사용자 가이드

AWS Tools for PowerShell

AWS Tools for PowerShell: 사용자 가이드

Table of Contents

AWS Tools for PowerShell이란 무엇입니까?

AWS Tools for PowerShell은 AWS SDK for .NET에서 공개하는 기능을 기반으로 하는 PowerShell 모듈 세트입니다. AWS Tools for PowerShell을 사용하면 PowerShell 명령줄에서 AWS 리소스에 대한 작업을 스크립팅할 수 있습니다.

cmdlet은 다양한 AWS 서비스 HTTP 쿼리 API를 사용하여 구현되는 경우에도 매개 변수를 지정하고 결과를 처리할 수 있도록 관용적인 PowerShell 환경을 제공합니다. 예를 들어, AWS Tools for PowerShell에 대한 cmdlet은 PowerShell 파이프라인을 지원합니다. 즉, cmdlet 안팎으로 PowerShell 개체를 파이프할 수 있습니다.

AWS Tools for PowerShell은 AWS Identity and Access Management(IAM) 인프라에 대한 지원을 포함하여 자격 증명을 처리하는 방법에 있어 유연성을 가지고 있습니다. IAM 사용자 자격 증명, 임시 보안 토큰 및 IAM 역할과 함께 도구를 사용할 수 있습니다.

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

> **Note**
>
> AWS Tools for PowerShell 버전 4는 최신 주 릴리스이며 AWS Tools for PowerShell 3.3 이전 버전과 호환되는 업데이트입니다. 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 PowerShell은 다음과 같은 세 가지 패키지로 제공됩니다.

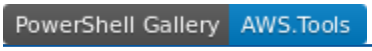
- `AWS.Tools`
- AWSPowerShell.NetCore
- AWSPowerShell

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 안내서에서 다음 내용을 참조하세요.

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

`AWS.Tools` - AWS Tools for PowerShell의 모듈화된 버전



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

이 버전은 AWS Tools for PowerShell의 최신 버전으로, Windows, Linux 및 macOS를 포함하여 지원되는 모든 운영 체제에서 실행됩니다. 이 패키지는 하나의 설치 모듈, `AWS.Tools.Installer`, 하나의 공통 모듈, `AWS.Tools.Common` 및 각 AWS 서비스에 대한 하나의 모듈을 제공합니다(예: `AWS.Tools.EC2`, `AWS.Tools.IAM`. `AWS.Tools.S3` 등).

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

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

이 버전의 AWS Tools for PowerShell은 다음을 실행 중인 컴퓨터에 설치할 수 있습니다.

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

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

AWSPowerShell.NetCore - AWS Tools for PowerShell의 단일 대형 모듈 버전

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

이 버전의 AWS Tools for PowerShell은 다음을 실행 중인 컴퓨터에 설치할 수 있습니다.

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

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

AWSPowerShell - Windows PowerShell의 단일 모듈 버전

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

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

이 설명서의 사용법

이 안내서는 다음과 같은 주요 단원으로 구성되어 있습니다.

AWS Tools for PowerShell 설치

이 단원에서는 AWS Tools for PowerShell를 설치하는 방법을 설명합니다. 여기에는 아직 계정이 없는 경우 AWS에 가입하는 방법과 cmdlet 실행에 사용할 수 있는 IAM 사용자를 생성하는 방법이 포함되어 있습니다.

AWS Tools for Windows PowerShell 시작

이 단원에서는 자격 증명 및 AWS 리전 지정, 특정 서비스용 cmdlet 찾기, cmdlet 별칭 사용 등 AWS Tools for PowerShell 사용에 관한 기초 사항을 설명합니다.

AWS Tools for PowerShell에서 AWS 서비스 작업

이 단원에는 AWS Tools for PowerShell를 사용해 가장 일반적인 몇몇 AWS 작업을 수행하는 방법에 대한 정보가 포함되어 있습니다.

AWS Tools for PowerShell 설치

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

주제

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.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 폴더 중 하나에 압축을 해제합니다. `PSModulePath` 환경 변수의 값을 표시하여 모듈 폴더를 찾을 수 있습니다.
- `Install-Module`cmdlet을 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.Common`도 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 `PSRepository`인 `PSGallery`(https://www.powershellgallery.com/)에서 요청된 모든 모듈을 다운로드하고 신뢰할 수 있는 소스로 간주합니다. `Get-PSRepository -Name PSGallery` 명령을 사용하여 이 `PSRepository`에 대해 자세히 알아볼 수 있습니다.

이전 명령을 실행하면 기본적으로 `%USERPROFILE%\Documents\WindowsPowerShell\Modules` 폴더에 모듈이 설치됩니다. 컴퓨터의 모든 AWS Tools for PowerShell 사용자용으로 를 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.zip을 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 PowerShellGet실행해야 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 PowerShellGet수 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 소개를 참조하십시오.

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

스크립트 실행을 활성화하려면

1. 실행 정책을 설정하려면 관리자 권한이 필요합니다. 관리자 권한이 있는 사용자로 로그인하지 않은 경우 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 PS를 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               ALXB        AWS.Tools.AlexaForBusiness
 3.7.0.11
Amplify Backend                  AMPB        AWS.Tools.AmplifyBackend
 3.7.0.11
```

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

실행 PowerShell 중인 버전을 확인하려면 $PS VersionTable 자동 변수의 내용을 `$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.Tools모듈화된 모듈을 업데이트하십시오.

AWS.Tools모듈을 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.Tools또는 둘 중 하나를 설치하십시오 AWSPowerShell. NetCore.
4. 적절한 Import-Module cmdlet을 실행하여 모듈을 세션으로 PowerShell 가져옵니다.
5. Initialize- AWSDefaultConfiguration cmdlet을 실행하여 자격 증명을 제공하십시오. AWS

필수 조건

를 AWS Tools for PowerShell Core실행하려면 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.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.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `$Env:PSModulePath` 변수의 값을 인쇄하여 모듈 디렉터리를 찾을 수 있습니다.
- `Install-Module`cmdlet을 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 큽니다. ~/.local/share/powershell/Modules 자세한 내용은 Microsoft PowerShell 웹 PowerShell 사이트의 Linux에 설치를 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.zip에서 모듈을 다운로드하고 모듈 디렉터리 중 하나에 압축을 해제합니다. `$Env:PSModulePath` 변수의 값을 인쇄하여 모듈 디렉터리를 찾을 수 있습니다.
- 다음 절차에 설명된 대로 `Install-Module` cmdlet을 사용하여 PowerShell 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 를 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 소개를 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 PowerShell`Initialize-AWSDefaultConfiguration`에 대한 자세한 정보는을 참조하세요AWS 자격 증명 사용.

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

버전 관리

AWS 새 서비스 및 기능을 지원하기 위해 AWS Tools for PowerShell 정기적으로 새 버전을 릴리스합니다. AWS 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 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
```

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-module`을 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.Filter`에 대한 참조가 있습니다. 이 유형의 참조는 자동 가져오기를 트리거할 수 없으므로 우선 `Import-Module`을 호출해야 합니다. 그렇지 않으면 명령이 실패합니다.

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

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

새로운 **Get-AWSService** cmdlet

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 cmdlet은 Amazon S3 SDK 작업을 호출합니다. ListObjects ListObjectsResponse이 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 항목에서 `-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 파이핑하고 `-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 PowerShell는 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 에서는 -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 검증을 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 설치한 후 Tools for Windows PowerShell 사용의 기본 사항에 대해 설명합니다. 예를 들면 Tools for Windows PowerShell에서 AWS와 상호 작용할 때 사용해야 할 보안 인증 정보와 AWS 리전을 지정하는 방법을 설명합니다.

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

주제

- 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 액세스를 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 Center먼저 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 를 사용하여 이러한 작업을 수행할 수 있습니다. AWS CLI방법을 알아보려면 을 참조하십시오. 포털 AWS CLI 로그인에 사용

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 설정을 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 경우 의 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 프로파일이 들어 있습니다. `my-sso-profile` PowerShell 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 공유 파일의 위치를 참조하세요.

Tools for PowerShell 에서는 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 획득합니다. AWS`sso_role_name`값은 IAM Identity Center 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용자에 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-STSCallerIdentity`cmdlet에 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. `-Region`cmdlet 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 사례를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 임시 보안 자격 증명을 참조하십시오.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 표준화된 보안 인증 공급자를 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`cmdlet을 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 CLI애플리케이션을 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 위치를 참조하세요. Tools for PowerShell 에서는 요청을 보내기 전에 프로필의 SSO 토큰 공급자를 사용하여 자격 증명을 획득합니다. AWS`sso_role_name`값은 IAM Identity Center 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용자에 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.SSOOIDC`

이전 버전의 Tools를 사용하고 있는데 이러한 모듈이 없으면 다음과 비슷한 오류가 발생합니다. “ AWSSDKAssembly .SSOOIDC를 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.

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 메시지에는 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 SSO, SSOOIDC

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

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

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

위에서 설명한 것처럼 [default] 프로필을 사용하고 있으므로 옵션을 사용하여 Get-S3Bucket cmdlet을 호출할 필요가 없습니다. -ProfileName SSO 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. `-Region`cmdlet 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 사례를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 임시 보안 자격 증명을 참조하십시오.
- 다른 보안 인증 공급자에 대해 알아보려면 AWS SDK 및 도구 참조 가이드에서 표준화된 보안 인증 공급자를 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 및 용 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-AWSRegion`을 통해 열거되지 않는 리전을 포함하여 지원되는 다른 AWS 리전을 모두 사용할 수 있습니다.

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

추가 정보

AWS 지역에 대한 추가 정보는 AWS SDK 및 도구 참조 안내서의AWS 지역을 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 Specifications를 참조하십시오. AWS Tools for PowerShell의 SAML 지원은 SAML 2.0과 호환됩니다.

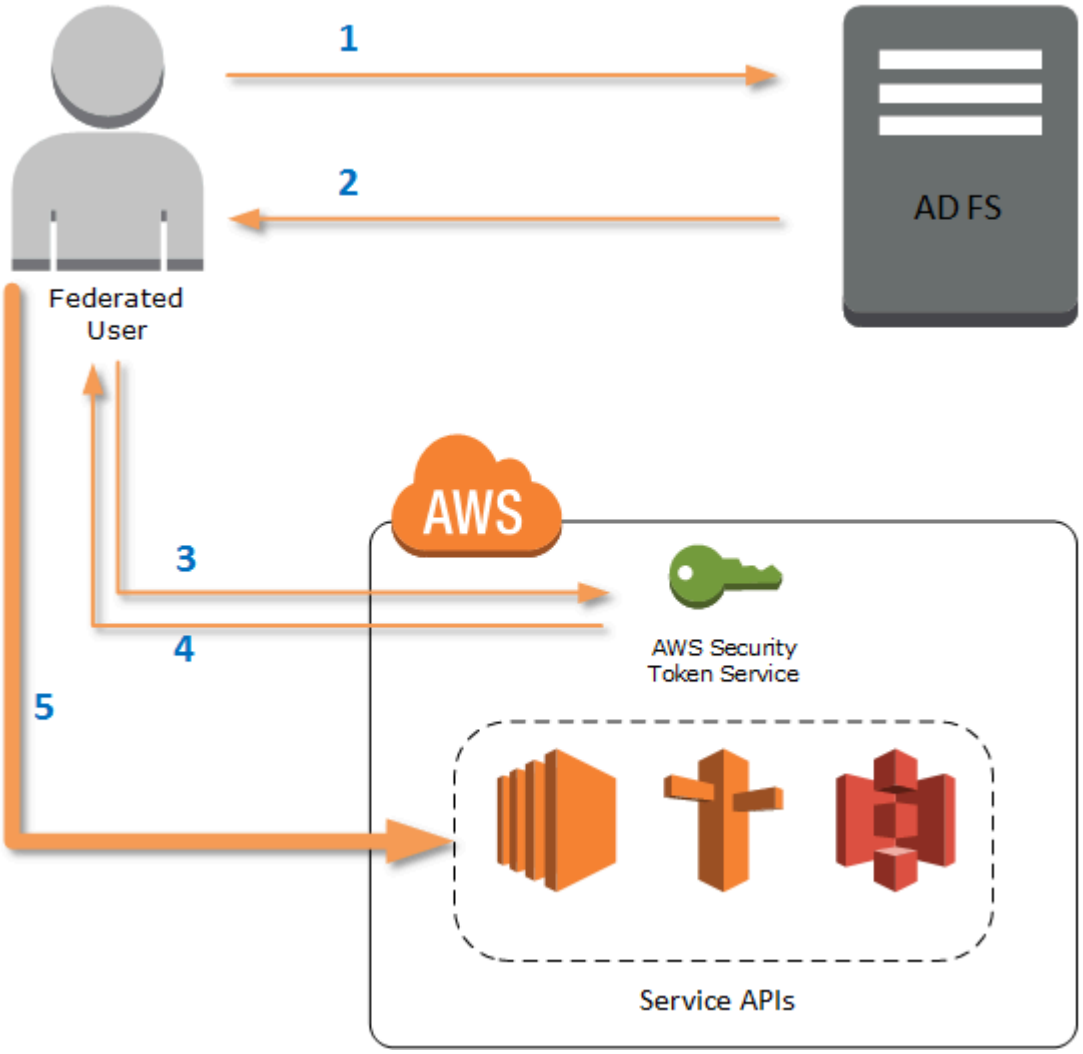
필수 조건

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 활성화를 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 PowerShell가 로컬 워크스테이션에 설치되어 있어야 합니다.

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

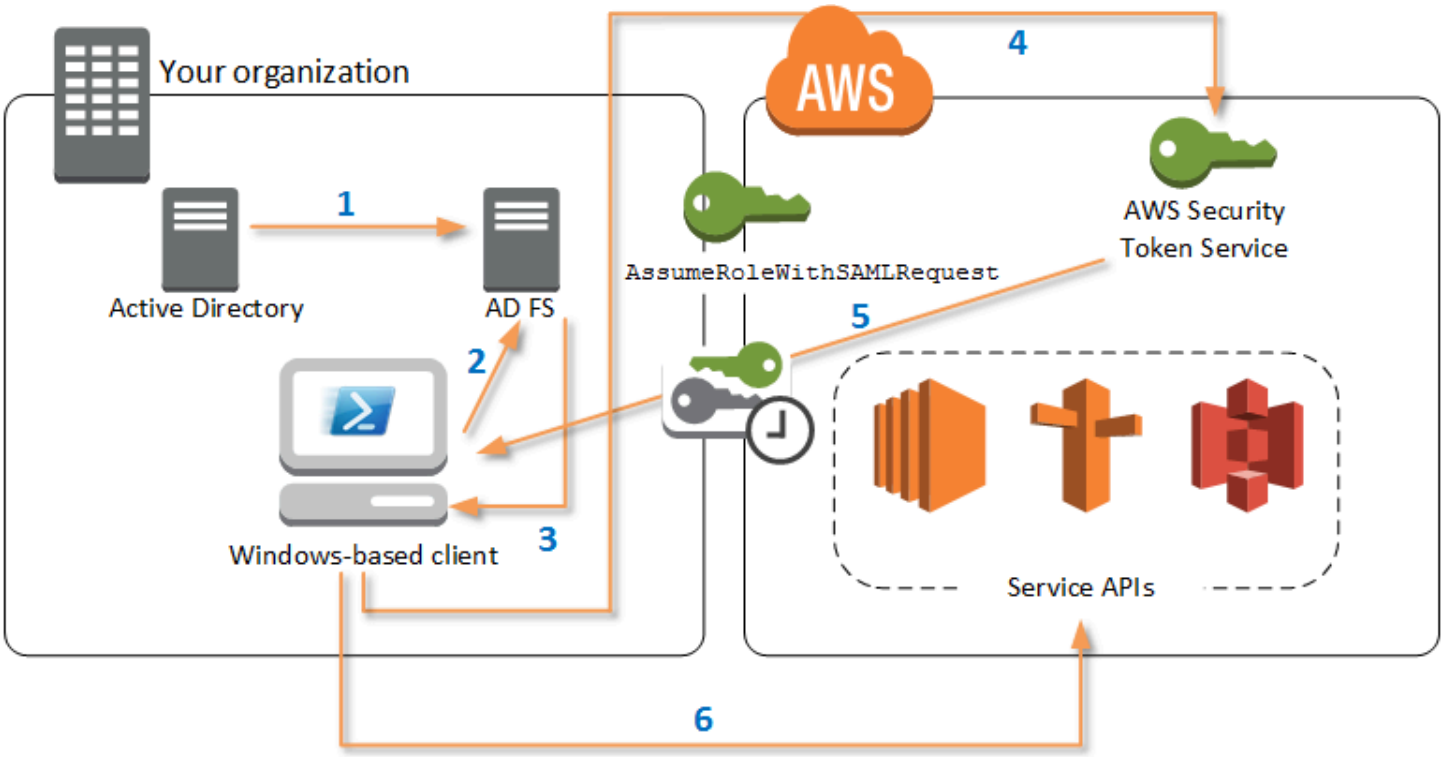
다음 프로세스에서는 Active Directory(AD) 사용자가 AD FS에 의해 연동되어 AWS 리소스에 대한 액세스 권한을 얻는 방법을 개략적으로 설명합니다.



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 수임할 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 사용자에 대한 SAML 기반 자격 증명 연동의 구성을 활성화하는 방법을 설명합니다.



1. AWS Tools for PowerShell은 Windows 사용자의 현재 자격 증명을 사용하여, 또는 (사용자가 AWS로의 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 사용자에 대한 정보를 식별하고 제공하는 것입니다. AWS Tools for PowerShell는 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 `SessionToken`을 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 증명이 필요하면 AWS Tools for PowerShell은 AD FS에서 자동으로 재인증을 수행하여 이후 1시간 동안 유효한 새 자격 증명을 획득합니다. 도메인에 조인된 계정의 사용자의 경우, 이 프로세스가 자동으로 수행됩니다. 도메인에 조인되지 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"
```

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

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

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

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

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

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

   또는 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-AWSCredential에 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-AWSCredential**을 사용하여 기본 역할 지정

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

```
PS > Set-AWSCredential -ProfileName SAMLDemoProfile
PS > Get-EC2Instance -Region us-west-2 | Format-Table -Property Instances,GroupNames

Instances                                                       GroupNames
---------                                                       ----------
{TestInstance1}                                                 {default}
{TestInstance2}                                                 {}
{TestInstance3}                                                 {launch-wizard-6}
{TestInstance4}                                                 {default}
{TestInstance5}                                                 {}
{TestInstance6}                                                 {AWS-OpsWorks-Default-
Server}
```

예제 2: PowerShell 세션 중 역할 프로파일 변경

이 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 조기에 역할 프로파일을 설정했으며 Get-S3Bucket cmdlet에 대해 다른 역할 프로파일을 사용하려는 경우에 유용

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

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

도메인에 조인된 사용자의 경우, 인증 시 현재 사용자의 Windows 자격 증명이 사용되므로 이 프로세스가 중단 없이 발생합니다. 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 2.0을 참조하십시오.

질문이나 의견이 있는 경우에는 PowerShell 스크립팅 또는 .NET 개발을 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 참조를 참조하세요. 대부분의 경우 AWS 서비스 API와 AWS PowerShell cmdlet 간에는 일대일 통신이 있습니다. AWS 서비스 API 이름에 해당하는 cmdlet 이름을 가져오려면 `-ApiOperation` 파라미터 및 AWS 서비스 API 이름을 지정하여 AWS `Get-AWSCmdletName` cmdlet을 실행합니다. 예를 들어 사용 가능한 `DescribeInstances` AWS 서비스 API를 기반으로 하는 모든 cmdlet 이름을 가져오려면 다음 명령을 실행합니다.

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances

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

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

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

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

```
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service EC2
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service Compute
PS > Get-AWSCmdletName -ApiOperation DescribeInstances -Service "Compute Cloud"

CmdletName        ServiceOperation    ServiceName                    CmdletNounPrefix
----------        ----------------    -----------                    ----------------
```

```
Get-EC2Instance   DescribeInstances   Amazon Elastic Compute Cloud   EC2
```

이러한 명령의 파라미터 값은 대/소문자를 구분합니다.

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

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

CmdletName                                  ServiceOperation
   ServiceName                         CmdletNounPrefix
----------                                  ----------------
   -----------                         ----------------
Approve-ECCacheSecurityGroupIngress         AuthorizeCacheSecurityGroupIngress
   Amazon ElastiCache                  EC
Get-ECCacheSecurityGroup                    DescribeCacheSecurityGroups
   Amazon ElastiCache                  EC
New-ECCacheSecurityGroup                    CreateCacheSecurityGroup
   Amazon ElastiCache                  EC
Remove-ECCacheSecurityGroup                 DeleteCacheSecurityGroup
   Amazon ElastiCache                  EC
Revoke-ECCacheSecurityGroupIngress          RevokeCacheSecurityGroupIngress
   Amazon ElastiCache                  EC
Add-EC2SecurityGroupToClientVpnTargetNetwrk
 ApplySecurityGroupsToClientVpnTargetNetwork Amazon Elastic Compute Cloud       EC2
Get-EC2SecurityGroup                        DescribeSecurityGroups
   Amazon Elastic Compute Cloud       EC2
Get-EC2SecurityGroupReference               DescribeSecurityGroupReferences
   Amazon Elastic Compute Cloud       EC2
Get-EC2StaleSecurityGroup                   DescribeStaleSecurityGroups
   Amazon Elastic Compute Cloud       EC2
Grant-EC2SecurityGroupEgress                AuthorizeSecurityGroupEgress
   Amazon Elastic Compute Cloud       EC2
Grant-EC2SecurityGroupIngress               AuthorizeSecurityGroupIngress
   Amazon Elastic Compute Cloud       EC2
New-EC2SecurityGroup                        CreateSecurityGroup
   Amazon Elastic Compute Cloud       EC2
Remove-EC2SecurityGroup                     DeleteSecurityGroup
   Amazon Elastic Compute Cloud       EC2
Revoke-EC2SecurityGroupEgress               RevokeSecurityGroupEgress
   Amazon Elastic Compute Cloud       EC2
```

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

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

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

```
CmdletName                                      ServiceOperation
   ServiceName                  CmdletNounPrefix
----------                                      ----------------
   -----------                  ----------------
Add-EC2SecurityGroupToClientVpnTargetNetwrk
 ApplySecurityGroupsToClientVpnTargetNetwork Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroup                            DescribeSecurityGroups
   Amazon Elastic Compute Cloud EC2
Get-EC2SecurityGroupReference                   DescribeSecurityGroupReferences
   Amazon Elastic Compute Cloud EC2
Get-EC2StaleSecurityGroup                       DescribeStaleSecurityGroups
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupEgress                    AuthorizeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Grant-EC2SecurityGroupIngress                   AuthorizeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
New-EC2SecurityGroup                            CreateSecurityGroup
   Amazon Elastic Compute Cloud EC2
Remove-EC2SecurityGroup                         DeleteSecurityGroup
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupEgress                   RevokeSecurityGroupEgress
   Amazon Elastic Compute Cloud EC2
Revoke-EC2SecurityGroupIngress                  RevokeSecurityGroupIngress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleEgressDescription  UpdateSecurityGroupRuleDescriptionsEgress
   Amazon Elastic Compute Cloud EC2
Update-EC2SecurityGroupRuleIngressDescription
 UpdateSecurityGroupRuleDescriptionsIngress  Amazon Elastic Compute Cloud EC2
```

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

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

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

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

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

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

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

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

그와 달리 서비스별로 결과를 정렬하려면 다음 명령을 실행합니다.

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

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

```
PS > Get-Command -Module AWSPowerShell.NetCore | Select-String region

Clear-DefaultAWSRegion
Copy-HSM2BackupToRegion
Get-AWSRegion
Get-DefaultAWSRegion
Get-EC2Region
Get-LSRegionList
Get-RDSSourceRegion
Set-DefaultAWSRegion
```

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

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

CommandType     Name                                               Version    Source
```

```
-----------     ----                                            -------    ------
Cmdlet          Add-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWEEventSource                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Disable-CWERule                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWEEventSource                           3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Enable-CWERule                                  3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBus                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventBusList                             3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSource                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEEventSourceList                          3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceAccountList            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEPartnerEventSourceList                   3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleDetail                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWERuleNamesByTarget                        3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Get-CWETargetsByRule                            3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEEventBus                                 3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          New-CWEPartnerEventSource                       3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEEventBus                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEPartnerEventSource                    3.3.563.1
 AWSPowerShell.NetCore
```

```
Cmdlet          Remove-CWEPermission                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWEResourceTag                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWERule                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Remove-CWETarget                                   3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Test-CWEEventPattern                               3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEEvent                                     3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPartnerEvent                              3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWEPermission                                3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWERule                                      3.3.563.1
 AWSPowerShell.NetCore
Cmdlet          Write-CWETarget                                    3.3.563.1
 AWSPowerShell.NetCore
```

Cmdlet 이름 지정 및 별칭

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

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

```
PS > Get-S3Object -BucketName text-content -Key aws-tech-docs

ETag         : "df000002a0fe0000f3c000004EXAMPLE"
BucketName   : aws-tech-docs
Key          : javascript/frameset.js
LastModified : 6/13/2011 1:24:18 PM
Owner        : Amazon.S3.Model.Owner
Size         : 512
StorageClass : STANDARD
```

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

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

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

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

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

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

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

```
Alias           Get-CFNAccountLimits                                3.3.343.0
 AWSPowerShell
Alias           Get-CFNStackEvents                                  3.3.343.0
 AWSPowerShell

...
```

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

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

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

```
PS > $MaximumAliasCount
32768
```

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

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

서비스 역할

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

서비스 역할을 만드는 프로세스는 상황에 따라 다르지만 기본적으로 다음과 같습니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 역할을 선택한 다음 역할 생성을 선택합니다.
3. AWS 서비스를 선택하고, EC2(예: EC2)를 찾아 선택한 다음 EC2 사용 사례(예: EC2)를 선택합니다.
4. 다음을 선택하고 애플리케이션에서 사용할 AWS 서비스에 대한 적절한 정책을 선택합니다.

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

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

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

6. 역할 생성을 선택합니다.

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

레거시 보안 인증 사용

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

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 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 방법을 참조하세요.
> 모범 보안 사례는 도구 인증 구성에 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 설명서의 보안 모범 사례 및 사용 사례를 참조하세요. 해당 설명과 더불어 다음 사항을 고려하세요.

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

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

- Amazon EC2 인스턴스에서 실행 중인 애플리케이션에 IAM 역할을 사용하세요.

주제

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 Center과 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 방법을 참조하세요.
> 모범 보안 사례는 도구 인증 구성에 설명된 대로 AWS IAM Identity Center를 사용하세요.

자격 증명이 노출되지 않도록 하기 위해 명령에 리터럴 자격 증명을 추가하지 않는 것이 좋습니다. 대신에, 사용할 각 자격 증명 세트에 대한 프로파일을 만들고 두 자격 증명 저장소 중 하나에 프로파일을 저장합니다. 명령에서 이름을 기준으로 올바른 프로파일을 지정하면 AWS Tools for 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 사례를 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

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 Studio에서도 AWS SDK 저장소를 사용할 수 있습니다.

- 홈 폴더에 있지만 자격 증명 파일을 일반 텍스트로 저장하는 공유 자격 증명 파일.

  기본적으로 자격 증명 파일은 다음에 저장됩니다.

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

  AWS 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 Studio를 사용하거나, AWS SDK for .NET을(를) 사용하여 프로그래밍 방식으로 AWS SDK 저장소를 관리할 수도 있습니다. 자격 증명 파일의 프로파일을 관리하는 방법에 대한 자세한 내용은 AWS 액세스 키 관리 모범 사례를 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/bPxRfiCYEXAMPLEKEY `
                -StoreAs MyNewProfile
```

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

프로파일 업데이트

AWS SDK 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-AWSCredential은 이전 버전과의 호환성을 위해 계속 제공되지만, Remove-AWSCredentialProfile가 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 순서에 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 `default`라는 이름의 프로파일이 있으면 `Initialize-AWSDefaultConfiguration` cmdlet이나 `Set-AWSCredential` cmdlet을 실행할 필요가 없습니다. 이들 도구는 해당 프로파일에 저장된 액세스 및 보안 키 데이터를 자동으로 사용합니다. 기본 리전(`Get-DefaultAWSRegion`의 결과) 이외의 AWS 리전을 사용하려면 `Set-DefaultAWSRegion`을 실행하고 리전을 지정합니다.

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

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

인스턴스 프로파일 없이 출시된 Amazon EC2 인스턴스의 PowerShell 세션을 실행하는 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-AWSDefaultConfiguration`를 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-AWSCredential`을 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).
   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.

AWS Tools for PowerShell Core의 자격 증명 처리

AWS Tools for PowerShell Core의 cmdlet은 AWS과 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 구성을 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\<userid>\.aws\credentials`입니다. Windows 이외의 플랫폼에서는 이 파일이 `~/.aws/credentials`에 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 보여줍니다.

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

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

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

Set-AWSCredential -ProfileName myProfileName

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

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

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

```
# 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 Center과 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 방법을 참조하세요.
모범 보안 사례는 도구 인증 구성에 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-AWSCredential에 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 수임할 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 수임할 때 선택적으로 멀티 팩터 인증(MFA) 디바이스 또는 `ExternalId` 코드를 사용하도록 요구할 수 있습니다.

파라미터 이름	설명
ExternalId	역할에 필요할 경우 역할 수임 시 사용될 사용자 정의 외부 ID입니다. 이는 일반적으로 계정에 대한 액세스 권한을 제3자에게 위임할 때만 필요합니다. 제3자는 할당된 역할을 위임할 때 외부 ID를 파라미터로 포함해야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리소스에 대한 액세스 권한을 서드 파티에 부여할 때 외부 ID를 사용하는 방법을 참조하세요.
MfaSerial	역할에 필요할 경우 역할 수임 시 사용될 MFA 일련 번호입니다. 자세한 내용은 IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.
RoleArn	역할 자격 증명 수임을 위해 수임할 역할의 ARN입니다. IAM 역할 생성 및 사용에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.
SourceProfile	역할 자격 증명 수임에 사용될 소스 프로파일의 이름입니다. 이 프로파일에서 확인된 자격 증명

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

역할 수임을 위한 프로파일 설정

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

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

```
PS > Set-AWSCredential -StoreAs my_source_profile -AccessKey access_key_id -
SecretKey secret_key
PS > Set-AWSCredential -StoreAs my_role_profile -SourceProfile my_source_profile -
RoleArn arn:aws:iam::123456789012:role/role-i-want-to-assume
PS > Get-AWSCredential -ProfileName my_role_profile

SourceCredentials                  RoleArn
 RoleSessionName                          Options
-----------------                  -------
 ---------------                          -------
Amazon.Runtime.BasicAWSCredentials arn:aws:iam::123456789012:role/
role-i-want-to-assume aws-dotnet-sdk-session-636238288466144357
 Amazon.Runtime.AssumeRoleAWSCredentialsOptions
```

Tools for Windows PowerShell 서비스 cmdlet과 함께 이 역할 프로파일을 사용하려면 -ProfileName 공통 파라미터를 명령에 추가하여 역할 프로파일을 참조합니다. 다음 예제에서는 이전 예제에 정의된 역할 프로파일을 사용하여 Get-S3Bucket cmdlet에 액세스합니다. AWS Tools for PowerShell는 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
 	
이들은 IAM 사용자에 대한 장기 자격 증명입니다.
	`-AccessKey`
	
	`-SecretKey`

세션:
 	
이들은 Use-STSRole cmdlet의 직접 호출 등을 통해 수동으로 검색하는 IAM 역할에 대한 단기 자격 증명입니다..
	`-AccessKey`
	
	`-SecretKey`
	
	`-SessionToken`

역할:
 	
이들은 사용자를 위해 AWS Tools for PowerShell이 검색하는 IAM 역할에 대한 단기 자격 증명입니다.
	`-SourceProfile`
	
	`-RoleArn`
	
	선택 사항: `-ExternalId`
	
	선택 사항: `-MfaSerial`

`ProfilesLocation` 공통 파라미터

`-ProfileLocation`을 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 *(user's home*

프로파일 위치 값	프로파일 해결 동작
	directory) \.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\auser\.aws\credentials -ProfileName
 basic_profile -AccessKey access_key2 -SecretKey secret_key2
PS > Get-Content C:\Users\auser\.aws\credentials

aws_access_key_id=access_key2
aws_secret_access_key=secret_key2
```

자격 증명 프로파일 표시

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

```
PS > Get-AWSCredential -ListProfileDetail

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

자격 증명 프로파일 제거

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

중요 정보

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

AWS Tools for 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 Reference를 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-R53ResourceRecordSet`이 바로 그 예입니다.

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

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

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

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

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

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

PowerShell 도구에 대해 반환된 객체

기본 PowerShell 환경에서 AWS Tools for PowerShell의 유용성을 높이기 위해 AWS Tools for PowerShell cmdlet에서 반환되는 개체는 AWS SDK의 해당 API에서 일반적으로 반환되는 JSON 텍스트 개체가 아니라 .NET 개체입니다. 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 콘솔을 사용한 유사한 프로세스를 설명하는 정적 웹 사이트 호스팅에 대한 시작 안내서를 모델링하고 있습니다.

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

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

다음 사항도 참조하세요.

- 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-example`이라는 버킷을 생성합니다. 버킷의 이름은 모든 리전에서 고유해야 합니다. 이 예제에서는 `us-west-1` 리전에 버킷을 생성합니다.

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

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

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

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

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

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

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

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

참고 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 객체로서 업로드해야 합니다.

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

```
AmazonId2      : T7hlDOeLqA5Q2XfTe8j2q3SLoP3/5XwhUU3RyJBGHU/LnC+CIWLeGgP0MY24xAlI
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Content-Length, Date...}
Metadata       : {}
ResponseXml    :
```

참고 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. `-File`에 대한 `Write-S3Object` 파라미터는 로컬 파일 시스템의 파일 이름을 지정합니다. `-Key` 파라미터는 Amazon S3에서 해당 객체에 사용될 이름을 지정합니다.

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

```
PS > # Create the two files using here-strings and the Set-Content cmdlet
PS > $index_html = @"
>> <html>
>>   <body>
>>     <p>
>>       Hello, World!
>>     </p>
>>   </body>
>> </html>
>> "@
>>
PS > $index_html | Set-Content index.html
PS > $error_html = @"
>> <html>
>>   <body>
>>     <p>
>>       This is an error page.
>>     </p>
>>   </body>
>> </html>
```

```
>> "@
>>
>>$error_html | Set-Content error.html
>># Upload the files to Amazon S3 using a foreach loop
>>foreach ($f in "index.html", "error.html") {
>> Write-S3Object -BucketName website-example -File $f -Key $f -CannedACLName public-
read
>> }
>>
PS > # Verify that the files were uploaded
PS > Get-S3BucketWebsite -BucketName website-example

IndexDocumentSuffix                                          ErrorDocument
-------------------                                          -------------
index.html                                                   error.html
```

미리 준비된 ACL 옵션

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

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

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

멀티파트 업로드에 관한 정보

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

웹 사이트 테스트

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

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

예:

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

참고 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` cmdlet을 실행하여 Amazon S3 버킷에서 HTML 파일에 대한 객체를 삭제합니다.

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

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

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

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

```
RequestId      : E480ED92A2EC703D
AmazonId2      : k6tqaqC1nMkoeYwbuJXUx1/UDa49BJd6dfLN0Ls1mWYNPHjbc8/Nyvm6AGbWcc2P
ResponseStream :
Headers        : {x-amz-id-2, x-amz-request-id, Date, Server}
Metadata       : {}
ResponseXml    :
```

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

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

참고 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 참조하세요AWS Tools for Windows PowerShell 시작

주제

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

키 페어 만들기

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

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

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

```
PS > $myPSKeyPair | Get-Member

    TypeName: Amazon.EC2.Model.KeyPair

  Name                MemberType   Definition
  ----                ----------   ----------
  Equals              Method       bool Equals(System.Object obj)
  GetHashCode         Method       int GetHashCode()
  GetType             Method       type GetType()
  ToString            Method       string ToString()
  KeyFingerprint      Property     System.String KeyFingerprint {get;set;}
  KeyMaterial         Property     System.String KeyMaterial {get;set;}
  KeyName             Property     System.String KeyName {get;set;}
```

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

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

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

```
                    -----END RSA PRIVATE KEY-----
```

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

키 페어 지문 보기

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

```
PS > Get-EC2KeyPair -KeyName myPSKeyPair | format-list KeyName, KeyFingerprint

  KeyName        : myPSKeyPair
  KeyFingerprint : 09:06:70:8e:26:b6:e7:ef:8f:fe:4a:1d:bc:9c:6a:63:11:ac:ad:3c
```

프라이빗 키 저장

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

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

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

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

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

키 페어 제거

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

```
PS > Remove-EC2KeyPair -KeyName myPSKeyPair

Confirm
```

```
Performing the operation "Remove-EC2KeyPair (DeleteKeyPair)" on target "myPSKeyPair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"):
```

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

Windows PowerShell을 사용하여 보안 그룹 생성

AWS Tools for PowerShell를 사용하여 보안 그룹을 생성 및 구성할 수 있습니다. 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)를 허용하도록 보안 그룹을 구성해야 합니다.

주제

- 사전 조건
- EC2-Classic에 대한 보안 그룹 생성
- EC2-VPC에 대한 보안 그룹 생성

사전 조건

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

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

EC2-Classic에 대한 보안 그룹 생성

> **Warning**
>
> EC2-Classic은 2022년 8월 15일에 사용 중지될 예정입니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서 또는 Windows 인스턴스용 Amazon EC2 사용 설명서의 EC2-Classic에서 VPC로 마이그레이션을 참조하세요. 또는 블로그 게시물EC2-Classic 네트워킹은 사용 중지 중입니다 - 준비 방법은 다음과 같습니다를 참조하세요.

다음 예제에서는 `New-EC2SecurityGroup` cmdlet을 사용하여 `EC2-Classic`에 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 `203.0.113.25/32`에서 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 개선 사항을 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, xvdcb, xvdcc…}
CreationDate        : 2019-08-16T09:36:09.000Z
Description         : Microsoft Windows Server 2016 Core Locale English AMI provided by
 Amazon
EnaSupport          : True
Hypervisor          : xen
ImageId             : ami-06f2a2afca06f15fc
ImageLocation       : amazon/Windows_Server-2016-English-Core-Base-2019.08.16
ImageOwnerAlias     : amazon
ImageType           : machine
KernelId            :
Name                : Windows_Server-2016-English-Core-Base-2019.08.16
OwnerId             : 801119661308
Platform            : Windows
ProductCodes        : {}
Public              : True
RamdiskId           :
RootDeviceName      : /dev/sda1
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {}
VirtualizationType  : hvm
```

Windows PowerShell을 사용하여 Amazon EC2 인스턴스 시작

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

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

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

주제

- 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 Linux 인스턴스용 Amazon EC2 사용 설명서 또는 Windows 인스턴스용 Amazon EC2 사용 설명서의 EC2-Classic에서 VPC로 마이그레이션을 참조하세요. 또는 블로그 게시물EC2-Classic 네트워킹은 사용 중지 중입니다 - 준비 방법은 다음과 같습니다를 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-EC2Instance을 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 표를 참조하세요.

필수 조건

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

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

AWSLambdaPSCore 모듈 설치

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

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

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

참고 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" :
 "llE17A2+XOuJZnw3TlgcXz4C4KPLXZxbxoEMIirelhl3u/oxkWmz5+9tJKFMns1ZOqQvKxk
+ExfEZcD5yWt6biVuBb8pyRmZ1bO3hUENl3ayv2WQiQT1vpLpM7VEQN5m+hLIiPFcs
                        vyuGkJReV7lOJWPHnCN
+qTE2lId2RPkFOeGtLGawTsSPTWEvJdDbLlf7E0zZ0q1niXTUtpsZ8Swx01X3QO6u9i9qBFt0ekJFZNJp6Avu05hIklb4yo
y0a8Yl9lWp7a7EoWaBn0zhCESe7o
```

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

AWS Tools for Windows 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 콘솔로 이동하여 이 작업의 결과를 볼 수 있습니다.

```
$dat = New-Object Amazon.CloudWatch.Model.MetricDatum
$dat.Timestamp = (Get-Date).ToUniversalTime()
$dat.MetricName = "New Posts"
$dat.Unit = "Count"
$dat.Value = ".50"
Write-CWMetricData -Namespace "Usage Metrics" -MetricData $dat
```

다음을 참조하세요.

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

참고 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)

이 AWS 제품 또는 서비스에 대한 보안

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객은 보안에 매우 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다. 보안은 AWS와 귀하의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드의 보안 – AWS는 AWS 클라우드에서 모든 서비스를 실행하는 인프라를 보호하며 안전하게 사용할 수 있는 서비스를 제공합니다. 당사의 보안 책임은 AWS에서 우선 순위가 가장 높으며, 서드 파티 감사자는 AWS 규정 준수 프로그램의 일환으로 정기적으로 보안 효율성을 테스트하고 검증합니다.

클라우드 내 보안 – 사용자의 책임은 사용하는 AWS 서비스, 그리고 데이터의 민감도, 조직의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 의해 결정됩니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 AWS 서비스를 참조하세요.

주제

이 AWS 제품 또는 서비스의 데이터 보호

AWS 공동 책임 모델은 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 Center또는 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) 140-2를 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 인증를 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

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 또는AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 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 서명을 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) 사용을 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 작업을 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 사용자는 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 교체를 참조하세요.

IAM 그룹은 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 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 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 사용을 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 차이를 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 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할: 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 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 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 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 개요를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는 지를 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 수임할 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 CLI또는 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 생성을 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 역

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 선택을 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 지정해야 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) 개요를 참조하세요.

기타 정책 유형

AWS은(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 경계를 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 사용자을(를) 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 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 정책을 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 로직을 참조하세요.

AWS 서비스에서 IAM을 사용하는 방식

AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 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 awes:*GetWidget* 권한이 없을 때 발생합니다.

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

이 경우 awes:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 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:PassRole작업을 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 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 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 차이를 참조하세요.

이 AWS 제품 또는 서비스에 대한 규정 준수 확인

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반적인 정보는 AWS규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 서드 파티 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드 섹션을 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS환경을 배포하기 위한 단계를 제공합니다.
- Architecting for HIPAA Security and Compliance on Amazon Web Services(Amazon Web Services에서 HIPAA 보안 및 규정 준수 기술 백서 설계) - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 만드는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스에 HIPAA 자격이 있는 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조 섹션을 참조하십시오.

- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 컬렉션입니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에서는 AWS 서비스를 보호하기 위한 모범 사례를 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 가이드의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub – 이 AWS 서비스는 AWS내의 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub는 보안 제어를 사용하여 AWS리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조 섹션을 참조하십시오.
- AWS Audit Manager - 이 AWS 서비스는 AWS사용을 지속해서 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있도록 지원합니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 AWS 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 서비스를 참조하세요.

PowerShell용 도구에서 최소 TLS 버전 적용

AWS 서비스와 통신할 때 보안을 강화하려면 적절한 TLS 버전을 사용하도록 PowerShell용 도구를 구성해야 합니다. 이 방법에 대한 자세한 내용은 AWS SDK for .NET 개발자 가이드에서 최소 TLS 버전 적용을 참조하세요.

Tools for PowerShell cmdlet 참조

Tools for PowerShell은 AWS 서비스에 액세스하는 데 사용할 수 있는 cmdlet을 제공합니다. 사용할 수 있는 cmdlet을 확인하려면 AWS Tools for PowerShell Cmdlet Reference를 참조하세요.

문서 기록

다음 주제에서는 AWS Tools for PowerShell설명서에서 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 PowerShell노트를 참조하십시오.

변경 사항	설명	날짜
도구 인증을 다음과 같이 구성하십시오. AWS	에 SSO 지원에 대한 정보가 추가되었습니다. AWS Tools for PowerShell	2024년 3월 15일
다음을 위한 도구에 대한 cmdlet 참조 PowerShell	PowerShell cmdlet용 도구 참조에 대한 링크가 포함된 섹션이 추가되었습니다.	2023년 11월 17일
IAM 모범 사례 업데이트 추가 포함	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오.	2023년 10월 12일
Windows에 설치	더 이상 사용되지 않는 MSI를 사용한 PowerShell Windows용 도구 설치에 대한 정보가 제거되었습니다.	2023년 9월 25일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오.	2023년 9월 8일
파이프라이닝 및 $ AWSHistory	`Set-AWSHistoryConfiguration` cmdlet에 `IncludeSensitiveData` 파라미터를 추가했습니다.	2023년 3월 9일

cmdlet의 ClientConfig 매개 변수 사용	매개 변수 지원에 대한 ClientConfig 정보가 추가되었습니다.	2022년 10월 28일
윈도우를 사용하여 Amazon EC2 인스턴스 시작 PowerShell	EC2-Classic에 대한 참고 사항이 추가되었습니다.	2022년 7월 26일
AWS Tools for PowerShell 버전 4	Windows 및 Linux/macOS에 대한 설치 지침을 포함한 버전 4에 대한 정보와 버전 3과의 차이점을 설명하고 새로운 기능을 소개하는 마이그레이션 주제가 추가되었습니다.	2019년 11월 21일
AWS Tools for PowerShell 3.3.563	`AWS.Tools.Common` 모듈의 평가판 버전을 설치하고 사용하는 방법에 대한 정보가 추가되었습니다. 이 새 모듈은 이전 모놀리식 패키지를 하나의 공유 모듈과 서비스당 하나의 모듈로 분리합니다. AWS	2019년 10월 18일
AWS Tools for PowerShell 3.3.343.0	PowerShell Core 개발자가 함수를 빌드할 수 있는 AWS Lambda 도구를 소개하는 사용 AWS Tools for PowerShell 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	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-S3Bucket은 - 스위치 옵션을 지원합니다. DeleteObjects Set-의 사용성 문제가 해결되었습니다. AWSCredentials 자격 증명 및 지역 데이터를 획득한 곳에서 AWSDefaults 보고서를 초기화합니다. Stop-EC2Instance는 입력으로 Amazon.EC2.Model.Reservation 인스턴스를 허용합니다. 일반 목록<T> 파라미터 유형이 어레이 유형(T[])으로 대체됩니다. 리소스를 삭제하거나 종료하는 cmdlet에서 삭제 전에 확인 메시지를 표시합니다. Write-S3Object는 Amazon S3에 업로드할 인라인 텍스트 콘텐츠를 지원합니다.	2013년 5월 15일

AWS Tools for Windows PowerShell 1.0.1.0	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 합니	2012년 12월 21일

	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.