SSMPatchGroup
```

출력:

```
BaselineIdentity                                PatchGroup
-----
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribePatchGroups](#).

## Get-SSMPatchGroupState

다음 코드 예시에서는 Get-SSMPatchGroupState을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribePatchGroupState](#).

## Get-SSMResourceComplianceSummaryList

다음 코드 예시에서는 Get-SSMResourceComplianceSummaryList을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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. 매개 변수가 지정되지 않은 경우 MaxResult 기본값은 100이고 이 값은 유효하지 않으므로 MaxResult 매개 변수가 추가되고 값은 50으로 설정됩니다.

```
$FilterValues = @{
    "Key"="Product"
    "Type"="EQUAL"
    "Values"="Windows10"
}
Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListResourceComplianceSummaries](#).

## Get-SSMResourceTag

다음 코드 예시에서는 Get-SSMResourceTag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 유지 관리 기간의 태그를 나열합니다.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
"MaintenanceWindow"
```

출력:

```
Key    Value
---    -
Stack Production
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ListTagsForResource](#).

## New-SSMActivation

다음 코드 예시에서는 New-SSMActivation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 관리형 인스턴스를 생성합니다.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

출력:

```
ActivationCode      ActivationId
-----
KWChh0xBTiwDcKE9B1KC 08e51e79-1e36-446c-8e63-9458569c1363
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateActivation](#).

## New-SSMAssociation

다음 코드 예시에서는 New-SSMAssociation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 :
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateAssociation](#).

**New-SSMAssociationFromBatch**

다음 코드 예시에서는 New-SSMAssociationFromBatch을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateAssociationBatch](#).

## New-SSMDocument

다음 코드 예시에서는 New-SSMDocument을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateDocument](#).

## New-SSMMaintenanceWindow

다음 코드 예시에서는 New-SSMMaintenanceWindow를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreateMaintenanceWindow](#).

## New-SSMPatchBaseline

다음 코드 예시에서는 New-SSMPatchBaseline을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 Microsoft에서 릴리스하고 7일 후에 프로덕션 환경에서 Windows Server 2019를 실행하는 관리형 인스턴스에 대한 패치를 승인하는 패치치 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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreatePatchBaseline](#).

## Register-SSMDefaultPatchBaseline

다음 코드 예시에서는 Register-SSMDefaultPatchBaseline을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 패치 기준을 기본 패치 기준으로 등록합니다.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

출력:

```
pb-03da896ca3b68b639
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterDefaultPatchBaseline](#).

## Register-SSMPatchBaselineForPatchGroup

다음 코드 예시에서는 Register-SSMPatchBaselineForPatchGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 패치 그룹의 패치 기준을 등록합니다.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

출력:

```
BaselineId          PatchGroup
-----
pb-03da896ca3b68b639 Production
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterPatchBaselineForPatchGroup](#).

## Register-SSMTargetWithMaintenanceWindow

다음 코드 예시에서는 Register-SSMTargetWithMaintenanceWindow을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterTargetWithMaintenanceWindow](#).

## Register-SSMTaskWithMaintenanceWindow

다음 코드 예시에서는 Register-SSMTaskWithMaintenanceWindow를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

예 4: 이 예제에서는 라는 **Create-Snapshots** 문서를 사용하여 AWS Systems Manager 자동화 작업을 등록합니다.

```

$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 참조를 참조하십시오 [RegisterTaskWithMaintenanceWindow](#).

## Remove-SSMActivation

다음 코드 예시에서는 Remove-SSMActivation을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 활성화를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteActivation](#).

## Remove-SSMAssociation

다음 코드 예시에서는 Remove-SSMAssociation을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 인스턴스와 문서 간의 연결을 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteAssociation](#).

## Remove-SSMDocument

다음 코드 예시에서는 Remove-SSMDocument을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 문서를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMDocument -Name "RunShellScript"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteDocument](#).

## Remove-SSMMaintenanceWindow

다음 코드 예시에서는 Remove-SSMMaintenanceWindow을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 유지 관리 기간을 제거합니다.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

출력:

```
mw-06d59c1a07c022145
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteMaintenanceWindow](#).

## Remove-SSMParameter

다음 코드 예시에서는 Remove-SSMParameter를 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 파라미터를 삭제합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMParameter -Name "helloWorld"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeleteParameter](#).

## Remove-SSMPatchBaseline

다음 코드 예시에서는 Remove-SSMPatchBaseline을 사용하는 방법을 보여 줍니다.

## 도구: PowerShell

예제 1: 이 예제에서는 패치 기준을 삭제합니다.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

출력:

```
pb-045f10b4f382baeda
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeletePatchBaseline](#).

## Remove-SSMResourceTag

다음 코드 예시에서는 Remove-SSMResourceTag을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 유지 관리 기간에서 태그를 제거합니다. 명령이 성공해도 출력은 없습니다.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
  "MaintenanceWindow" -TagKey "Production"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RemoveTagsFromResource](#).

## Send-SSMCommand

다음 코드 예시에서는 Send-SSMCommand을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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-Win320penSSH"}'; "commandLine"=".\\Install-
Win320penSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [SendCommand](#).

## Start-SSMAutomationExecution

다음 코드 예시에서는 Start-SSMAutomationExecution을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StartAutomationExecution](#).

## Stop-SSMAutomationExecution

다음 코드 예시에서는 Stop-SSMAutomationExecution을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 자동화 실행을 중지합니다. 명령이 성공해도 출력은 없습니다.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-  
f944-11e6-9d32-8fb2db27a909"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [StopAutomationExecution](#).

## Stop-SSMCommand

다음 코드 예시에서는 Stop-SSMCommand을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 명령 취소를 시도합니다. 작업이 성공해도 출력은 없습니다.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CancelCommand](#).

## Unregister-SSMManagedInstance

다음 코드 예시에서는 Unregister-SSMManagedInstance을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 관리형 인스턴스를 등록 취소합니다. 명령이 성공해도 출력은 없습니다.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeregisterManagedInstance](#).

## Unregister-SSMPatchBaselineForPatchGroup

다음 코드 예시에서는 Unregister-SSMPatchBaselineForPatchGroup을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 패치 기준에서 패치 그룹을 등록 취소합니다.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

출력:

```
BaselineId          PatchGroup
-----
pb-045f10b4f382baeda Production
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeregisterPatchBaselineForPatchGroup](#).

## Unregister-SSMTargetFromMaintenanceWindow

다음 코드 예시에서는 Unregister-SSMTargetFromMaintenanceWindow을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeregisterTargetFromMaintenanceWindow](#).

## Unregister-SSMTaskFromMaintenanceWindow

다음 코드 예시에서는 Unregister-SSMTaskFromMaintenanceWindow를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DeregisterTaskFromMaintenanceWindow](#).

## Update-SSMAssociation

다음 코드 예시에서는 Update-SSMAssociation을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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 :
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAssociation](#).

## Update-SSMAssociationStatus

다음 코드 예시에서는 Update-SSMAssociationStatus을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateAssociationStatus](#).

## Update-SSMDocument

다음 코드 예시에서는 Update-SSMDocument을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 지정한 json 파일의 업데이트된 콘텐츠를 포함하는 문서의 새 버전을 생성합니다. 문서는 JSON 형식이어야 합니다. "Get-SSMDocumentVersionList" cmdlet을 사용하여 문서 버전을 구할 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

```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [UpdateDocument](#) AWS Tools for PowerShell

**Update-SSMDocumentDefaultVersion**

다음 코드 예시에서는 Update-SSMDocumentDefaultVersion을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 여기에서는 문서의 기본 버전을 업데이트합니다. "Get-SSMDocumentVersionList" cmdlet 을 사용하여 사용 가능한 문서 버전을 얻을 수 있습니다.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

**출력:**

```

DefaultVersion Name
-----
2              RunShellScript

```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [UpdateDocumentDefaultVersion](#) AWS Tools for PowerShell

## Update-SSMMaintenanceWindow

다음 코드 예시에서는 Update-SSMMaintenanceWindow를 사용하는 방법을 보여 줍니다.

도구: PowerShell

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

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateMaintenanceWindow](#).

**Update-SSManagedInstanceRole**

다음 코드 예시에서는 Update-SSManagedInstanceRole을 사용하는 방법을 보여 줍니다.

도구: PowerShell

예제 1: 이 예제에서는 관리형 인스턴스의 역할을 업데이트합니다. 명령이 성공해도 출력은 없습니다.

```

Update-SSManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
"AutomationRole"

```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdateManagedInstanceRole](#).

**Update-SSMPatchBaseline**

다음 코드 예시에서는 Update-SSMPatchBaseline을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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}
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [UpdatePatchBaseline](#).

## Write-SSMComplianceItem

다음 코드 예시에서는 Write-SSMComplianceItem을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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"
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutComplianceItems](#).

## Write-SSMInventory

다음 코드 예시에서는 Write-SSMInventory을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutInventory](#).

## Write-SSMParameter

다음 코드 예시에서는 Write-SSMParameter을 사용하는 방법을 보여 줍니다.

도구: PowerShell

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
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [PutParameter](#).

## 다음 도구를 사용한 Amazon Translate 예제 PowerShell

다음 코드 예제는 Amazon Translate와 AWS Tools for PowerShell 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

### 작업

#### ConvertTo-TRNTargetLanguage

다음 코드 예시에서는 ConvertTo-TRNTargetLanguage을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 지정된 영어 텍스트를 프랑스어로 변환합니다. 변환할 텍스트를 -Text 매개 변수로 전달할 수도 있습니다.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- API에 대한 자세한 내용은 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [TranslateText](#).

## AWS WAFV2 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS WAFV2. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. [GitHub](#)

주제

- [작업](#)

## 작업

### New-WAF2WebACL

다음 코드 예시에서는 New-WAF2WebACL을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 명령은 "waf-test"라는 새 웹 ACL을 생성합니다. 참고로 서비스 API 설명서에 따르면 DefaultAction "는 필수 속성입니다. 따라서 '- DefaultAction \_Allow' 및/또는 '- DefaultAction \_Block' 값을 지정해야 합니다. DefaultAction'-\_Allow' 및 '- DefaultAction \_Block'은 필수 속성이 아니므로 위 예와 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 Cmdlet 참조를 참조하십시오. [CreateWebAcl](#) AWS Tools for PowerShell

## WorkSpaces 다음과 같은 도구를 사용하는 예 PowerShell

다음 코드 예제는 with를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 WorkSpaces. AWS Tools for PowerShell

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

각 예제에는 컨텍스트에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

주제

- [작업](#)

작업

### Approve-WKSIpRule

다음 코드 예시에서는 Approve-WKSIpRule을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 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 참조를 참조하십시오 [AuthorizeIpRules](#).

### Copy-WKSWorkspaceImage

다음 코드 예시에서는 Copy-WKSWorkspaceImage을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 지정된 ID의 작업 공간 이미지를 us-west-2에서 이름이 ""인 현재 지역으로 복사합니다. CopiedImageTest

예 1: 이 샘플은 지정된 ID의 작업 공간 이미지를 us-west-2에서 이름이 ""인 현재 지역으로 복사합니다. CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId wsi-djfoedhw6
```

출력:

```
wsi-456abaqfe
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [CopyWorkspaceImage](#). AWS Tools for PowerShell

## Edit-WKSClientProperty

다음 코드 예시에서는 Edit-WKSClientProperty을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 Workspaces 클라이언트의 재연결을 활성화합니다.

예 1: 이 샘플은 Workspaces 클라이언트의 재연결을 활성화합니다.

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled "ENABLED" -ResourceId d-123414a369
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyClientProperties](#).

## Edit-WKSSelfServicePermission

다음 코드 예시에서는 Edit-WKSSelfServicePermission을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 셀프 서비스 권한을 사용하여 지정된 디렉터리의 컴퓨팅 유형을 변경하고 볼륨 크기를 늘릴 수 있도록 합니다.

예 1: 이 샘플은 셀프 서비스 권한을 사용하여 지정된 디렉터리의 컴퓨팅 유형을 변경하고 볼륨 크기를 늘릴 수 있도록 합니다.

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifySelfservicePermissions](#).

## Edit-WKSWorkspaceAccessProperty

다음 코드 예시에서는 Edit-WKSWorkspaceAccessProperty을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 Android 및 Chrome OS에서 지정된 디렉터리의 워크스페이스 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 참조를 참조하십시오 [ModifyWorkspaceAccessProperties](#).

## Edit-WKSWorkspaceCreationProperty

다음 코드 예시에서는 Edit-WKSWorkspaceCreationProperty을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 이 샘플은 Workspace를 생성할 때 인터넷 액세스 및 유지 관리 모드를 기본값으로 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 참조를 참조하십시오 [ModifyWorkspaceCreationProperties](#).

## Edit-WKSWorkspaceProperty

다음 코드 예시에서는 Edit-WKSWorkspaceProperty을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지정된 작업 영역의 작업 영역 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 참조를 참조하십시오 [ModifyWorkspaceProperties](#).

## Edit-WKSWorkspaceState

다음 코드 예시에서는 Edit-WKSWorkspaceState을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지정된 작업 공간의 상태를 사용 가능으로 변경합니다.

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState  
AVAILABLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [ModifyWorkspaceState](#).

## Get-WKSClientProperty

다음 코드 예시에서는 Get-WKSClientProperty을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지정된 디렉터리에 대한 Workspace 클라이언트의 클라이언트 속성을 가져옵니다.

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet [DescribeClientProperties](#) 참조의 내용을 참조하십시오.

## Get-WKSIpGroup

다음 코드 예시에서는 Get-WKSIpGroup을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 지정된 지역의 지정된 IP 그룹에 대한 세부 정보를 가져옵니다.

예 1: 이 샘플은 지정된 지역의 지정된 IP 그룹에 대한 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 참조를 참조하십시오 [DescribeIpGroups](#).

## Get-WKSTag

다음 코드 예시에서는 Get-WKSTag을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 지정된 Workspace의 태그를 가져옵니다.

예 1: 이 샘플은 지정된 Workspace의 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 참조를 참조하십시오 [DescribeTags](#).

## Get-WKSWorkspace

다음 코드 예시에서는 Get-WKSWorkspace을 사용하는 방법을 보여 줍니다.

## 예 1: 모든 연결 파이프라인의 세부 정보를 검색합니다. WorkSpaces

예 1: 모든 연결 파이프라인의 세부 정보를 검색합니다. WorkSpaces

```
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
```

예 2: 이 명령은 해당 **us-west-2** 지역의 작업 공간에 **WorkspaceProperties** 대한 하위 속성 값을 보여줍니다. 의 **WorkspaceProperties** 하위 속성에 대한 자세한 내용은 [https://docs.aws.amazon.com/workspaces/latest/api/API\\_WorkspaceProperties.html](https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html)을 참조하십시오.

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-xdaf7hc9s).WorkspaceProperties
```

출력:

```
ComputeTypeName      : STANDARD
RootVolumeSizeGib   : 80
RunningMode          : AUTO_STOP
RunningModeAutoStopTimeoutInMinutes : 60
UserVolumeSizeGib    : 50
```

예 3: 이 명령은 해당 **us-west-2** 지역의 작업 공간에 **RootVolumeSizeGib** **WorkspaceProperties** 대한 하위 속성 값을 보여줍니다. 루트 볼륨 크기 (GiB) 는 80입니다.

```
(Get-WKSWorkspace -Region us-west-2 -WorkspaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

출력:

```
80
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeWorkspaces](#).

## Get-WKSWorkspaceBundle

다음 코드 예시에서는 Get-WKSWorkspaceBundle을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 현재 지역에 있는 모든 Workspace 번들의 세부 정보를 가져옵니다.

```
Get-WKSWorkspaceBundle
```

출력:

```
BundleId       : wsb-sfhgfv342
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 Cmdlet 참조를 참조하십시오 [DescribeWorkspaceBundles](#). AWS Tools for PowerShell

## Get-WKSWorkspaceDirectory

다음 코드 예시에서는 Get-WKSWorkspaceDirectory을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 등록된 디렉터리의 디렉터리 세부 정보를 나열합니다.

예 1: 이 샘플은 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 참조를 참조하십시오 [DescribeWorkspaceDirectories](#).

## Get-WKSWorkspaceImage

다음 코드 예시에서는 Get-WKSWorkspaceImage을 사용하는 방법을 보여 줍니다.

## 예 1: 이 샘플은 해당 지역의 모든 이미지에 대한 모든 세부 정보를 가져옵니다.

예 1: 이 샘플은 해당 지역의 모든 이미지에 대한 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 참조를 참조하십시오 [DescribeWorkpaceImages](#).

## Get-WKSWorkspaceSnapshot

다음 코드 예시에서는 Get-WKSWorkspaceSnapshot을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 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 참조를 참조하십시오 [DescribeWorkspaceSnapshots](#).

## Get-WKSWorkspacesConnectionStatus

다음 코드 예시에서는 Get-WKSWorkspacesConnectionStatus을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 지정된 Workspace의 연결 상태를 가져옵니다.

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DescribeWorkspacesConnectionStatus](#).

## New-WKSIpGroup

다음 코드 예시에서는 New-WKSIpGroup을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 샘플은 이름이 지정된 빈 Ip 그룹을 만듭니다. FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

출력:

```
wsipg-w45rty4ty
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [CreatelpGroup](#).

## New-WKSTag

다음 코드 예시에서는 New-WKSTag을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 이 예제에서는 라는 이름의 작업 공간에 새 태그를 추가합니다 **ws-wsname**. 태그의 키는 "Name"이고 키 값은 입니다 **AWS\_Workspace**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

예 2: 이 예제에서는 라는 작업 공간에 여러 태그를 추가합니다 **ws-wsname**. 한 태그의 키는 "Name"이고 키 값은 **AWS\_Workspace 1**이고 다른 태그의 태그 키는 "Stage"이고 키 값은 "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 참조를 참조하십시오 [CreateTags](#).

## New-WKSWorkspace

다음 코드 예시에서는 New-WKSWorkspace을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 제공된 번들, 디렉토리 및 사용자에게 Workspace 대한 a를 생성합니다.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
"d-1a2b3c4d"; "UserName" = "USERNAME"}
```

- 예 2: 이 예제에서는 여러 개를 생성합니다. WorkSpaces

```
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 참조를 참조하십시오 [CreateWorkspaces](#).

## Register-WKSIpGroup

다음 코드 예시에서는 Register-WKSIpGroup을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

- 예 1: 이 샘플은 지정된 IP 그룹을 지정된 디렉터리에 등록합니다.

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [AssociateIpGroups](#).

## Register-WKSWorkspaceDirectory

다음 코드 예시에서는 Register-WKSWorkspaceDirectory을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 PowerShell

예 1: 이 샘플은 Workspace Service의 지정된 디렉토리를 등록합니다.

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [RegisterWorkspaceDirectory](#).

## Remove-WKSIpGroup

다음 코드 예시에서는 Remove-WKSIpGroup을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 PowerShell

예 1: 이 샘플은 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 참조를 참조하십시오 [DeleteIpGroup](#).

## Remove-WKSTag

다음 코드 예시에서는 Remove-WKSTag을 사용하는 방법을 보여 줍니다.

예 1: 이 샘플은 PowerShell

예 1: 이 샘플은 작업공간과 관련된 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 참조를 참조하십시오 [DeleteTags](#).

## Remove-WKSWorkspace

다음 코드 예시에서는 Remove-WKSWorkspace을 사용하는 방법을 보여 줍니다.

예 대한 도구 PowerShell

예 1: 복수를 종료합니다 WorkSpaces. -Force 스위치를 사용하면 cmdlet에서 확인 메시지를 표시하지 않습니다.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5", "ws-6a7b8c9d0" -Force
```

예 2: 모든 사용자 컬렉션을 검색하고 ID를 WorkSpaces Remove-WKsWorkspace의 -WorkspaceId 매개 변수로 파이프하여 모두 종료합니다. WorkSpaces cmdlet은 각각이 종료되기 전에 메시지를 표시합니다. Workspace 확인 프롬프트를 표시하지 않으려면 -Force 스위치를 추가하십시오.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

예 3: 이 예제에서는 종료할 를 정의하는 TerminateRequest 객체를 전달하는 방법을 보여줍니다. WorkSpaces -Force 스위치 매개 변수도 함께 지정하지 않은 경우 cmdlet은 계속 진행하기 전에 확인 메시지를 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
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오 [TerminateWorkspaces](#).AWS Tools for PowerShell

## Reset-WKSWorkspace

다음 코드 예시에서는 Reset-WKSWorkspace을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 지정된 WorkSpace 항목을 다시 빌드합니다.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

예 2: 모든 사용자 WorkSpaces 컬렉션을 검색하고 ID를 Reset-WKsWorkspace의 - WorkspaceId 매개 변수로 파이핑하여 이 다시 빌드되도록 합니다. WorkSpaces

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- API 세부 정보는 Cmdlet 참조를 참조하십시오. [RebuildWorkspaces](#)AWS Tools for PowerShell

## Restart-WKSWorkspace

다음 코드 예시에서는 Restart-WKSWorkspace을 사용하는 방법을 보여 줍니다.

에 대한 도구 PowerShell

예 1: 지정된 WorkSpace 항목을 재부팅합니다.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

예 2: 여러 번 재부팅합니다. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d", "ws-5a6b7c8d"
```

예 3: 모든 사용자 WorkSpaces 컬렉션을 검색하고 ID를 Restart-WksWorkspace의 - WorkspaceId 매개 변수로 파이핑하여 를 다시 시작합니다. WorkSpaces

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- API에 대한 자세한 내용은 Cmdlet 참조를 참조하십시오. [RebootWorkspaces](#) AWS Tools for PowerShell

## Stop-WKSWorkspace

다음 코드 예시에서는 Stop-WKSWorkspace을 사용하는 방법을 보여 줍니다.

예 1: 여러 개를 WorkSpaces 중지합니다.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5", "ws-6a7b8c9d0"
```

예 2: 모든 사용자 WorkSpaces 컬렉션을 검색하고 ID를 Stop-WKSWorkspace의 - WorkspaceId 매개 변수로 파이핑하여 를 중지시킵니다. WorkSpaces

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

예 3: 이 예제에서는 중지할 StopRequest 대상을 정의하는 객체를 전달하는 방법을 보여줍니다. WorkSpaces

```
$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 참조를 참조하십시오 [StopWorkspaces](#).

## Unregister-WKSIpGroup

다음 코드 예시에서는 Unregister-WKSIpGroup을 사용하는 방법을 보여 줍니다.

## 예 대한 도구 PowerShell

예 1: 이 샘플은 지정된 디렉터리에서 지정된 IP 그룹을 등록 취소합니다.

```
Unregister-WKSIPGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조를 참조하십시오 [DisassociateIpGroups](#).

## 이 AWS 제품 또는 서비스의 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 사용자와 사용자 간의 AWS 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 — AWS 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 제공하는 역할을 합니다. AWS 당사의 보안 책임은 최우선 과제이며 AWS, [AWS 규정 준수 프로그램의 일환](#)으로 타사 감사자가 보안 효과를 정기적으로 테스트하고 검증합니다.

클라우드에서의 보안 — 사용자의 책임은 사용 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항, 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 [AWS 규정 준수 프로그램의 규정 준수 노력 범위에 속하는 AWS 서비스를 참조](#)하십시오.

### 주제

- [이 AWS 제품 또는 서비스의 데이터 보호](#)
- [ID 및 액세스 관리](#)
- [이 AWS 제품 또는 서비스에 대한 규정 준수 검증](#)
- [PowerShell용 도구에서 최소 TLS 버전 적용](#)
- [도구에 대한 추가 보안 고려 사항 PowerShell](#)

## 이 AWS 제품 또는 서비스의 데이터 보호

AWS [공동 책임 모델](#)은 이 AWS 제품 또는 서비스의 데이터 보호에 적용됩니다. 이 모델이 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [Data Privacy FAQ\(데이터 프라이버시 FAQ\)](#)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS Shared Responsibility Model and GDPR](#) 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정보안 인증 정보를 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하

면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용합니다.
- 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 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [FIPS\(Federal Information Processing Standard\) 140-2](#)를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 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 보안 인증을 URL에 포함시켜서는 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 참조하십시오.

### 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.

## ID 및 액세스 관리

AWS Identity and Access Management(IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- [고객](#)
- [보안 인증을 통한 인증](#)
- [정책을 사용한 액세스 관리](#)
- [AWS 서비스에서 IAM을 사용하는 방식](#)
- [AWS 보안 인증 및 액세스 문제 해결](#)

### 고객

AWS Identity and Access Management (IAM)를 사용하는 방법은 AWS에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 - AWS 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. AWS의 기능에 액세스할 수 없는 경우 [AWS 보안 인증 및 액세스 문제 해결](#) 또는 사용 중인 AWS 서비스의 사용 설명서를 참조하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 담당자라면 AWS에 대한 전체 액세스 권한을 가지고 있을 것입니다. 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 이해해 두세요. 회사에서 AWS와 함께 IAM을 사용하는 방법에 대한 자세한 내용은 사용 중인 AWS 서비스의 사용 설명서를 참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 AWS 보안 인증 기반 정책 예제를 보려면 사용 중인 AWS 서비스의 사용 설명서를 참조하세요.

## 보안 인증을 통한 인증

인증은 ID 보안 인증 정보를 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자 또는 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

보안 인증 정보 소스를 통해 제공된 보안 인증 정보를 사용하여 페더레이션형 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증 정보가 페더레이션형 ID의 예입니다. 페더레이션 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 [AWS 계정에 로그인하는 방법](#)을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 [AWS API 요청에 서명](#)을 참조하세요.

사용하는 인증 방법과 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS에서는 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [다중 인증](#) 및 IAM 사용 설명서의 [AWS에서 다중 인증\(MFA\) 사용](#)을 참조하세요.

### AWS 계정 루트 사용자

AWS 계정을 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 자격 증명으로 시작합니다. 이 보안 인증 정보는 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에는 루트 사용자를 가급적 사용하지 않는 것이 좋습니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 작업의 전체 목록은 IAM 사용자 안내서의 [루트 사용자 보안 인증이 필요한 작업](#)을 참조하세요.

### 페더레이션 ID

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 자격 증명 공급자와의 페더레이션을 통해 임시 보안 인증을 사용하여 AWS 서비스에 액세스하도록 요구하는 것입니다.

페더레이션 보안 인증 정보는 엔터프라이즈 사용자 디렉터리, 웹 자격 증명 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 보안 인증 정보 소스를 통해 제공된 보안 인증 정보를 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 보안 인증 정보는 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 보안 인증 정보를 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한 보안 인증 정보 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 [IAM Identity Center란 무엇인가요?](#)를 참조하세요.

## IAM 사용자 및 그룹

[IAM 사용자](#)는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 보안 인증이 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 [장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체](#)를 참조하세요.

[IAM 그룹](#)은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 귀하는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서의 [IAM 사용자를 만들어야 하는 경우\(역할이 아님\)](#)를 참조하세요.

## IAM 역할

[IAM 역할](#)은 특정 권한을 가지고 있는 AWS 계정 계정 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. [역할 전환](#)하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 [IAM 역할 사용](#)을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 보안 인증 정보가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 [서드 파티 보안 인증 정보 공급자의 역할 생성](#) 섹션을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연결합니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [권한 세트](#) 섹션을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스: IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 정책을 리소스에 직접 연결할 수 있습니다(역할을 프록시로 사용하는 대신). 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조하세요.
- 교차 서비스 액세스: 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 직접적으로 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
- 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어 집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.
- 서비스 역할: 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 수임하는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [AWS 서비스에 대한 권한을 위임할 역할 생성](#)을 참조하세요.
- 서비스 연결 역할: 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증 정보를 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역

할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증 정보를 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여](#)를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 [IAM 사용 설명서](#)의 IAM 역할(사용자 대신)을 생성하는 경우 섹션을 참조하세요.

## 정책을 사용한 액세스 관리

정책을 생성하고 AWS 자격 증명 또는 리소스에 연결하여 AWS 내 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되는지 또는 거부되는지를 결정합니다. 대부분의 정책은 AWS에 JSON 설명서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 내용은 IAM 사용 설명서의 [JSON 정책 개요](#)를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수입할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI 또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

## ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [IAM 정책 생성](#)을 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할

할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 [관리형 정책과 인라인 정책의 선택](#)을 참조하세요.

## 리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제로 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 [보안 주체를 지정](#)해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스가 포함될 수 있습니다.

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 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 [액세스 제어 목록\(ACL\) 개요](#)를 참조하세요.

## 기타 정책 유형

AWS는(는) 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 ID 기반 정책 및 해당 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 IAM 사용 설명서의 [IAM 엔터티에 대한 권한 경계](#)를 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹

화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자(를) 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 [SCP 작동 방식](#)을 참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에게 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책 및 세션 정책의 교집합입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM 사용 설명서의 [세션 정책](#)을 참조하세요.

## 여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지 여부를 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하세요.

## AWS 서비스에서 IAM을 사용하는 방식

AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 [IAM으로 작업하는 AWS 서비스](#)를 참조하세요.

특정 AWS 서비스에 IAM을 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 단원을 참조하세요.

## AWS 보안 인증 및 액세스 문제 해결

다음 정보를 사용하여 AWS 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

### 주제

- [AWS에서 작업을 수행할 권한이 없음](#)
- [iam:PassRole을 수행하도록 인증되지 않음](#)
- [내 AWS 계정 외부의 사람이 내 AWS 리소스에 액세스할 수 있게 허용하기를 원합니다.](#)

### AWS에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM user(IAM 사용자)가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 *aws:GetWidget* 권한이 없을 때 발생합니다.

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

이 경우 *aws:GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 보안 인증을 제공한 사용자입니다.

### 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 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신, 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 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 수 있는 권한을 가지고 있지 않습니다.

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

이 경우 Mary가 *iam:PassRole* 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의합니다. 관리자는 로그인 보안 인증 정보를 제공하는 사람입니다.

내 AWS 계정 외부의 사람이 내 AWS 리소스에 액세스할 수 있게 허용하기를 원합니다.

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.

자세히 알아보려면 다음을 참조하세요.

- AWS에서 이러한 기능을 지원하는지 여부를 알아보려면 [AWS 서비스에서 IAM을 사용하는 방식](#) 단원을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [자신이 소유한 다른 AWS 계정의 IAM 사용자에게 대한 액세스 권한 제공](#)을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 [서드 파티가 소유한 AWS 계정에 대한 액세스 제공](#)을 참조하세요.
- 자격 증명 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [외부에서 인증된 사용자에게 액세스 권한 제공\(자격 증명 페더레이션\)](#)을 참조하세요.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조하세요.

## 이 AWS 제품 또는 서비스에 대한 규정 준수 검증

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면 AWS 서비스 규정 준수 [프로그램의 AWS 서비스 범위별, 규정](#) 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 [AWS 규정 준수 프로그램](#) [AWS 보증 프로그램](#) [규정](#) [AWS](#) 참조하십시오.

를 사용하여 AWS Artifact 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 <https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html> 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- [보안 및 규정 준수 킷 스타트 가이드](#) - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- [Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.](#)

### Note

모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 [HIPAA 적격 서비스 참조](#)를 참조하십시오.

- [AWS 규정 준수 리소스AWS](#) — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- [AWS 고객 규정 준수 가이드](#) — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 [규칙을 사용하여 리소스 평가](#) — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- [AWS Security Hub](#) — 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 [Security Hub 제어 참조](#)를 참조하십시오.
- [Amazon GuardDuty](#) — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- [AWS Audit Manager](#) — 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services (AWS) 서비스를 통해 [공동 책임 모델](#)을 따릅니다. AWS 서비스 보안 정보는 [AWS 서비스 보안 설명서 페이지](#) 및 [AWS 규정 준수 프로그램의 규정 준수 노력 범위에 속하는AWS 서비스를](#) 참조하십시오.

## 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 가이드](#)에서 [최소 TLS 버전 적용](#)을 참조하세요.

## 도구에 대한 추가 보안 고려 사항 PowerShell

이 항목에는 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. 이 정보가 노출되면 특정 시나리오에서 보안 위험이 발생할 수 있습니다. 예를 들어 해당 정보가 CI/CD (지속적 통합 및 지속적 배포) 로그에 포함될 수 있습니다. 따라서 이러한 출력을 로그의 일부로 포함할 때는 검토하고 필요하지 않을 때는 출력을 표시하지 않는 것이 중요합니다. 민감한 데이터 보호에 대한 추가 정보는 [을 참조하십시오](#) [이 AWS 제품 또는 서비스의 데이터 보호](#).

다음 모범 사례를 고려하세요.

- 환경 변수를 사용하여 서버리스 리소스의 민감한 값을 저장하지 마십시오. 대신 서버리스 코드가 암호 저장소 (예:) 에서 프로그래밍 방식으로 암호를 검색하도록 하세요. AWS Secrets Manager
- 빌드 로그의 내용을 검토하여 민감한 정보가 포함되어 있지 않은지 확인하세요. /dev/null로 파일링하거나 출력을 bash 또는 변수로 캡처하여 명령 출력을 억제하는 등의 방법을 고려해 보세요.  
PowerShell
- 로그 액세스를 고려하고 사용 사례에 맞게 액세스 범위를 적절하게 지정하세요.

## 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 Reference](#)를 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.

[의 변경 사항 및 업데이트에 대한 추가 정보는 릴리스 AWS Tools for PowerShell 노트를 참조하십시오.](#)

변경 사항	설명	날짜
<a href="#">코드 예제</a>	cmdlet 예제가 포함된 장이 포함되어 있습니다.	2024년 4월 17일
<a href="#">추가 보안 고려 사항</a>	민감한 데이터의 잠재적 로깅에 대한 정보가 포함되어 있습니다.	2024년 4월 16일
<a href="#">도구 인증을 다음과 같이 구성하십시오. AWS</a>	에 SSO 지원에 대한 정보가 추가되었습니다. AWS Tools for PowerShell	2024년 3월 15일
<a href="#">도구에 대한 Cmdlet 참조 PowerShell</a>	PowerShell cmdlet용 도구 참조에 대한 링크가 포함된 섹션이 추가되었습니다.	2023년 11월 17일
<a href="#">IAM 모범 사례 업데이트 추가 포함</a>	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 <a href="#">IAM의 보안 모범 사례</a> 를 참조하십시오.	2023년 10월 12일
<a href="#">Windows에 설치</a>	더 이상 사용되지 않는 MSI를 사용한 PowerShell Windows용 도구 설치에 대한 정보가 제거되었습니다.	2023년 9월 25일
<a href="#">IAM 모범 사례 업데이트</a>	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한	2023년 9월 8일

	한 내용은 <a href="#">IAM의 보안 모범 사례</a> 를 참조하십시오.	
<a href="#">파이프라이닝 및 \$ AWSHistory</a>	Set-AWSHistoryConfiguration cmdlet에 IncludeSensitiveData 파라미터를 추가했습니다.	2023년 3월 9일
<a href="#">cmdlet의 ClientConfig 매개 변수 사용</a>	매개 변수 지원에 대한 ClientConfig 정보가 추가되었습니다.	2022년 10월 28일
<a href="#">윈도우를 사용하여 Amazon EC2 인스턴스 시작 PowerShell</a>	EC2-Classic에 대한 참고 사항이 추가되었습니다.	2022년 7월 26일
<a href="#">AWS Tools for PowerShell 버전 4</a>	<a href="#">Windows</a> 및 <a href="#">Linux/macOS</a> 에 대한 설치 지침을 포함한 버전 4에 대한 정보와 버전 3과의 차이점을 설명하고 새로운 기능을 소개하는 <a href="#">마이그레이션</a> 주제가 추가되었습니다.	2019년 11월 21일
<a href="#">AWS Tools for PowerShell 3.3.563</a>	AWS.Tools.Common 모듈의 평가판 버전을 설치하고 사용하는 방법에 대한 정보가 추가되었습니다. 이 새 모듈은 이전 모놀리식 패키지를 하나의 공유 모듈과 서비스당 하나의 모듈로 분리합니다. AWS	2019년 10월 18일
<a href="#">AWS Tools for PowerShell 3.3.343.0</a>	PowerShell Core 개발자가 함수를 빌드할 수 있는 <a href="#">AWS Lambda 도구를 소개하는 사용 AWS Tools for PowerShell</a> 섹션에 정보가 추가되었습니다. PowerShell AWS Lambda	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](#)

사용자가 원하는 [cmdlet을 더 쉽게 찾을 수 있도록 도와주는 새 Get-AWSCmdletName cmdlet에 대한 정보를 Cmdlet의 검색 및 별칭 섹션에 추가했습니다.](#) AWS

2015년 2월 5일

## [AWS Tools for Windows PowerShell 1.1.1.0](#)

2013년 5월 15일

cmdlet의 컬렉션 출력은 항상 파이프라인에 열거됩니다. PowerShell 페이징 가능한 서비스 호출에 대한 자동 지원 새 \$ AWSHistory 셸 변수는 서비스 응답과 선택적으로 서비스 요청을 수집합니다. AWSRegion인스턴스는 파이프라이닝을 지원하는 대신 SystemName Region 필드를 사용합니다. Remove-S3Bucket- DeleteObjects 스위치 옵션을 지원합니다. AWSCredentialsSet-의 사용성 문제가 해결되었습니다. 자격 증명 및 지역 데이터를 획득한 곳에서 AWSDefaults 보고서를 초기화합니다. Stop-EC2InstanceAmazon.ec2.Model.예약 인스턴스를 입력으로 받아들입니다. 일반 목록<T> 파라미터 유형이 어레이 유형(T[])으로 대체됩니다. 리소스를 삭제하거나 종료하는 cmdlet에서 삭제 전에 확인 메시지를 표시합니다. Write-S3ObjectAmazon S3에 업로드할 인라인 텍스트 콘텐츠를 지원합니다.

## [AWS Tools for Windows PowerShell 1.0.1.0](#)

2012년 12월 21일

Windows PowerShell 버전 3을 사용하는 환경에서 자동 로딩을 활용할 수 있도록 Windows용 도구 PowerShell 모듈의 설치 위치가 변경되었습니다. 이제 모듈 및 지원 파일이 AWS ToolsPowerShell 아래의 AWSPowerShell 하위 폴더에 설치됩니다. AWS ToolsPowerShell 폴더에 있는 이전 버전의 파일이 설치 관리자에 의해 자동으로 제거됩니다. 이번 릴리스에서는 PSModulePath Windows용 PowerShell (모든 버전) 이 모듈의 상위 폴더 (AWS ToolsPowerShell)를 포함하도록 업데이트되었습니다. Windows PowerShell 버전 2가 설치된 시스템의 경우 새 위치에서 모듈을 가져온 다음 실행하도록 시작 메뉴 바로 가기가 업데이트됩니다. Initialize-AWSDefaults . Windows PowerShell 버전 3이 설치된 시스템의 경우 시작 메뉴 바로 가기가 업데이트되어 Import-Module 명령이 제거되고 나머지는 Initialize-AWSDefaults 남습니다. AWSPowerShell.psd1 파일 Import-Module 중 하나를 수행하도록 PowerShell 프로필을 편집한 경우 파일의 새 위치를 가리키도록 업데이트해야 합니다.

다. 또는 PowerShell 버전 3을 사용하는 경우 더 이상 필요하지 않으므로 `Import-Module` 명령문을 제거해야 합니다. 이러한 변경으로 인해 이제 Windows용 도구 PowerShell 모듈이 실행 `Get-Module -ListAvailable` 시 사용 가능한 모듈로 나열됩니다. 또한 Windows PowerShell 버전 3 사용자의 경우 모듈에서 내보낸 cmdlet을 실행하면 모듈을 먼저 사용할 필요 없이 현재 PowerShell 셸에 자동으로 로드됩니다. `Import-Module` 따라서 실행 정책에 스크립트 실행이 허용되지 않는 시스템에서 cmdlet을 대화형으로 사용할 수 있습니다.

[AWS Tools for Windows PowerShell 1.0.0.0](#)

최초 릴리스

2012년 12월 6일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.