AWS Command Line Interface

사용 설명서

AWS Command Line Interface: 사용 설명서

Table of Contents

AWS Command Line Interface이란 무엇입니까?

AWS CLI는 명령줄 셸의 명령을 사용하여 AWS 제품과 상호 작용할 수 있는 오픈 소스 도구입니다. 최소한의 구성으로, 원하는 터미널 프로그램에 있는 명령 프롬프트에서 브라우저 기반 AWS Management 콘솔이 제공하는 것과 동일한 기능을 사용할 수 있습니다.

- Linux 셸 – `bash`, `zsh` 및 `tsch` 등의 일반적인 셸 프로그램을 사용하여 Linux, macOS, or Unix에서 명령을 실행합니다.
- Windows 명령줄 – Microsoft Windows의 PowerShell 또는 Windows 명령 프롬프트에서 명령을 실행합니다.
- 원격 – PuTTY 또는 SSH 등의 원격 터미널을 통해 또는 Amazon EC2 시스템 관리자를 사용하여 Amazon EC2 인스턴스에서 명령을 실행하십시오.

AWS Management 콘솔의 모든 IaaS(서비스로서의 인프라) AWS 관리, 관리 및 액세스 함수는 AWS API 및 CLI에서 사용 가능합니다. 새 AWS IaaS 기능 및 서비스는 출시할 때 또는 출시 후 180일 이내에 API 및 CLI를 통해 전체 AWS Management 콘솔 기능을 제공합니다.

AWS CLI를 사용하면 AWS 서비스의 퍼블릭 API를 직접 액세스할 수 있습니다. AWS CLI를 사용하여 서비스의 기능을 살펴보고 리소스를 관리할 셸 스크립트를 개발할 수 있습니다. 또는 AWS SDK를 사용하여 프로그램을 다른 언어로 개발하기 위해 학습한 내용을 활용할 수 있습니다.

하위 수준 API와 상응한 명령 외에 여러 AWS 제품에서도 AWS CLI에 대한 사용자 지정 기능을 제공합니다. 사용자 지정에는 복잡한 API와 서비스의 사용을 간소화하는 상위 수준 명령이 포함될 수 있습니다. 예를 들어 `aws s3` 명령 세트는 Amazon S3에서 파일 관리에 익숙한 구문을 제공합니다.

Example Amazon S3로 파일 업로드

`aws s3 cp`는 셸 같은 복사 명령을 제공하고 멀티파트 업로드를 자동으로 수행하여 대용량 파일을 빠르고 탄력적으로 전송합니다.

```
~$ aws s3 cp myvideo.mp4 s3://mybucket/
```

하위 수준 명령(`aws s3api`에서 사용 가능)으로 같은 작업을 수행할 경우 더 많은 노력이 필요합니다.

사용 사례에 따라 AWS SDK, 도구 키트 또는 Windows PowerShell용 AWS 도구을 사용하고자 할 수 있습니다.

- Windows PowerShell용 AWS 도구
- AWS SDK for Java
- .NET용 AWS SDK

  AWS SDK for JavaScript
- Ruby용 AWS SDK
- AWS SDK for Python (Boto)
- PHP용 AWS SDK
- Go용 AWS SDK
- AWS Toolkit for Eclipse
- AWS Toolkit for Visual Studio

- AWS Mobile SDK for iOS
- Android용 AWS Mobile SDK

aws-cli 리포지토리의 GitHub에서 AWS CLI에 대한 소스 코드를 조회 및 분기할 수 있습니다. GitHub의 사용자 커뮤니티에 참여하여 피드백을 제공하고 기능을 요청하며 자체적으로 기여하십시오!

이 설명서의 예제 사용

이 설명서의 예제는 다음과 같은 규칙에 따라 서식 지정됩니다.

- 프롬프트 – 명령 프롬프트는 달러 기호와 공백('$ ')으로 표시됩니다. 명령을 입력할 때 프롬프트를 포함시키지 마십시오.
- 디렉터리 – 특정 디렉터리에서 명령을 실행해야 하는 경우 프롬프트 기호 앞에 디렉터리 이름이 표시됩니다.
- 사용자 입력 – 명령줄에 입력해야 하는 명령 텍스트는 **`user input`**으로 서식 지정됩니다.
- 대체 가능한 텍스트 – 선택하는 리소스의 이름 또는 명령에 포함시켜야 하는 AWS 서비스에서 생성된 ID를 포함한 변수 텍스트는 ***`## ### ###`***로 서식 지정됩니다. 특정 키보드 입력이 필요한 여러 줄 명령에서는 키보드 명령도 대체 가능한 텍스트로 표시될 수 있습니다.
- 출력 – AWS 제품에서 반환되는 출력은 사용자 입력 아래에 `computer output` 형식으로 표시됩니다.

예를 들어, 다음 명령에는 사용자 입력, 대체 가능한 텍스트 및 출력이 포함됩니다.

```
aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: ENTER
```

이 예제를 사용하려면 명령줄에 **`aws configure`**를 입력하고 ENTER를 누릅니다. **`aws configure`**는 명령입니다. 이 명령은 대화형이므로 AWS CLI는 추가 정보를 입력하라고 알리는 텍스트 줄을 출력합니다. 각 액세스 키를 차례로 입력한 다음 ENTER를 누릅니다. 그런 다음, 표시된 형식으로 리전 이름을 입력하고 ENTER를 누른 다음 마지막으로 ENTER를 눌러 출력 형식 설정을 건너뜁니다. 마지막 ENTER 명령은 해당 줄에 대한 사용자 입력이 없기 때문에 대체 가능한 텍스트로 표시됩니다. 그렇지 않으면 값이 내재되어 있습니다.

다음 예제에서는 JSON 형식의 간단한 비대화형 명령과 서비스의 출력을 보여줍니다.

```
aws ec2 create-security-group --group-name my-sg --description "My security group"
{
    "GroupId": "sg-903004f8"
}
```

이 예제를 사용하려면 명령의 전체 텍스트를 입력하고(프롬프트 다음에 강조 표시된 텍스트) ENTER를 누릅니다. 보안 그룹의 이름인 *`my-sg`*는 대체 가능합니다. 이 경우 표시된 그룹 이름을 사용할 수 있지만 더 설명적인 이름을 사용할 수 있습니다.

> **Note**
>
> 대체해야 하는 인수(예: AWS 액세스 키 ID)와 대체해야 할 항목(예: 그룹 이름)은 모두 *`replaceable text`*로 표시됩니다. 인수를 대체해야 하는 경우 예제를 설명하는 텍스트에 표시됩니다.

중괄호를 포함한 JSON 문서는 출력입니다. 텍스트 또는 테이블 형식으로 출력할 CLI를 구성하는 경우 출력이 다르게 서식 지정됩니다. 기본 출력 형식은 JSON입니다.

Amazon Web Services에 대하여

AWS(Amazon Web Services)는 애플리케이션을 개발할 때 개발자들이 활용할 수 있는 디지털 인프라 서비스의 컬렉션입니다. 이 서비스에는 컴퓨팅, 스토리지, 데이터베이스, 애플리케이션 동기화(메시징 및 대기열)가 있습니다. AWS는 "사용한 만큼 지불" 서비스 모델을 사용합니다. 사용자 또는 애플리케이션이 사용하는 서비스에 대해서만 청구됩니다. 또한 AWS를 프로토타입 생성 및 실험용 플랫폼으로 더욱 쉽게 이용할 수 있도록 프리 티어를 제공합니다. 이 계층에서 특정 사용 수준 미만의 서비스는 무료입니다. AWS 비용 및 프리 티어에 대해 자세히 알아보려면 프리 티어에서 AWS 테스트 드라이브를 참조하십시오. AWS 계정을 얻으려면 AWS 홈 페이지를 연 다음 [Sign Up]을 클릭하십시오.

AWS Command Line Interface 설치

AWS CLI 설치 방법

요구 사항

- Python 2 버전 2.6.5+ 또는 Python 3 버전 3.3+
- Windows, Linux, macOS, or Unix

> **Note**
>
> Python 구 버전은 일부 AWS 제품을 지원하지 않을 수 있습니다. &CLI;를 설치하거나 사용할 때 `InsecurePlatformWarning` 또는 운영 중단 공지가 표시되는 경우, 최신 버전으로 업데이트하십시오.

*pip*를 사용하여 AWS CLI 설치

Linux, Windows및 macOS에서 AWS CLI를 배포하는 기본 방법은 `pip`입니다. 이는 Python 패키지 및 해당 종속 항목을 쉽게 설치, 업그레이드 및 제거하는 방법을 제공하는 Python용 패키지 관리자입니다.

현재 AWS CLI 버전

AWS CLI는 새로운 서비스 및 명령에 대한 지원으로 자주 업데이트됩니다. 최신 버전을 사용하는지 확인하려면 GitHub의 릴리스 페이지를 참조하십시오.

`pip` 및 지원되는 버전의 Python이 이미 있는 경우 다음 명령을 사용하여 AWS CLI를 설치할 수 있습니다.

```
pip install awscli --upgrade --user
```

`--upgrade` 옵션은 `pip`에게 이미 설치된 요구 사항을 업그레이드하라고 지시합니다. `--user` 옵션은 운영 체제에서 사용한 라이브러리를 수정하지 않도록 사용자 디렉터리의 하위 디렉터리에 프로그램을 설치할 것을 `pip`에게 지시합니다.

가상 환경에 AWS CLI 설치

`pip`를 사용하여 AWS CLI를 설치할 때 문제가 발생할 경우 가상 환경에 AWS CLI를 설치 (p. 14)하여 도구 및 해당 종속 항목을 격리하거나 일반적으로 사용하는 것과 다른 버전의 Python을 사용할 수 있습니다.

설치 관리자를 사용하여 AWS CLI 설치

Linux, macOS, or Unix에서 오프라인 또는 자동 설치하는 경우 번들 설치 관리자 (p. 15)를 사용해 보십시오. 이 번들 설치 관리자에는 AWS CLI 및 해당 종속 항목과 대신 설치를 수행하는 셸 스크립트가 포함되어 있습니다.

Windows에서는 MSI 설치 관리자 (p. 9)를 사용할 수도 있습니다. 두 방법 모두 최초 설치는 간단하지만, 대신 새 버전의 AWS CLI가 릴리스되었을 경우 업그레이드는 더 어렵습니다.

설치 후 수행할 단계

AWS CLI를 설치한 후 실행 파일에 대한 경로를 PATH 변수에 추가해야 할 수 있습니다. 플랫폼별 지침은 다음 주제를 참조하십시오.

- Linux – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 7)
- Windows – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 11)
- macOS – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 13)

`aws --version`을 실행하여 AWS CLI가 올바르게 설치되었는지 확인하십시오.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/4.4.0-59-generic botocore/1.5.47
```

AWS CLI는 정기적으로 업데이트되어 새로운 서비스 및 명령에 대한 지원을 추가합니다. 최신 버전의 AWS CLI로 업데이트하려면 설치 명령을 다시 실행하십시오.

```
pip install awscli --upgrade --user
```

AWS CLI를 제거해야 하는 경우 `pip uninstall`을 사용하십시오.

```
pip uninstall awscli
```

Python 및 `pip`가 없는 경우 운영 체제에 해당하는 절차를 사용합니다.

각 환경에 대한 상세 지침

- Linux에 AWS Command Line Interface 설치 (p. 5)
- Microsoft Windows에 AWS Command Line Interface 설치 (p. 9)
- macOS에 AWS Command Line Interface을 설치하십시오. (p. 11)
- 가상 환경에 AWS Command Line Interface을 설치하십시오. (p. 14)
- 번들 설치 관리자(Linux, macOS, or Unix)를 사용하여 AWS CLI 설치 (p. 15)

Linux에 AWS Command Line Interface 설치

Python용 패키지 관리자인 `pip`를 사용하여 대부분의 Linux 배포에서 AWS Command Line Interface 및 해당 종속성을 설치할 수 있습니다.

> **Important**
>
> `awscli` 패키지는 APT 및 yum과 같은 다른 패키지 관리자용 리포지토리에서 사용할 수 있지만, `pip`에서 가져오거나 번들 설치 관리자 (p. 15)를 사용하지 않는 한 최신 버전이 보장되지 않습니다.

pip가 이미 있는 경우 기본 설치 주제 (p. 4)의 지침을 따릅니다. `pip --version`을 실행하여 해당 버전의 Linux에 Python과 pip가 이미 포함되어 있는지 확인합니다.

```
pip --version
```

pip가 없는 경우 설치되어 있는 Python 버전을 확인합니다.

```
python --version
```

또는

```
python3 --version
```

아직 Python 2 버전 2.6.5+ 또는 Python 3 버전 3.3+를 설치하지 않은 경우 Python을 설치 (p. 7)해야 합니다. 이미 Python이 설치되어 있는 경우 pip 및 AWS CLI 설치로 계속 진행합니다.

단원

pip 설치

`pip`가 아직 설치되지 않은 경우 Python Packaging Authority에서 제공하는 스크립트를 사용하여 설치할 수 있습니다.

pip를 설치하려면

1. `curl` 명령을 사용하여 설치 스크립트를 다운로드합니다.

   ```
   curl -O https://bootstrap.pypa.io/get-pip.py
   ```

2. 이 스크립트는 최신 버전의 `pip`와 `setuptools`라는 다른 필수 패키지를 다운로드하고 설치합니다. Python을 사용하여 스크립트를 실행합니다.

   ```
   python get-pip.py --user
   ```

3. 실행 경로 `~/.local/bin`을 PATH 변수에 추가합니다.

   a. 사용자 폴더에서 셸의 프로파일 스크립트를 찾습니다. 어떤 셸을 가지고 있는지 잘 모르는 경우 `echo $SHELL`을 실행합니다.

   ```
   ls -a ~
   .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
   ```

   - Bash – `.bash_profile`, `.profile` 또는 `.bash_login`.
   - Zsh – `.zshrc`
   - Tcsh – `.tcshrc`, `.cshrc` 또는 `.login`.

   b. 프로필 스트립트의 끝에 내보내기 명령을 추가합니다.

   ```
   export PATH=~/.local/bin:$PATH
   ```

   이 명령은 경로(이 예제에서 `~/.local/bin`)를 현재 PATH 변수에 추가합니다.

c. 현재 세션에 프로필을 다시 로드하여 해당 변경 사항을 적용합니다.

```
source ~/.bash_profile
```

4. 이제 pip가 올바르게 설치되었는지 확인하기 위해 테스트할 수 있습니다.

```
pip --version
pip  from ~/.local/lib/python3.7/site-packages (python 3.7)
```

pip를 사용하여 AWS CLI 설치

pip를 사용하여 AWS CLI를 설치하십시오.

```
pip install awscli --upgrade --user
```

AWS CLI가 올바르게 설치되었는지 확인하십시오.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/4.4.0-59-generic botocore/1.5.47
```

오류가 발생한 경우 AWS CLI 오류 문제 해결 (p. 91)를 참조하십시오.

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

```
$ pip install awscli --upgrade --user
```

명령줄 경로에 AWS CLI 실행 파일 추가

pip를 사용하여 설치한 후 aws 실행 파일을 OS의 PATH 환경 변수에 추가해야 할 수 있습니다.

Example AWS CLI 설치 위치 - pip 사용 Linux(사용자 모드)

```
~/.local/bin
```

사용자 모드에서 설치하지 않았다면 실행 파일이 Python의 bin 폴더에 있을 수 있습니다. Python 설치 위치를 모르는 경우, which python을 실행하십시오.

```
which python
/usr/local/bin/python
```

실제 실행 파일이 아니라 symlink 경로가 출력될 수 있습니다. ls -al을 실행하여 어디를 가리키는지 확인합니다.

```
$ ls -al /usr/local/bin/python
~/.local/Python/3.7/bin/python3.7
```

pip 설치 (p. 6)의 3단계에서 경로에 추가한 것과 동일한 폴더인 경우 완료된 것입니다. 그렇지 않으면 3a ~ 3c의 동일한 단계를 다시 수행하여 이 폴더를 경로에 추가합니다.

Linux에 Python 설치

배포가 Python과 함께 제공되지 않았거나 다른 버전과 함께 제공된 경우 pip와 AWS CLI를 설치하기 전에 Python을 설치하십시오.

Linux에 Python 3를 설치하려면

1. Python이 이미 설치되어 있는지 확인합니다.

```
python --version
```

   Note

   Linux 배포가 Python과 함께 제공된 경우 확장명을 컴파일하는 데 필요한 헤더와 라이브러리를 가져오기 위해 Python 개발자 패키지를 설치하고 AWS CLI를 설치해야 할 수 있습니다. 패키지 관리자를 사용하여 개발자 패키지(일반적으로 `python-dev` 또는 `python-devel`)를 설치하십시오.

2. Python 2.7 이상이 설치되어 있지 않은 경우 배포의 패키지 관리자를 사용하여 Python을 설치합니다. 다음과 같이 명령과 패키지 이름이 다릅니다.

   - Ubuntu와 같은 Debian 계열 시스템에는 `APT`를 사용합니다.

```
sudo apt-get install python3
```

   - Red Hat 및 계열 시스템에는 `yum`을 사용합니다.

```
sudo yum install python
```

   - SUSE 및 계열 시스템에는 `zypper`를 사용합니다.

```
sudo zypper install python3
```

3. 명령 프롬프트 또는 셸을 열고 다음 명령을 실행하여 Python이 올바르게 설치되었는지 확인합니다.

```
python3 --version
Python 3.6.2
```

Amazon Linux에 AWS Command Line Interface 설치

AWS CLI는 Amazon Linux 및 Amazon Linux 2에 사전 설치되어 있습니다. 다음 명령을 사용하여 현재 설치된 버전을 점검하십시오.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/ botocore/1.5.47
```

`sudo yum update`를 사용하여 yum 리포지토리에서 사용 가능한 최신 버전을 가져올 수 있지만, 이 버전은 최신 버전이 아닐 수 있습니다. 대신 `pip`를 사용하여 최신 버전을 가져오는 것이 좋습니다.

사전 조건

Python 및 pip가 이미 설치되어 있는지 확인하십시오. 자세한 내용은 Linux에 AWS Command Line Interface 설치 (p. 5) 항목을 참조하십시오.

Amazon Linux(루트)에 AWS CLI를 업그레이드하려면

1. `pip install`을 사용하여 최신 버전의 AWS CLI를 설치하십시오.

```
sudo pip install --upgrade awscli
```

2. `aws --version`으로 새 버전을 확인합니다.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/ botocore/1.5.47
```

루트 권한이 없는 경우 AWS CLI를 사용자 모드로 설치하십시오.

Amazon Linux(사용자)에서 AWS CLI를 업그레이드하려면

1. `pip install`을 사용하여 최신 버전의 AWS CLI를 설치하십시오.

```
sudo pip install --upgrade --user awscli
```

2. 설치 위치를 `PATH` 변수의 시작 부분에 추가합니다.

```
export PATH=/home/ec2-user/.local/bin:$PATH
```

   이 명령을 `~/.bashrc`의 끝 부분에 추가하여 세션 간 변경을 유지합니다.

3. `aws --version`으로 새 버전을 확인합니다.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/ botocore/1.5.47
```

Microsoft Windows에 AWS Command Line Interface 설치

독립 실행형 설치 관리자 또는 Python용 패키지 관리자인 `pip`를 사용하여 Windows에 AWS CLI를 설치할 수 있습니다. `pip`가 이미 있는 경우 기본 설치 주제 (p. 4)의 지침을 따릅니다.

단원

- MSI 설치 관리자 (p. 9)
- Windows에서 Python, pip 및 AWS CLI 설치 (p. 10)
- 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 11)

MSI 설치 관리자

AWS CLI는 Microsoft Windows XP 이상에서 지원됩니다. Windows 사용자의 경우 MSI 설치 패키지는 다른 사전 요구 사항을 설치하지 않고 AWS CLI를 설치할 수 있는 친숙하고 편리한 방법을 제공합니다.

업데이트가 릴리스되면 설치 프로세스를 반복하여 최신 버전의 AWS CLI를 가져와야 합니다. 자주 업데이트하려는 경우 간편한 업데이트를 위해 pip 사용 (p. 10)을 고려해 보십시오.

MSI 설치 관리자를 사용하여 AWS CLI를 설치하려면

1. 적절한 MSI 설치 관리자를 다운로드합니다.

   - Windows용 AWS CLI MSI 설치 관리자(64비트) 다운로드
   - Windows용 AWS CLI MSI 설치 관리자(32비트) 다운로드
   - AWS CLI 설정 파일 다운로드(32비트 및 64비트 MSI 설치 관리자 모두 포함하며, 정확한 버전을 자동으로 설치함)

Note

AWS CLI용 MSI 설치 관리자는 Windows Server 2008(버전 6.0.6002)에서 작동하지 않습니다. 이 버전의 Windows에는 pip (p. 10)를 사용하여 설치하십시오.

2. 다운로드한 MSI 설치 관리자 또는 설정 파일을 실행하십시오.
3. 화면에 표시되는 지시 사항을 따릅니다.

CLI는 기본적으로 `C:\Program Files\Amazon\AWSCLI`(64비트 버전) 또는 `C:\Program Files (x86)\Amazon\AWSCLI`(32비트 버전)에 설치됩니다. 설치를 확인하려면 명령 프롬프트에서 `aws --version` 명령을 사용합니다. 시작 메뉴를 열고 `cmd`를 검색하여 명령 프롬프트를 시작할 수 있습니다.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Windows/7 botocore/1.5.47
```

명령을 입력할 때 프롬프트 기호(위의 'C:\>')를 포함시키지 마십시오. 이 기호는 입력하는 명령을 CLI에서 반환되는 출력과 구별하기 위해 프로그램 목록에 포함되어 있습니다. 이 설명서의 나머지 부분에서는 명령이 Windows에 특정한 경우를 제외하고 일반 프롬프트 기호 '$ '를 사용합니다.

Windows에서 프로그램을 찾을 수 없는 경우 명령 프롬프트를 닫고 다시 열어 경로를 새로 고치거나 설치 디렉터리를 PATH (p. 11) 환경 변수에 수동으로 추가해야 할 수 있습니다.

MSI 설치 업데이트

AWS CLI는 정기적으로 업데이트됩니다. 최신 버전이 언제 릴리스되었는지를 알아보려면 GitHub의 릴리스 페이지를 확인하십시오. 최신 버전으로 업데이트하려면 위의 설명에 따라 MSI 설치 관리자를 다시 다운로드하여 실행합니다.

제거

AWS CLI를 제거하려면 제어판을 열고 Programs and Features(프로그램 및 기능)를 선택합니다. AWS Command Line Interface라는 항목을 선택하고 Uninstall(제거)을 클릭하여 제거 프로그램을 시작합니다. 메시지가 나타나면 를 설치 제거할지 확인하십시오.

명령줄에서 다음 명령을 사용하여 Programs and Features(프로그램 및 기능) 프로그램을 시작할 수도 있습니다.

```
appwiz.cpl
```

Windows에서 Python, pip 및 AWS CLI 설치

Python Software Foundation은 `pip`가 포함된 Windows용 설치 관리자를 제공합니다.

Python 3 및 pip 설치 방법(Windows)

1. Python.org의 다운로드 페이지에서 Python 3 Windows x86-64 설치 관리자를 다운로드합니다.
2. 설치 관리자를 실행합니다.
3. Add Python 3 to PATH(PATH에 Python 3 추가)를 선택합니다.
4. Install Now를 선택합니다.

설치 관리자가 사용자 폴더에 Python을 설치하고, 사용자 경로에 프로그램 폴더를 추가합니다.

pip를 사용하여 AWS CLI를 설치하려면(Windows)

1. 시작 메뉴에서 Windows 명령 프롬프트를 여십시오.
2. 다음 명령을 사용하여 Python과 pip가 모두 올바르게 설치되었는지 확인합니다.

```
C:\Windows\System32> python --version
Python 3.7.1
C:\Windows\System32> pip --version
pip 18.1 from c:\program files\python37\lib\site-packages\pip (python 3.7)
```

3. `pip`를 사용하여 AWS CLI를 설치하십시오.

```
C:\Windows\System32> pip install awscli
```

4. AWS CLI가 올바르게 설치되었는지 확인하십시오.

```
C:\Windows\System32> aws --version
aws-cli/1.11.84 Python/3.6.2 Windows/10 botocore/1.5.47
```

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

```
C:\Windows\System32> pip install --user --upgrade awscli
```

명령줄 경로에 AWS CLI 실행 파일 추가

`pip`를 사용하여 설치한 후 `aws` 프로그램을 OS의 `PATH` 환경 변수에 추가합니다. MSI 설치에서는 이 작업이 자동으로 수행되지만, `aws` 명령이 작동하지 않는 경우 이 항목을 수동으로 설정해야 할 수 있습니다.

- Python 3 및 pip – `C:\Program Files\Python37\Scripts\`
- Python 3 및 pip --사용자 옵션 – `%USERPROFILE%\AppData\Local\Programs\Python\Python37\Scripts`
- MSI 설치 관리자(64비트) – `C:\Program Files\Amazon\AWSCLI`
- MSI 설치 관리자(32비트) – `C:\Program Files (x86)\Amazon\AWSCLI`

PATH 변수를 수정하려면(Windows)

1. Windows 키를 누르고 **`environment variables`**를 입력하십시오.
2. [Edit environment variables for your account]를 선택합니다.
3. [PATH]를 선택한 후 [Edit]를 선택합니다.
4. 세미콜론으로 구분하여 경로를 [Variable value] 필드에 추가합니다. 예: *`C:\existing\path`***`;C:\new\path`**
5. [OK]를 두 번 선택하여 새로운 설정을 적용합니다.
6. 실행 중인 명령 프롬프트를 모두 닫았다가 다시 엽니다.

macOS에 AWS Command Line Interface을 설치하십시오.

macOS에 AWS CLI를 설치할 때 번들 설치 관리자를 사용하는 것이 좋습니다. 번들 설치 관리자에는 모든 종속 항목이 포함되고 오프라인으로 사용할 수 있습니다.

Important

번들 설치 관리자는 공백을 포함하는 경로에 설치하는 것을 지원하지 않습니다.

단원

사전 조건

- Python 2 버전 2.6.5+ 또는 Python 3 버전 3.3+

Python 설치를 확인합니다.

```
python --version
```

컴퓨터에 아직 Python이 설치되지 않았거나 다른 버전의 Python을 설치하려는 경우 Linux에 AWS Command Line Interface 설치 (p. 5)의 절차를 수행하십시오.

번들 설치 관리자를 사용하여 AWS CLI 설치

번들 설치 관리자를 사용하여 AWS CLI를 설치하려면 명령줄에서 다음 단계를 수행하십시오.

번들 설치 관리자를 사용하여 AWS CLI를 설치하려면

1. AWS CLI 번들 설치 관리자를 다운로드하십시오.

   ```
   curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
   ```

2. 패키지의 압축을 풉니다.

   ```
   unzip awscli-bundle.zip
   ```

   Note

   `unzip`이 없는 경우 Linux 배포의 내장된 패키지 관리자를 사용하여 설치하십시오.

3. 설치 프로그램을 실행합니다.

   ```
   sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
   ```

   Note

   기본적으로 설치 스크립트는 시스템 기본 버전의 Python에서 실행됩니다. 대체 버전의 Python을 설치하고 이를 사용하여 AWS CLI를 설치하려는 경우, Python 프로그램에 대한 절대 경로를 포함하여 해당 버전을 지정하는 설치 스크립트를 실행하십시오. 예:

   ```
   $ sudo /usr/local/bin/python3.6 awscli-bundle/install -i /usr/local/aws -b /usr/
   local/bin/aws
   ```

이 명령은 `/usr/local/aws`에 AWS CLI를 설치하고 `/usr/local/bin` 디렉터리에 symlink `aws`를 생성합니다. `-b` 옵션을 사용하여 symlink를 생성하면 사용자의 `$PATH` 변수에 설치 디렉터리를 지정할 필요가 없습니다. 이렇게 하면 모든 사용자가 임의 디렉터리에서 `aws`를 입력하여 AWS CLI를 호출 가능해야 합니다.

`-i` 및 `-b` 옵션에 대한 설명을 보려면 `-h` 옵션을 사용합니다.

```
./awscli-bundle/install -h
```

pip을 사용하여 macOS에 AWS CLI를 설치하십시오.

pip을 사용하여 직접 AWS CLI를 설치할 수도 있습니다. pip가 없는 경우 기본 설치 주제 (p. 4)의 지침을 따릅니다. `pip --version`을 실행하여 macOS 버전에 Python과 pip가 이미 포함되어 있는지 확인합니다.

```
pip --version
```

macOS에 AWS CLI를 설치하려면

1. Python.org의 다운로드 페이지에서 Python 3.6을 다운로드 및 설치합니다.
2. Python Packaging Authority에서 제공하는 `pip` 설치 스크립트를 다운로드하고 실행합니다.

   ```
   curl -O https://bootstrap.pypa.io/get-pip.py
   python3 get-pip.py --user
   ```
3. 새로 설치된 `pip`를 사용하여 AWS CLI를 설치합니다.

   ```
   pip install awscli --upgrade --user
   ```
4. AWS CLI가 올바르게 설치되었는지 확인하십시오.

   ```
   aws --version
   AWS CLI 1.11.84 (Python 3.7.1)
   ```

   프로그램을 찾을 수 없는 경우 프로그램을 명령줄 경로에 추가 (p. 13)합니다.

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

```
pip install awscli --upgrade --user
```

명령줄 경로에 AWS CLI 실행 파일 추가

`pip`를 사용하여 설치한 후 `aws` 프로그램을 OS의 `PATH` 환경 변수에 추가해야 할 수 있습니다. 프로그램의 위치는 Python 설치 위치에 따라 달라집니다.

Example AWS CLI 설치 위치 - Python 3.7 및 pip가 포함된 macOS(사용자 모드)

```
~/Library/Python/3.7/bin
```

Python 설치 위치를 모르는 경우, `which python`을 실행하십시오.

```
which python
/usr/local/bin/python
```

실제 프로그램이 아니라 symlink 경로가 출력될 수 있습니다. `ls -al`을 실행하여 어디를 가리키는지 확인합니다.

```
ls -al /usr/local/bin/python
~/Library/Python/3.7/bin/python3.7
```

`pip`는 Python 프로그램이 있는 것과 동일한 폴더에 프로그램을 설치합니다. 이 폴더를 `PATH` 변수에 추가합니다.

PATH 변수(Linux, macOS, or Unix)를 수정하려면

1. 사용자 폴더에서 셸의 프로파일 스크립트를 찾습니다. 어떤 셸을 가지고 있는지 잘 모르는 경우 `echo $SHELL`을 실행합니다.

   ```
   $ ls -a ~
   .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
   ```

   - Bash – `.bash_profile`, `.profile` 또는 `.bash_login`.
   - Zsh – `.zshrc`
   - Tcsh – `.tcshrc`, `.cshrc` 또는 `.login`.
2. 내보내기 명령을 프로필 스크립트에 추가하십시오.

   ```
   export PATH=~/.local/bin:$PATH
   ```

   이 명령은 이 예제의 `~/.local/bin` 경로를 현재 PATH 변수에 추가합니다.
3. 프로필을 현재 세션에 로드합니다.

   ```
   $ source ~/.bash_profile
   ```

가상 환경에 AWS Command Line Interface을 설치하십시오.

가상 환경에 AWS CLI를 설치하면 요구 사항 버전이 다른 pip 패키지와 충돌하는 것을 방지할 수 있습니다.

가상 환경에 AWS CLI를 설치하려면

1. `pip`를 사용하여 `virtualenv`를 설치합니다.

   ```
   pip install --user virtualenv
   ```
2. 가상 환경을 생성하고 이름을 지정합니다.

   ```
   virtualenv ~/cli-ve
   ```

   또는 `-p` 옵션을 사용하여 기본 버전 이외의 Python 버전을 지정할 수 있습니다.

   ```
   virtualenv -p /usr/bin/python3.4 ~/cli-ve
   ```
3. 새 가상 환경을 활성화합니다.

   Linux, macOS, or Unix

```
source ~/cli-ve/bin/activate
```

Windows

```
%USERPROFILE%\cli-ve\Scripts\activate
```

4. 가상 환경에 AWS CLI를 설치합니다.

```
(cli-ve)~$ pip install --upgrade awscli
```

5. AWS CLI가 올바르게 설치되었는지 확인하십시오.

```
aws --version
aws-cli/1.11.84 Python/3.6.2 Linux/4.4.0-59-generic botocore/1.5.47
```

`deactivate` 명령을 사용하여 가상 환경을 종료할 수 있습니다. 새 세션을 시작할 때마다 환경을 다시 활성화해야 합니다.

최신 버전으로 업그레이드하려면 설치 명령을 다시 실행합니다.

```
(cli-ve)~$ pip install --upgrade awscli
```

번들 설치 관리자(Linux, macOS, or Unix)를 사용하여 AWS CLI 설치

Linux, macOS, or Unix에서 번들 설치 관리자를 사용하여 AWS CLI를 설치할 수 있습니다. 번들 설치 관리자에는 모든 종속 항목이 포함되고 오프라인으로 사용할 수 있습니다.

> **Important**
>
> 번들 설치 관리자는 공백을 포함하는 경로에 설치하는 것을 지원하지 않습니다.

단원

사전 조건

- Linux, macOS, or Unix
- Python 2 버전 2.6.5+ 또는 Python 3 버전 3.3+

Python 설치를 확인합니다.

```
python --version
```

컴퓨터에 아직 Python이 설치되지 않았거나 다른 버전의 Python을 설치하려는 경우 Linux에 AWS Command Line Interface 설치 (p. 5)의 절차를 수행하십시오.

번들 설치 관리자를 사용하여 AWS CLI 설치

번들 설치 관리자를 사용하여 AWS CLI를 설치하려면 명령줄에서 다음 단계를 수행하십시오.

번들 설치 관리자를 사용하여 AWS CLI를 설치하려면

1. AWS CLI 번들 설치 관리자를 다운로드하십시오.

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
```

2. 패키지의 압축을 풉니다.

```
$ unzip awscli-bundle.zip
```

   Note

   `unzip`이 없는 경우 Linux 배포의 내장된 패키지 관리자를 사용하여 설치하십시오.

3. 설치 실행 파일을 실행합니다.

```
$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

   Note

   기본적으로 설치 스크립트는 시스템 기본 버전의 Python에서 실행됩니다. 대체 버전의 Python을 설치하고 이를 사용하여 AWS CLI를 설치하려는 경우, Python 실행 파일에 대한 절대 경로로 해당 버전의 설치 스크립트를 실행하십시오. 예:

```
$ sudo /usr/local/bin/python3.7 awscli-bundle/install -i /usr/local/aws -b /usr/
local/bin/aws
```

설치 관리자는 `/usr/local/aws`에서 AWS CLI를 설치하고 `/usr/local/bin` 디렉터리에 `aws`를 생성합니다. `-b` 옵션을 사용하여 symlink를 생성하면 사용자의 `$PATH` 변수에 설치 디렉터리를 지정할 필요가 없습니다. 이렇게 하면 모든 사용자가 임의 디렉터리에서 `aws`를 입력하여 AWS CLI를 호출 가능해야 합니다.

`-i` 및 `-b` 옵션에 대한 설명을 보려면 `-h` 옵션을 사용합니다.

```
$ ./awscli-bundle/install -h
```

Sudo(Linux, macOS, or Unix) 없이 AWS CLI 설치

sudo 권한이 없거나 현재 사용자에 대해서만 AWS CLI를 설치할 경우 위 명령의 수정된 버전을 사용할 수 있습니다.

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
$ unzip awscli-bundle.zip
$ ./awscli-bundle/install -b ~/bin/aws
```

이렇게 하여 AWS CLI를 기본값 위치(`~/.local/lib/aws`)에 설치하고 `~/bin/aws`에 심볼 링크(symlink)를 생성합니다. symlink가 작동하려면 `~/bin`이 `PATH` 환경 변수에 있어야 합니다.

```
$ echo $PATH | grep ~/bin      // See if $PATH contains ~/bin (output will be empty if it
 doesn't)
$ export PATH=~/bin:$PATH      // Add ~/bin to $PATH if necessary
```

Tip

세션 간에 $PATH 설정이 유지되도록 하려면 `export` 줄을 셸 프로필에 추가합니다(~/.profile, ~/.bash_profile 등).

제거

번들 설치 관리자는 선택적 symlink를 제외하고 설치 디렉터리 외부에 아무 것도 넣지 않으므로, 설치 제거는 이 두 항목을 삭제하는 것만큼 간단합니다.

```
$ sudo rm -rf /usr/local/aws
$ sudo rm /usr/local/bin/aws
```

AWS CLI 구성

이 단원에서는 보안 자격 증명, 기본 출력 형식, 기본 리전을 비롯하여 AWS CLI가 AWS와 상호 작용하는 데 사용하는 설정을 구성하는 방법을 설명합니다.

> Note
>
> AWS에서는 모든 수신 요청이 암호화 서명되어야 합니다. AWS CLI에서 이 작업을 수행합니다. '서명'에는 날짜/시간 타임스탬프가 포함됩니다. 따라서 컴퓨터의 날짜 및 시간이 올바르게 설정되어야 합니다. 잘못 설정되면 서명의 날짜/시간과 AWS 제품에서 인식한 날짜/시간의 차이가 극심하여 AWS에서 요청을 거부합니다.

단원

빠른 구성

일반적인 용도에서 `aws configure` 명령은 AWS CLI 설치를 설정할 수 있는 가장 빠른 방법입니다.

```
aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

이 명령을 입력하면 AWS CLI에서 네 가지 정보를 묻는 메시지를 표시하고 이를 `default`라는 프로필(설정 모음)에 저장합니다. 이 프로필은 사용할 프로필을 명시적으로 지정하지 않는 AWS CLI 명령을 실행할 때마다 사용됩니다.

액세스 키/자격 증명

`AWS Access Key ID` 및 `AWS Secret Access Key`는 AWS 자격 증명입니다. 이러한 자격 증명은 사용자에게 부여되는 권한을 결정하는 IAM 사용자 또는 역할과 연결되어 있습니다. IAM 서비스를 통해 사용자를 생성하는 방법에 대한 자습서는 IAM 사용 설명서의 첫 번째 IAM 관리자 사용자 및 그룹 생성을 참조하십시오.

IAM 사용자에 대한 액세스 키 ID 및 보안 액세스 키를 얻어야 합니다.

액세스 키는 액세스 키 ID 및 보안 액세스 키로 이루어져 있는데, 이를 사용하여 AWS에 보내는 프로그래밍 방식의 요청에 서명할 수 있습니다. 액세스 키가 없는 경우에는 AWS Management 콘솔에서 액세스 키를 생

성할 수 있습니다. AWS 계정 루트 사용자 액세스 키 대신에 IAM 액세스 키를 사용하는 것이 좋습니다. IAM으로 AWS 서비스와 AWS 계정의 리소스에 대한 액세스 권한을 안전하게 제어할 수 있습니다.

보안 액세스 키는 액세스 키를 생성하는 시점에만 보고 다운로드할 수 있습니다. 나중에 복구할 수 없습니다. 하지만 언제든지 새 액세스 키를 생성할 수 있습니다. 필요한 IAM 작업을 수행할 수 있는 권한도 있어야 합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 리소스에 액세스하는 데 필요한 권한을 참조하십시오.

1. IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [Users]를 선택합니다.
3. IAM 사용자 이름(확인란이 아님)을 선택합니다.
4. [Security credentials] 탭을 선택한 후 [Create access key]를 선택합니다.
5. 새 액세스 키를 보려면 [Show]를 선택합니다. 자격 증명은 다음()과 동일합니다.
   - 액세스 키 ID: AKIAIOSFODNN7EXAMPLE
   - 보안 액세스 키: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. 키 페어 파일을 다운로드하려면 [Download .csv file]을 선택합니다. 안전한 위치에 키를 저장합니다.

   AWS 계정을 보호하기 위해서는 키를 기밀로 유지하도록 하고, 결코 이메일로 전송해서는 안 됩니다. AWS 또는 Amazon.com의 이름으로 문의가 온다 할지라도 조직 외부로 키를 공유하지 마십시오. Amazon을 합법적으로 대표하는 사람이라면 결코 보안 키를 요구하지 않을 것입니다.

관련 주제

- IAM이란?(출처: IAM 사용 설명서) 단원 참조.
- AWS 보안 자격 증명(출처: AWS General Reference) 단원 참조.

리전

`Default region name`은 기본적으로 요청을 전송할 서버가 있는 리전을 식별합니다. 이 리전은 일반적으로 가장 가까운 리전이지만 어떤 리전이든 될 수 있습니다. 예를 들어, `us-west-2`를 입력하여 미국 서부(오레곤)를 사용할 수 있습니다. 개별 명령으로 달리 지정하지 않는 한 이후의 모든 요청이 전송되는 리전입니다.

Note

AWS CLI를 사용하여 명시적으로 또는 기본 리전을 설정하여 AWS 리전을 지정해야 합니다. 사용 가능한 리전 목록은 리전 및 엔드포인트를 참조하십시오. AWS CLI에서 사용하는 리전 표기는 AWS Management 콘솔 URL 및 서비스 엔드포인트에서 사용하는 것과 동일한 이름입니다.

출력 형식

`Default output format`은 결과의 형식을 지정하는 방법을 지정합니다. 값은 다음 목록에 있는 값 중 하나일 수 있습니다. 출력 형식을 지정하지 않으면 `json`이 기본값으로 사용됩니다.

빠른 구성 및 다중 프로필

위에 표시된 명령을 사용할 경우 결과는 이름이 `default`인 단일 프로필이 됩니다. `--profile` 옵션을 통해 프로필 이름을 지정하여 추가 구성을 생성할 수도 있습니다.

```
aws configure --profile user2
AWS Access Key ID [None]: AKIAI44QH8DHBEXAMPLE
AWS Secret Access Key [None]: je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

```
Default region name [None]: us-east-1
Default output format [None]: text
```

그런 다음 명령을 실행할 때 `--profile` 옵션을 생략하고 `default` 프로필에 저장된 설정을 사용할 수 있습니다.

```
aws s3 ls
```

또는 `--profile` *profilename*을 지정하고 해당 이름으로 저장된 설정을 사용할 수 있습니다.

```
aws s3 ls --profile myuser
```

설정을 업데이트하려면 `aws configure`를 다시 실행(업데이트할 프로필에 따라 `--profile` 파라미터를 사용하거나 사용하지 않고)하고 새 값을 적절하게 입력합니다. 다음 단원에서는 `aws configure`에서 생성되는 파일, 추가 설정 및 명명된 프로필에 대해 자세히 설명합니다.

구성 설정 및 우선 순위

AWS CLI는 여러 자격 증명 공급자를 사용하여 AWS 자격 증명을 찾습니다. 각 자격 증명 공급자는 시스템 또는 사용자 환경 변수, 로컬 AWS 구성 파일 또는 명령줄에서 파라미터로 명시적으로 선언된 위치 등 다양한 장소에서 자격 증명을 찾습니다. AWS CLI는 다음 순서로 공급자를 호출하고 사용할 자격 증명 세트를 찾은 경우 중지하여 자격 증명 및 구성 설정을 찾습니다.

1. 명령줄 옵션 (p. 24) – 명령줄의 파라미터로 `--region`, `--output` 및 `--profile`을 지정할 수 있습니다.
2. 환경 변수 (p. 23) – 환경 변수 `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` 및 `AWS_SESSION_TOKEN`에 값을 저장할 수 있습니다. 변수가 있는 경우 해당 변수가 사용됩니다.
3. CLI 자격 증명 파일 (p. 20) – 이 파일은 `aws configure` 명령을 실행할 때 업데이트되는 파일 중 하나입니다. 이 파일은 Linux, macOS, or Unix의 경우 `~/.aws/credentials`에, Windows의 경우 `C:\Users\`*USERNAME*`\.aws\credentials`에 있습니다. 이 파일에는 `default` 프로필 및 모든 명명된 프로필에 대한 자격 증명 세부 정보가 포함되어 있습니다.
4. CLI 구성 파일 (p. 20) – 이 파일은 `aws configure` 명령을 실행할 때 업데이트되는 파일 중 또 다른 하나입니다. 이 파일은 Linux, macOS, or Unix의 경우 `~/.aws/config`에, Windows의 경우 `C:\Users\`*USERNAME*`\.aws\config`에 있습니다. 이 파일에는 기본 프로필 및 모든 명명된 프로필에 대한 구성 설정이 포함되어 있습니다.
5. 컨테이너 자격 증명 – IAM 역할을 각각의 Amazon Elastic Container Service 작업 정의와 연결할 수 있습니다. 그러면 작업의 컨테이너에 대해 해당 역할의 임시 자격 증명을 사용할 수 있습니다. 자세한 내용은 Amazon Elastic Container Service Developer Guide의 작업에 대한 IAM 역할을 참조하십시오.
6. 인스턴스 프로필 자격 증명 – IAM 역할을 각각의 Amazon EC2(Amazon Elastic Compute Cloud) 인스턴스와 연결할 수 있습니다. 그러면 인스턴스에서 실행되는 코드에 대해 해당 역할의 임시 자격 증명을 사용할 수 있습니다. 자격 증명은 Amazon EC2 메타데이터 서비스를 통해 전달됩니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2에 대한 IAM 역할 및 IAM 사용 설명서의 인스턴스 프로필 사용을 참조하십시오.

구성 및 자격 증명 파일

CLI는 `aws configure`를 사용하여 지정하는 자격 증명을 홈 디렉터리의 `.aws`라는 폴더에 있는 `credentials`라는 로컬 파일에 저장합니다. `aws configure`를 사용하여 지정하는 다른 구성 옵션은 `config`라는 로컬 파일에 저장되며, 홈 디렉터리의 `.aws` 폴더에도 저장됩니다.

홈 디렉터리 위치는 운영 체제에 따라 달라지지만 Windows에서는 `%UserProfile%` 환경 변수를, Unix 기반 시스템에서는 `$HOME` 또는 `~`(물결표) 환경 변수를 사용하여 참조됩니다.

예를 들어, 다음 명령은 `.aws` 폴더의 내용을 나열합니다.

Linux, macOS, or Unix

```
ls  ~/.aws
```

Windows

```
dir "%UserProfile%\.aws"
```

AWS CLI는 두 개의 파일을 사용하여 중요한 자격 증명 정보(`~/.aws/credentials`에 보관)를 중요도가 낮은 구성 옵션(`~/.aws/config`에 보관)과 구분하여 보관합니다.

`AWS_CONFIG_FILE` 환경 변수를 다른 로컬 경로로 설정하여 `config` 파일에 대한 기본 위치가 아닌 위치를 지정할 수 있습니다. 세부 정보는 환경 변수 (p. 23) 단원을 참조하십시오.

Config에 자격 증명 저장

AWS CLI는 `config` 파일에서 자격 증명을 읽을 수도 있습니다. 모든 프로필 설정을 단일 파일에 저장하려는 경우 그렇게 할 수 있습니다. 두 위치에 프로필에 대한 자격 증명이 있는 경우(예를 들어 `aws configure`를 사용하여 프로필의 키를 업데이트한 경우) 자격 증명 파일의 키가 우선적으로 적용됩니다.
AWS CLI에 추가하여 SDK 중 하나를 사용하는 경우 자격 증명을 고유의 파일에 저장하지 않으면 추가 경고가 표시될 수 있습니다.

이전 단원에서 구성한 프로필에 대해 CLI에서 생성된 파일은 다음과 같습니다.

`~/.aws/credentials`

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

`~/.aws/config`

```
[default]
region=us-west-2
output=json
```

Note

이전 예제는 단일의 기본 프로필이 있는 파일을 보여줍니다. 여러 개의 명명된 프로필이 있는 파일의 예는 명명된 프로필 (p. 22) 단원을 참조하십시오.

지원되는 설정은 다음과 같습니다.

`aws_access_key_id` – AWS 액세스 키

`aws_secret_access_key` – AWS 보안 키

`aws_session_token` – AWS 세션 토큰. 세션 토큰은 임시 보안 자격 증명을 사용하는 경우에만 필요합니다.

`region` – 이 프로필에서 요청을 전송할 기본 AWS 리전

output (p. 19) – 이 프로필에 대한 기본 출력 형식

명명된 프로필

AWS CLI는 config 및 credentials 파일에 저장된 여러 명명된 프로필의 사용을 지원합니다. aws configure를 --profile 옵션과 함께 사용하거나 config 및 credentials 파일에 항목을 추가하여 추가 프로필을 구성할 수 있습니다.

다음은 두 개의 프로필이 있는 credentials 파일을 보여주는 예입니다. 첫 번째 프로필은 프로필이 없는 CLI 명령을 실행할 때 사용되며, 두 번째 프로필은 --profile user1 파라미터와 함께 CLI 명령을 실행할 때 사용됩니다.

~/.aws/credentials

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[user1]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

각 프로필은 다른 자격 증명을 사용하며(아마도 다른 IAM 사용자의 자격 증명), 다른 리전 및 출력 형식도 사용할 수 있습니다.

~/.aws/config

```
[default]
region=us-west-2
output=json

[profile user1]
region=us-east-1
output=text
```

> **Important**
>
> credentials 파일은 명명된 프로필에 대해 CLI config 파일과 다른 명명 형식을 사용합니다. config 파일에서 명명된 프로필을 구성할 때에만 'profile' 접두사를 포함합니다. credentials 파일을 구성할 때에는 profile을 사용하지 마십시오.

AWS CLI에서 프로필 사용

명명된 프로필을 사용하려면 --profile *profile-name* 옵션을 명령에 추가합니다. 다음은 이전 예제 파일의 user1 프로필을 사용하여 Amazon EC2 인스턴스를 모두 나열하는 예입니다.

```
aws ec2 describe-instances --profile user2
```

여러 명령에 대해 명명된 프로필을 사용하려는 경우 명령줄에서 AWS_PROFILE 환경 변수를 설정하면 모든 명령에서 매번 프로필을 지정하는 것을 피할 수 있습니다.

Linux, macOS, or Unix

```
export AWS_PROFILE=user2
```

Windows

```
set AWS_PROFILE=user2
```

환경 변수를 설정하면 기본 프로필이 변경되어 셸 세션이 종료될 때까지 또는 변수를 다른 값으로 설정할 때까지 유지됩니다. 변수에 대해서는 다음 단원에서 자세히 설명합니다.

환경 변수

환경 변수는 구성 옵션과 자격 증명을 지정하는 다른 방법을 제공하며, 스크립팅을 수행하거나 명명된 프로필을 임시로 기본값으로 설정할 때 유용할 수 있습니다.

> **Important**
>
> 다음 환경 변수 중 하나를 설정하면 해당 옵션을 지정하는 다른 모든 방법이 재정의됩니다. 단, 옵션을 명령줄 파라미터로 지정하는 방법은 제외됩니다.

AWS CLI는 다음과 같은 환경 변수를 지원합니다.

- `AWS_ACCESS_KEY_ID` – IAM 사용자 또는 역할과 연결된 AWS 액세스 키를 지정합니다.
- `AWS_SECRET_ACCESS_KEY` – 액세스 키와 연결된 보안 키를 지정합니다. 이는 액세스에 대한 기본적인 '암호'입니다.
- `AWS_SESSION_TOKEN` – 임시 보안 자격 증명을 사용하는 경우 필요한 세션 토큰 값을 지정합니다. 자세한 내용은 AWS CLI Command Reference의 assume-role 명령의 출력 섹션을 참조하십시오.
- `AWS_DEFAULT_REGION` – 요청을 전송할 AWS 리전 (p. 19)을 지정합니다.
- `AWS_DEFAULT_OUTPUT` – 사용할 출력 형식을 지정합니다.
- `AWS_PROFILE` – 사용할 자격 증명과 옵션이 있는 CLI 프로필 (p. 22)의 이름을 지정합니다. 이 이름은 `credentials` 또는 `config` 파일에 저장된 프로필 이름이거나 기본 프로필을 사용할 값 `default`일 수 있습니다.
- `AWS_CA_BUNDLE` – HTTPS 인증서 확인에 사용할 인증서 번들의 경로를 지정합니다.
- `AWS_SHARED_CREDENTIALS_FILE` – AWS CLI가 액세스 키를 저장하는 데 사용하는 파일의 위치를 지정합니다(기본값: `~/.aws/credentials`).
- `AWS_CONFIG_FILE` – AWS CLI가 구성 프로필을 저장하는 데 사용하는 파일의 위치를 지정합니다(기본값: `~/.aws/config`).

다음은 기본 사용자에 대한 환경 변수를 구성하는 방법을 보여주는 예입니다. 이러한 값은 명명된 프로필 또는 인스턴스 메타데이터에서 찾은 모든 값을 재정의합니다. 설정된 경우 CLI 명령줄의 파라미터를 지정하거나 환경 변수를 변경 또는 제거하여 해당 값을 재정의할 수 있습니다.

Linux, macOS, or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_DEFAULT_REGION=us-west-2
```

Windows

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_DEFAULT_REGION=us-west-2
```

명령줄 옵션

다음 명령줄 옵션을 사용하여 단일 명령에 대한 기본 구성 설정을 재정의할 수 있습니다. 명령줄 옵션을 통해 사용할 프로필을 지정할 수 있지만, 해당 옵션으로 자격 증명을 직접 지정할 수 없습니다.

--profile

이 명령에 사용할 명명된 프로필 (p. 22)을 지정합니다. 명명된 프로필을 추가로 설정하려면 `aws configure` 명령을 `--profile` 옵션과 함께 사용하면 됩니다.

```
aws configure --profile <profilename>
```

--region

이 명령의 AWS 요청을 전송할 AWS 리전을 지정합니다. 지정할 수 있는 모든 리전 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.

--출력

이 명령에 사용할 출력 형식을 지정합니다. 다음 값 중 하나를 지정할 수 있습니다.

--endpoint-url

요청을 전송할 URL입니다. 대부분의 명령에서는 AWS CLI가 선택된 서비스 및 지정된 AWS 리전을 기반으로 자동으로 URL을 결정합니다. 하지만 일부 명령에서는 계정별 URL을 지정해야 합니다. 일부 AWS 제품을 구성하여 프라이빗 VPC 내에서 직접 엔드포인트를 호스팅할 수도 있습니다. 이렇게 하려면 지정해야 합니다.

각 리전에서 사용할 수 있는 표준 서비스 엔드포인트 목록은 Amazon Web Services 일반 참조의 AWS 리전 및 엔드포인트를 참조하십시오.

이러한 옵션 중 하나 이상을 명령줄 파라미터로 제공하면 단일 명령에 대한 기본 구성 또는 해당 프로필 설정이 재정의됩니다. 각 옵션은 공백이나 등호(=)를 사용하여 인수를 옵션 이름과 구분하는 문자열 인수를 가져옵니다. 인수 값에 공백이 포함되는 경우 해당 인수의 앞뒤에 따옴표를 사용해야 합니다.

명령줄 옵션의 일반적인 용도는 여러 AWS 리전에서 리소스를 확인하고 읽기 쉽게 또는 스크립팅할 때 사용하기 쉽게 출력 형식을 변경하는 것입니다. 예를 들어, 어떤 리전에서 인스턴스가 실행 중인지 잘 모르는 경우 다음과 같이 해당 리전을 찾을 때까지 각 리전에 대해 describe-instances 명령을 실행할 수 있습니다.

```
$ aws ec2 describe-instances --output table --region us-east-1
------------------
|DescribeInstances|
+-----------------+
$ aws ec2 describe-instances --output table --region us-west-1
------------------
|DescribeInstances|
+-----------------+
$ aws ec2 describe-instances --output table --region us-west-2
-----------------------------------------------------------------------------
|                            DescribeInstances                              |
+---------------------------------------------------------------------------+
||                              Reservations                                 ||
|+-----------------------------------+-------------------------------------+|
||  OwnerId                          |  012345678901                       ||
||  ReservationId                    |  r-abcdefgh                         ||
|+-----------------------------------+-------------------------------------+|
|||                               Instances                                 |||
||+-------------------------+-----------------------------------------------+||
|||  AmiLaunchIndex         |  0                                            |||
|||  Architecture           |  x86_64                                       |||
```

```
...
```

각 명령줄 옵션에 대한 인수 유형(문자열, Boolean 등)은 파라미터 값 지정 (p. 40)에서 자세히 설명합니다.

인스턴스 메타데이터

Amazon EC2 인스턴스 내에서 AWS CLI를 실행할 때 명령에 자격 증명을 간편하게 제공할 수 있습니다. 각 Amazon EC2 인스턴스에는 AWS CLI가 임시 자격 증명에 대해 직접 쿼리할 수 있는 메타데이터가 포함되어 있습니다. 이를 제공하려면 필요한 리소스에 액세스할 수 있는 IAM 역할을 생성하고 해당 역할을 시작할 때 Amazon EC2 인스턴스에 연결합니다.

인스턴스를 시작하고 AWS CLI가 이미 설치되어 있는지 확인하십시오(Amazon Linux에는 미리 설치된 상태로 제공됨). 필요한 경우 AWS CLI를 설치합니다. 모든 명령에서 지정할 필요가 없도록 기본 리전을 계속 구성해야 합니다.

Enter를 두 번 눌러 처음 두 개의 프롬프트를 건너뛰면 자격 증명을 지정하지 않고 `aws configure`를 실행하여 리전 및 기본 출력 형식을 설정할 수 있습니다.

```
aws configure
AWS Access Key ID [None]: ENTER
AWS Secret Access Key [None]: ENTER
Default region name [None]: us-west-2
Default output format [None]: json
```

IAM 역할이 인스턴스에 연결되면 AWS CLI가 인스턴스 메타데이터에서 자격 증명을 자동으로 안전하게 검색합니다. 자세한 내용은 IAM 사용 설명서의 Amazon EC2 인스턴스에서 실행하는 애플리케이션에 AWS 리소스에 대한 액세스 권한 부여를 참조하십시오.

HTTP 프록시 사용

프록시 서버를 통해 AWS에 액세스해야 하는 경우 프록시 서버에서 사용되는 IP 주소 및 포트가 포함된 `HTTP_PROXY` 및 `HTTPS_PROXY` 환경 변수를 구성할 수 있습니다.

Linux, macOS, or Unix

```
export HTTP_PROXY=http://a.b.c.d:n
export HTTPS_PROXY=http://w.x.y.z:m
```

Windows

```
set HTTP_PROXY=http://a.b.c.d:n
set HTTPS_PROXY=http://w.x.y.z:m
```

이 예제에서 `http://a.b.c.d:n` 및 `http://w.x.y.z:m`은 HTTP 및 HTTPS 프록시에 대한 IP 주소 및 포트 번호입니다.

프록시에 인증

AWS CLI는 HTTP 기본 인증을 지원합니다. 다음과 같이 프록시 URL에 사용자 이름 및 암호를 지정합니다.

Linux, macOS, or Unix

```
export HTTP_PROXY=http://username:password@a.b.c.d:n
export HTTPS_PROXY=http://username:password@w.x.y.z:m
```

Windows

```
set HTTP_PROXY=http://username:password@a.b.c.d:n
set HTTPS_PROXY=http://username:password@w.x.y.z:m
```

> **Note**
>
> AWS CLI는 NTLM 프록시를 지원하지 않습니다. NTLM 또는 Kerberos 프록시를 사용하는 경우 Cntlm과 같은 인증 프록시를 통해 연결할 수 있습니다.

EC2 인스턴스에서 프록시 사용

연결된 IAM 역할을 사용하여 시작한 Amazon EC2 인스턴스에서 프록시를 구성하는 경우 인스턴스 메타데이터에 액세스하는 데 사용된 주소를 제외해야 합니다. 이렇게 하려면 `NO_PROXY` 환경 변수를 인스턴스 메타데이터 서비스의 IP 주소 로 설정합니다.

Linux, macOS, or Unix

```
export NO_PROXY=
```

Windows

```
set NO_PROXY=
```

IAM 역할 수임

IAM 역할은 IAM 사용자가 추가(또는 다른) 권한을 얻을 수 있도록 하거나 다른 AWS 계정에서 작업을 수행할 권한을 주는 인증 도구입니다.

`~/.aws/config` 파일에서 역할에 대한 프로필을 정의하여 IAM 역할을 사용하도록 AWS Command Line Interface를 구성할 수 있습니다. 다음은 `marketingadmin` 프로필을 지정하는 명령을 실행할 때 수임되는 `marketingadmin`이라는 역할 프로필을 보여주는 예입니다.

```
[profile marketingadmin]
role_arn = arn:aws:iam::123456789012:role/marketingadmin
source_profile = user1
```

역할은 역할을 수임할 권한이 있는 IAM 사용자 자격 증명이 포함된 별도의 명명된 프로필에 연결되어야 합니다. 앞의 예제에서는 `source-profile` 필드를 통해 `marketingadmin` 프로필이 `user1` 프로필에 연결됩니다. AWS CLI 명령이 `marketingadmin` 프로필을 사용하도록 지정하면 CLI가 연결된 `user1` 프로필에 대한 자격 증명을 자동으로 찾고 이를 사용하여 지정된 IAM 역할에 대한 임시 자격 증명을 요청합니다. 이러한 임시 자격 증명은 CLI 명령을 실행하는 데 사용됩니다. 지정된 역할은 CLI 명령이 실행되도록 허용하는 IAM 권한 정책에 연결되어야 합니다.

단원

- 캐시된 자격 증명 지우기 (p. 29)

역할 구성 및 사용

IAM 역할을 지정하는 프로필을 사용하여 명령을 실행하면 AWS CLI는 원본 프로필의 자격 증명을 사용하여 AWS STS(AWS Security Token Service)를 호출하고 지정된 역할에 대한 임시 자격 증명을 요청합니다. 원본 프로필의 사용자에는 지정된 프로필의 역할에 대한 `sts:assume-role`을 호출할 권한이 있어야 합니다. 이 역할에는 원본 프로필의 사용자가 역할을 수임할 수 있도록 허용하는 신뢰 관계가 있어야 합니다. 역할에 대한 임시 자격 증명을 가져온 다음 사용하는 프로세스는 종종 역할 수임이라고 합니다.

AWS Identity and Access Management 사용 설명서의 IAM 사용자에게 권한을 위임할 역할 생성에 있는 절차를 따라 사용자가 수임하도록 할 수 있는 권한이 있는 새로운 역할을 IAM에 생성할 수 있습니다. 역할과 원본 프로필의 IAM 사용자가 동일한 계정에 있는 경우 역할의 신뢰 관계를 구성할 때 자신의 계정 ID를 입력할 수 있습니다.

역할을 생성한 후 IAM 사용자(또는 AWS 계정의 사용자)가 해당 역할을 수임할 수 있도록 신뢰 관계를 수정합니다. 다음은 123456789012 계정의 관리자가 사용자에게 `sts:assumerole` 권한을 명시적으로 부여한 경우 해당 계정의 IAM 사용자가 역할을 수임할 수 있는 신뢰 관계를 보여주는 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

신뢰 정책은 실제로 권한을 부여하지 않습니다. 계정의 관리자는 정책을 적절한 권한에 연결하여 역할을 수임할 권한을 개별 사용자에게 위임해야 합니다. 다음에 연결된 IAM 사용자가 `marketingadmin` 역할만 수임하도록 하는 예입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/marketingadmin"
    }
  ]
}
```

IAM 사용자는 추가 권한이 없더라도 역할 프로필을 사용하여 CLI 명령을 실행할 수 있습니다. 대신 명령을 실행하는 데 필요한 권한은 역할에 연결된 권한으로부터 나옵니다. 그러나 사용자가 역할을 사용하지 않고 AWS 리소스에 액세스할 수 있도록 하려면 해당 리소스에 대한 권한을 부여하는 추가 인라인 또는 관리형 정책을 IAM 사용자에게 연결해야 합니다.

이제 역할 프로필, 역할 권한, 역할 신뢰 관계 및 사용자 권한이 적절하게 구성되었으므로 `--profile` 옵션을 호출하여 명령줄에서 역할을 사용할 수 있습니다. 예를 들어, 다음 명령은 이 주제의 시작 부분에 있는 예에서 정의된 대로 `marketingadmin` 역할에 연결된 권한을 사용하여 Amazon S3 `ls` 명령을 호출합니다.

```
aws s3 ls --profile marketingadmin
```

여러 호출에 역할을 사용하려면 명령줄에서 현재 세션에 대한 `AWS_PROFILE` 환경 변수를 설정하면 됩니다. 환경 변수를 정의하는 동안 각 명령에서 `--profile` 옵션을 지정할 필요가 없습니다.

Linux, macOS, or Unix

```
export AWS_PROFILE=marketingadmin
```

Windows

```
set AWS_PROFILE=marketingadmin
```

IAM 사용자 및 역할 구성에 대한 자세한 내용은 IAM 사용 설명서의 사용자 및 그룹과 역할을 참조하십시오.

멀티 팩터 인증 사용

보안을 강화하기 위해, 사용자가 역할 프로필을 사용하여 호출을 수행하려고 할 때 멀티 팩터 인증 디바이스, U2F 디바이스 또는 모바일 앱에서 생성된 일회용 키를 제공하도록 사용자에게 요구할 수 있습니다.

먼저, 멀티 팩터 인증을 요구하도록 IAM 역할에 대한 신뢰 관계를 수정합니다. 해당하는 예제는 다음 샘플의 `Condition` 행을 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::123456789012:user/jonsmith" },
      "Action": "sts:AssumeRole",
      "Condition": { "Bool": { "aws:multifactorAuthPresent": true } }
    }
  ]
}
```

다음에는 사용자 MFA 기기의 ARN을 지정하는 줄을 역할 프로필에 추가합니다.

```
[profile marketingadmin]
role_arn = arn:aws:iam::123456789012:role/marketingadmin
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/saanvi
```

`mfa_serial` 설정은 표시된 대로 ARN 또는 하드웨어 MFA 토큰의 일련 번호를 가져올 수 있습니다.

교차 계정 역할

역할을 교차 계정 역할로 구성하면 IAM 사용자가 다른 계정에 속한 역할을 수임할 수 있습니다. IAM 사용자에게 권한을 위임할 역할 생성에 설명된 대로 역할 생성 중에 역할 유형을 다른 AWS 계정으로 설정하고 필요에 따라 MFA 필요를 선택합니다. MFA 필요 옵션은 멀티 팩터 인증 사용 (p. 28)의 설명과 같이 신뢰 관계에서 적절한 조건을 구성합니다.

누가 계정 간의 역할을 수임할 수 있는지에 대한 추가 제어를 제공하기 위해 외부 ID를 사용하는 경우 역할 프로필에 `external_id` 파라미터도 추가해야 합니다. 일반적으로 회사 또는 조직 외부에 있는 사람이 다른 계정을 제어하는 경우에만 이를 사용합니다.

```
[profile crossaccountrole]
role_arn = arn:aws:iam::234567890123:role/SomeRole
```

```
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/saanvi
external_id = 123456
```

캐시된 자격 증명 지우기

역할을 수임하면 AWS CLI는 임시 자격 증명이 만료될 때까지 로컬에서 자격 증명을 캐시해 줍니다. 역할의 임시 자격 증명이 취소되면 캐시를 삭제하여 AWS CLI가 새 자격 증명을 검색하도록 할 수 있습니다.

Linux, macOS, or Unix

```
rm -r ~/.aws/cli/cache
```

Windows

```
del /s /q %UserProfile%\.aws\cli\cache
```

명령 완성

Unix와 같은 시스템에서는 TAB 키를 사용하여 부분적으로 입력한 명령을 완성할 수 있는 명령 완성 기능이 AWS CLI에 포함됩니다. 대부분의 시스템에는 이 기능이 자동으로 설치되지 않으므로 수동으로 기능을 구성해야 합니다.

명령 완성을 구성하려면 사용 중인 셸의 이름과 `aws_completer` 스크립트의 위치라는 두 가지 정보가 있어야 합니다.

Amazon Linux

Amazon Linux를 실행하는 Amazon EC2 인스턴스에서는 명령 완성이 기본적으로 자동으로 구성되고 활성화됩니다.

단원

셸 식별

어떤 셸을 사용하고 있는지 잘 모르면 다음 명령 중 하나를 사용하여 파악할 수 있습니다.

echo $SHELL – 셸의 설치 디렉터리를 표시합니다. 이 항목은 로그인 후 다른 셸을 시작하지 않은 한 일반적으로 사용 중인 셸과 일치합니다.

```
echo $SHELL
/bin/bash
```

ps – 현재 사용자에 대해 실행 중인 프로세스를 표시합니다. 셸은 이 프로세스 중 하나입니다.

```
ps
  PID TTY          TIME CMD
 2148 pts/1    00:00:00 bash
```

```
8756 pts/1    00:00:00 ps
```

AWS Completer 찾기

이 위치는 사용한 설치 방법에 따라 다를 수 있습니다.

패키지 관리자 – `pip`, `yum`, `brew` 및 `apt-get`과 같은 프로그램은 일반적으로 AWS Completer(또는 이 기능에 대한 symlink)를 표준 경로 위치에 설치합니다. 이 경우 `which` 명령은 Completer를 자동으로 찾을 수 있습니다.

```
$ which aws_completer
/usr/local/bin/aws_completer
```

번들 설치 관리자 – 이전 단원의 지침에 따라 번들 설치 관리자를 사용한 경우 AWS Completer는 설치 디렉터리의 `bin` 하위 폴더에 위치하게 됩니다.

```
$ ls /usr/local/aws/bin
activate
activate.csh
activate.fish
activate_this.py
aws
aws.cmd
aws_completer
...
```

다른 모든 방법이 실패하면 `find`를 사용하여 전체 파일 시스템에서 AWS Completer를 검색할 수 있습니다.

```
$ find / -name aws_completer
/usr/local/aws/bin/aws_completer
```

명령 완성 활성화

명령 완성을 활성화하기 위해 명령을 실행합니다. 완료를 활성화하는 데 사용하는 명령은 사용 중인 셸에 따라 다릅니다. 셸의 RC 파일에 명령을 추가하여 새 셸을 열 때마다 실행되도록 할 수 있습니다.

- bash – 내장 명령인 `complete`을 사용합니다.

  ```
  complete -C '/usr/local/bin/aws_completer' aws
  ```

  `~/.bashrc`에 명령을 추가하여 새 셸을 열 때마다 실행되도록 합니다. `~/.bash_profile`은 `~/.bashrc`를 소스로 하여 로그인 셸에서도 이 명령이 실행되도록 합니다.
- tcsh – tcsh에 대한 complete는 완성 동작을 정의하기 위한 단어 유형과 패턴을 가져옵니다.

  ```
  > complete aws 'p/*/`aws_completer`/'
  ```

  `~/.tschrc`에 명령을 추가하여 새 셸을 열 때마다 실행되도록 합니다.
- zsh – 소스 `bin/aws_zsh_completer.sh`.

  ```
  % source /usr/local/bin/aws_zsh_completer.sh
  ```

  AWS CLI는 zsh 지원을 위해 bash 호환성 자동 완성(`bashcompinit`)을 사용합니다. 추가 세부 정보는 `aws_zsh_completer.sh`의 상단을 참조하십시오.

`~/.zshrc`에 명령을 추가하여 새 셸을 열 때마다 실행되도록 합니다.

명령 완성 테스트

명령 완성을 활성화한 후 부분 명령을 입력하고 Tab을 눌러 사용 가능한 명령을 봅니다.

```
aws sTAB
s3              ses             sqs             sts             swf
s3api           sns             storagegateway  support
```

AWS Command Line Interface를 사용하여 Amazon EC2에서 개발 환경 배포

이 자습서에서는 AWS CLI를 사용하여 Amazon EC2에서 개발 환경을 설정하는 방법을 자세히 설명합니다. 여기에는 간단한 버전의 설치 및 구성 지침이 포함되고, Windows 및 에서 시작부터 끝까지 실행할 수 있습니다.

단계

AWS CLI 설치

설치 관리자(Windows)를 사용하거나 Python용 패키지 관리자인 `pip`,를 사용하여 AWS CLI를 설치할 수 있습니다.

Windows

1. MSI 설치 관리자를 다운로드합니다.
   - Windows용 AWS CLI MSI 설치 관리자(64비트) 다운로드
   - Windows용 AWS CLI MSI 설치 관리자(32비트) 다운로드
2. 다운로드한 MSI 설치 관리자를 실행합니다.
3. 다음에 나타나는 지침을 따릅니다.

Linux, macOS, or Unix

이러한 단계를 수행하려면 유효한 Python 2 버전 2.6.5+ 또는 Python 3 버전 3.3+이 설치되어 있어야 합니다. 다음 단계를 사용하는 중에 문제가 발생할 경우 AWS Command Line Interface 사용 설명서의 전체 설치 지침을 참조하십시오.

1. pip 웹 사이트에서 설치 스크립트를 다운로드하여 실행합니다.

```
$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
$ python get-pip.py --user
```

2. `pip`를 사용하여 AWS CLI를 설치하십시오.

```
$ pip install awscli --user
```

AWS CLI 구성

명령줄에서 `aws configure`를 실행하여 자격 증명 및 설정을 설정합니다.

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-2
Default output format [None]: json
```

AWS CLI에서 다음 정보를 묻는 메시지를 표시합니다.

- AWS 액세스 키 ID 및 AWS 보안 액세스 키 – 계정 자격 증명입니다. 키가 없는 경우 Amazon Web Services 일반 참조의 보안 자격 증명을 가져오는 방법을 참조하십시오.
- 기본값 리전 이름 – 기본적으로 호출을 작성할 리전의 이름입니다.
- 기본값 출력 형식 – 이 형식은 json, 텍스트 또는 테이블일 수 있습니다. 출력 형식을 지정하지 않으면 json이 사용됩니다.

명령을 실행하여 자격 증명이 올바르게 구성되었고 AWS에 연결할 수 있는지 확인합니다.

```
$ aws ec2 describe-regions --output table
----------------------------------------------------------
|                    DescribeRegions                     |
+--------------------------------------------------------+
||                        Regions                       ||
|+-----------------------------------+------------------+|
||             Endpoint              |   RegionName     ||
|+-----------------------------------+------------------+|
||  ec2.ap-south-1.amazonaws.com     |  ap-south-1      ||
||  ec2.eu-west-3.amazonaws.com      |  eu-west-3       ||
||  ec2.eu-west-2.amazonaws.com      |  eu-west-2       ||
||  ec2.eu-west-1.amazonaws.com      |  eu-west-1       ||
||  ec2.ap-northeast-3.amazonaws.com |  ap-northeast-3  ||
||  ec2.ap-northeast-2.amazonaws.com |  ap-northeast-2  ||
||  ec2.ap-northeast-1.amazonaws.com |  ap-northeast-1  ||
||  ec2.sa-east-1.amazonaws.com      |  sa-east-1       ||
||  ec2.ca-central-1.amazonaws.com   |  ca-central-1    ||
||  ec2.ap-southeast-1.amazonaws.com |  ap-southeast-1  ||
||  ec2.ap-southeast-2.amazonaws.com |  ap-southeast-2  ||
||  ec2.eu-central-1.amazonaws.com   |  eu-central-1    ||
||  ec2.us-east-1.amazonaws.com      |  us-east-1       ||
||  ec2.us-east-2.amazonaws.com      |  us-east-2       ||
||  ec2.us-west-1.amazonaws.com      |  us-west-1       ||
||  ec2.us-west-2.amazonaws.com      |  us-west-2       ||
|+-----------------------------------+------------------+|
```

EC2 인스턴스에 대한 보안 그룹과 키 페어 생성

다음 단계는 SSH를 사용하여 액세스할 수 있는 EC2 인스턴스를 시작하기 위한 사전 요구사항을 설정하는 것입니다. Amazon EC2 기능에 대한 자세한 내용을 확인하려면 Linux 인스턴스용 Amazon EC2 사용 설명서로 이동하십시오.

보안 그룹, 키 페어, 및 역할을 생성하려면

1. 먼저 새 보안 그룹을 생성하고 SSH용 포트 22에서 수신 트래픽을 허용하는 규칙을 추가합니다. 리전의 기본 VPC를 사용하는 경우 `--vpc-id` 파라미터를 생략해도 됩니다. 기본 VPC가 아닌 경우에는 인스

턴스를 실행할 VPC의 ID를 지정하십시오. 보안을 강화하기 위해 `0.0.0.0/0` CIDR 범위를, 인스턴스에 연결할 네트워크의 범위로 바꾸십시오.

```
$ aws ec2 create-security-group --group-name devenv-sg --vpc-id vpc-xxxxxxxx --
description "security group for development environment"
{
    "GroupId": "sg-b018ced5"
}
$ aws ec2 authorize-security-group-ingress --group-name devenv-sg --protocol tcp --port
 22 --cidr 0.0.0.0/0
```

나중에 인스턴스를 시작할 때 사용하기 위해 보안 그룹 ID를 기록해 둡니다.

2. 그런 다음 인스턴스에 연결할 수 있게 해주는 키 페어를 생성합니다. 이 명령은 키의 콘텐츠를 `devenv-key.pem`이라는 파일에 저장합니다.

```
$ aws ec2 create-key-pair --key-name devenv-key --query 'KeyMaterial' --output text >
 devenv-key.pem
```

Windows가 설치된

Windows 명령 프롬프트에서 쿼리를 작은따옴표 대신 큰따옴표를 사용합니다.

3. 또한 Linux에서는 사용자만 키 파일에 액세스할 수 있도록 파일 모드를 변경해야 합니다.

```
$ chmod 400 devenv-key.pem
```

인스턴스 시작 및 연결

마지막으로 인스턴스를 시작하고 인스턴스에 연결할 수 있도록 준비됩니다.

인스턴스를 시작하고 연결하려면

1. 이전 단계에서 만든 보안 그룹의 ID를 사용하여 다음 명령을 실행합니다. `--image-id` 파라미터는 Amazon EC2가 인스턴스를 부트스트랩하는 데 사용하는 Amazon 머신 이미지(AMI)를 지정합니다. Amazon EC2 콘솔을 사용하여 해당 리전 및 운영 체제의 이미지 ID를 확인할 수 있습니다. 기본 VPC의 기본 서브넷을 사용하는 경우 `--subnet-id` 파라미터를 생략해도 됩니다. 그렇지 않은 경우에는 인스턴스를 실행할 서브넷의 ID를 지정하십시오.

```
$ aws ec2 run-instances --image-id ami-xxxxxxxx --subnet-id subnet-xxxxxxxx --security-
group-ids sg-b018ced5 --count 1 --instance-type t2.micro --key-name devenv-key --query
 'Instances[0].InstanceId'
"i-0787e4282810ef9cf"
```

2. 인스턴스를 시작하는 데 몇 분 정도 걸립니다. 인스턴스가 시작되어 실행되면 연결할 인스턴스의 퍼블릭 IP 주소가 필요합니다. 다음 명령을 사용하여 퍼블릭 IP 주소를 확인합니다.

```
$ aws ec2 describe-instances --instance-ids i-0787e4282810ef9cf --query
 'Reservations[0].Instances[0].PublicIpAddress'
"54.183.22.255"
```

3. 인스턴스에 연결하려면 기본 설정된 터미널 프로그램과 함께 퍼블릭 IP 주소 및 프라이빗 키를 사용합니다. Linux, macOS, or Unix에서 다음 명령을 사용하여 명령줄에서 이를 수행할 수 있습니다.

```
$ ssh -i devenv-key.pem user@54.183.22.255
```

인스턴스에 연결을 시도할 때 권한 거부(퍼블릭 키) 같은 오류가 발생하면 다음이 올바른지 확인합니다.

- 키 – 지정된 키는 표시된 경로에 있어야 하고 퍼블릭 키가 아니라 프라이빗 키여야 합니다. 키에 대한 권한은 소유자로 제한되어야 합니다.
- 사용자 – 이 사용자 이름은 인스턴스를 실행하는 데 사용한 AMI와 연결된 기본 사용자 이름과 일치해야 합니다. Ubuntu AMI의 경우 `ubuntu`입니다. Amazon Linux AMI의 경우 `ec2-user`입니다.
- 인스턴스 – 인스턴스의 퍼블릭 IP 주소 또는 DNS 이름입니다. 주소가 퍼블릭 주소인지, 그리고 포트 22가 인스턴스 보안 그룹의 로컬 시스템에 열려 있는지 확인합니다.

`-v` 옵션을 사용하여 오류와 관련된 추가 정보를 볼 수도 있습니다.

Windows의 SSH

Windows에서는 여기에서 사용 가능한 PuTTY 터미널 애플리케이션을 사용할 수 있습니다. 다운로드 페이지에서 `putty.exe` 및 `puttygen.exe`를 가져오십시오.
`puttygen.exe`를 사용하여 프라이빗 키를 PuTTY에 필요한 `.ppk` 파일로 변환하십시오.
`putty.exe`를 시작한 다음 [Host Name] 필드에 인스턴스의 퍼블릭 IP 주소를 입력하고, 연결 유형을 SSH로 설정하십시오.
[Category] 패널에서 [Connection] > [SSH] > [Auth]로 이동하고 [Browse]를 클릭하여 `.ppk` 파일을 선택한 다음 [Open]을 클릭하여 연결하십시오.

4. 터미널에서 서버의 퍼블릭 키를 수락하라는 메시지를 표시합니다. `yes`를 입력하고 [Enter]를 클릭하여 연결을 완료합니다.

이제 보안 그룹을 구성하고, 키 페어를 생성하며, EC2 인스턴스를 시작하고, 명령줄을 그만두지 않고도 연결하게 되었습니다.

AWS Command Line Interface 사용

이 섹션에서는 AWS Command Line Interface에서 사용되는 일반적인 기능 및 호출 패턴을 소개합니다.

> **Note**
>
> AWS CLI는 HTTPS를 통해 서비스에 대한 API 호출을 수행합니다. 호출을 수행하려면 TCP 포트 443에서 아웃바운드 연결을 활성화해야 합니다.

항목

AWS Command Line Interface를 사용하여 도움말 보기

AWS CLI,를 사용할 때 도움말을 보려면 명령 끝에 `help`를 추가하면 됩니다. 예를 들어, 다음 명령은 일반 AWS CLI 옵션에 대한 도움말과 사용 가능한 최상위 명령을 나열합니다.

```
$ aws help
```

다음 명령은 Amazon EC2에 사용 가능한 하위 명령을 나열합니다.

```
$ aws ec2 help
```

다음 예제에서는 입력 매개 변수, 필터 및 출력을 포함하여 EC2 `DescribeInstances` 작업에 대한 상세 도움말을 나열합니다. 명령을 표시하는 방법을 모를 경우 도움말의 예제 단원을 확인하십시오.

```
$ aws ec2 describe-instances help
```

각 명령에 대한 도움말은 다음과 같은 6개 섹션으로 나뉩니다.

이름 –명령의 이름입니다.

```
NAME
       describe-instances -
```

설명 – 명령이 호출하는 API 작업에 대한 설명으로 명령 서비스에 대한 API 설명서에서 가져옵니다.

```
DESCRIPTION
```

```
       Describes one or more of your instances.

       If you specify one or more instance IDs, Amazon EC2 returns information
       for those instances. If you do not specify instance IDs, Amazon EC2
       returns information for all relevant instances. If you specify an
       instance ID that is not valid, an error is returned. If you specify an
       instance that you do not own, it is not included in the returned
       results.
...
```

시놉시스 – 명령 및 해당 옵션의 목록입니다. 옵션을 대괄호로 표시할 경우 선택적인 옵션이거나, 기본값을 가지거나, 대신 사용할 수 있는 대체 옵션을 갖고 있습니다.

```
SYNOPSIS
            describe-instances
          [--dry-run | --no-dry-run]
          [--instance-ids <value>]
          [--filters <value>]
          [--cli-input-json <value>]
          [--starting-token <value>]
          [--page-size <value>]
          [--max-items <value>]
          [--generate-cli-skeleton]
```

`describe-instances`에는 현재 계정 및 리전의 모든 인스턴스를 설명하는 기본 동작이 있습니다. 1개 이상의 인스턴스를 설명할 `instance-ids` 목록을 선택적으로 지정할 수 있습니다. `dry-run`은 값을 가지지 않는 선택 사항인 부울 플래그입니다. 부울 플래그를 사용하려면 표시된 값을 지정합니다. 이 경우 `--dry-run` 또는 `--no-dry-run`입니다. 마찬가지로 `--generate-cli-skeleton`도 값을 갖고 있지 않습니다. 옵션 사용 시 조건이 있는 경우 해당 조건을 `OPTIONS` 섹션에서 설명하거나 예제에 표시해야 합니다.

옵션 – 개요에 표시된 각 옵션에 대한 설명입니다.

```
OPTIONS
       --dry-run | --no-dry-run (boolean)
          Checks whether you have the required permissions for the action,
          without actually making the request, and provides an error response.
          If you have the required permissions, the error response is DryRun-
          Operation . Otherwise, it is UnauthorizedOperation .

       --instance-ids (list)
          One or more instance IDs.

          Default: Describes all your instances.
...
```

예제 – 명령 및 해당 옵션의 사용을 보여 주는 예제입니다. 필요한 명령 또는 사용 사례에 대해 사용 가능한 예제가 없는 경우 이 페이지의 피드백 링크를 사용하여 요청하거나 명령의 도움말 페이지에 있는 AWS CLI 명령 참조에서 요청할 수 있습니다.

```
    EXAMPLES
    To describe an Amazon EC2 instance

    Command:

    aws ec2 describe-instances --instance-ids i-5203422c

    To describe all instances with the instance type m1.small

    Command:

    aws ec2 describe-instances --filters "Name=instance-type,Values=m1.small"
```

```
    To describe all instances with a Owner tag

    Command:

    aws ec2 describe-instances --filters "Name=tag-key,Values=Owner"
...
```

출력 – AWS의 응답에서 반환되는 각 필드 및 데이터 형식에 대한 설명입니다.

`describe-instances`의 경우 출력은 예약 객체의 목록이며, 각 객체에는 연관된 인스턴스에 대한 정보를 포함하는 여러 필드 및 객체가 포함됩니다. 이 정보는 Amazon EC2에서 사용한 예약 데이터 형식에 대한 API 설명서에서 가져옵니다.

```
OUTPUT
       Reservations -> (list)
          One or more reservations.

          (structure)
              Describes a reservation.

              ReservationId -> (string)
                 The ID of the reservation.

              OwnerId -> (string)
                 The ID of the AWS account that owns the reservation.

              RequesterId -> (string)
                 The ID of the requester that launched the instances  on  your
                 behalf (for example, AWS Management Console or Auto Scaling).

              Groups -> (list)
                 One or more security groups.

                 (structure)
                     Describes a security group.

                     GroupName -> (string)
                        The name of the security group.

                     GroupId -> (string)
                        The ID of the security group.

              Instances -> (list)
                 One or more instances.

                 (structure)
                     Describes an instance.

                     InstanceId -> (string)
                        The ID of the instance.

                     ImageId -> (string)
                        The ID of the AMI used to launch the instance.

                     State -> (structure)
                        The current state of the instance.

                        Code -> (integer)
                            The  low  byte represents the state. The high byte
                            is an opaque internal value and should be ignored.
...
```

AWS CLI에 의해 출력이 JSON으로 렌더링될 경우 다음과 같은 예약 객체의 배열이 됩니다.

```
{
    "Reservations": [
        {
            "OwnerId": "012345678901",
            "ReservationId": "r-4c58f8a0",
            "Groups": [],
            "RequesterId": "012345678901",
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "ec2-52-74-16-12.us-west-2.compute.amazonaws.com",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
...
```

각 예약 객체에는 예약을 설명하는 필드와 인스턴스 객체의 배열이 포함됩니다. 각 배열에는 고유한 필드(예: `PublicDnsName`) 및 이를 설명하는 객체(예: `State`)가 있습니다.

Windows 사용자

도움말 파일을 한 번에 한 페이지씩 보려면 help 명령의 출력을 `more`에 파이프합니다. 문서를 더 보려면 스페이스바 또는 Page Down 키를 누르고, 중지하려면 `q`를 누릅니다.

```
> aws ec2 describe-instances help | more
```

AWS CLI 문서

AWS CLI Command Reference에서는 모든 AWS CLI 명령의 도움말 파일의 콘텐츠를 제공합니다. 이 콘텐츠는 쉽게 탐색하고 모바일, 태블릿 및 데스크톱 화면에서 볼 수 있도록 컴파일되어 온라인으로 제공됩니다.

도움말 파일에 명령줄 보기에서 보거나 따라갈 수 없는 링크가 포함되는 경우가 있습니다. 이 링크는 온라인 AWS CLI 참조에서 유지됩니다.

API 설명서

AWS CLI의 모든 하위 명령은 서비스의 퍼블릭 API에 대한 호출에 해당합니다. 퍼블릭 API를 사용하는 각 서비스에는 AWS 설명서 웹 사이트의 해당 서비스 홈 페이지에서 찾을 수 있는 API 참조 설명서 세트가 있습니다.

API 참조 콘텐츠는 API 구성 방법 및 사용되는 프로토콜에 따라 다릅니다. 일반적으로 API 참조에는 API에서 지원하는 작업에 대한 세부 정보, 서비스와 주고 받은 데이터 및 가능한 오류 조건이 포함됩니다.

API 설명서 섹션

- 작업 – 파라미터(길이 및 콘텐츠에 대한 제약 등)에 대한 세부 정보 및 작업 관련 오류입니다. 작업은 AWS CLI의 하위 명령에 해당합니다.
- 데이터 유형 – 하위 명령에서 반환한 객체 데이터에 대한 추가 정보를 포함할 수 있습니다.
- 범용 파라미터 – 서비스의 모든 작업에서 사용하는 파라미터에 대한 세부 정보입니다.
- 범용 오류 – 모든 서비스 작업에서 반환한 오류에 대한 세부 정보입니다.

각 섹션의 이름 및 가용성은 서비스에 따라 다를 수 있습니다.

서비스별 CLI

모든 서비스에서 작동하는 단일 AWS CLI가 작성되기 전에 일부 서비스에 별도의 AWS CLI가 있습니다. 이 서비스별 CLI에는 해당 서비스의 설명서 페이지에서 링크되는 별도의 설명서가 있습니다. 서비스별 CLI에 대한 설명서는 AWS CLI에 적용되지 않습니다.

AWS Command Line Interface의 명령 구조

AWS CLI는 명령줄에서 멀티파트 구조를 사용합니다. 이 구조는 `aws`에 대한 기본 호출로 시작됩니다. 다음 부분은 최상위 명령을 지정합니다. 이 명령은 주로 AWS CLI에서 지원되는 AWS 서비스를 나타냅니다. 각 AWS 서비스에는 수행할 작업을 지정하는 추가 하위 명령이 있습니다. 명령줄에서 작업에 대한 일반 CLI 옵션 또는 특정 파라미터를 어떤 순서로든 지정할 수 있습니다. 독점적인 파라미터를 여러 번 지정하면 마지막 값만 적용됩니다.

```
$ aws <command> <subcommand> [options and parameters]
```

파라미터는 숫자, 문자열, 목록, 맵, JSON 구조와 같은 다양한 유형의 입력 값을 가져올 수 있습니다.

AWS Command Line Interface에 대한 파라미터 값 지정

대부분의 파라미터는 간단한 문자열 또는 숫자 값입니다(예: 다음 예제의 `my-key-pair` 키 페어 이름).

```
$ aws ec2 create-key-pair --key-name my-key-pair
```

공백 문자가 없는 문자열은 따옴표로 묶거나 묶지 않을 수 있습니다. 그러나 하나 이상의 공백 문자가 포함된 문자열은 따옴표로 묶어야 합니다. 다음 예제에 표시된 것처럼 Linux, macOS, or Unix 및 Windows PowerShell에서는 작은따옴표(')를 사용하고, Windows 명령 프롬프트에서는 큰따옴표(")를 사용하십시오.

Windows PowerShell, Linux, macOS, or Unix

```
$ aws ec2 create-key-pair --key-name 'my key pair'
```

Windows 명령 처리기

```
> aws ec2 create-key-pair --key-name "my key pair"
```

공백 대신 등호를 사용할 수도 있습니다. 일반적으로 파라미터 값이 하이픈으로 시작되는 경우에만 필요합니다.

```
$ aws ec2 delete-key-pair --key-name=-mykey
```

항목

공통 파라미터 유형

이 단원에서는 몇 가지 공통 파라미터 유형과 서비스에서 파라미터가 부합해야 하는 형식에 대해 설명합니다. 특정 명령에 대한 파라미터의 형식 지정에 문제가 있는 경우, 명령 이름 다음에 **help**를 입력하여 매뉴얼을 점검하십시오. 예를 들면, 다음과 같습니다.

```
$ aws ec2 describe-spot-price-history help
```

각 하위 명령의 도움말에는 해당 기능, 옵션, 출력 및 예제가 설명되어 있습니다. 이 옵션 섹션에는 각 옵션의 이름 및 설명이 포함되고 괄호 안에는 옵션의 파라미터 유형이 있습니다.

문자열 – 문자열 파라미터에는 영숫자 문자, 기호 및 ASCII 문자 세트의 공백이 포함될 수 있습니다. 공백이 포함된 문자열은 따옴표로 묶어야 합니다. 표준 공백 문자 이외의 공백 및 기호를 사용하는 것은 권장하지 않습니다. 를 사용할 때 문제를 일으킬 수 있습니다.

일부 문자열 파라미터는 파일의 이진 데이터를 허용할 수 있습니다. 예제는 이진 파일 (p. 44) 단원을 참조하십시오.

타임스탬프 – 타임스탬프는 ISO 8601 표준에 따라 형식이 지정됩니다. 이를 "DateTime" 또는 "Date" 형식 파라미터라고도 합니다.

```
$ aws ec2 describe-spot-price-history --start-time 2014-10-13T19:00:00Z
```

허용 가능한 형식은 다음과 같습니다.

- YYYY-MM-DDThh:mm:ss.sssTZD(UTC), 예: 2014-10-01T20:30:00.000Z
- YYYY-MM-DDThh:mm:ss.sssTZD(오프셋 포함), 예: 2014-10-01T12:30:00.000-08:00
- YYYY-MM-DD, 예: 2014-10-01
- 초 단위의 Unix 시간, 예: 1412195400

목록 – 공백으로 구분된 하나 이상의 문자열입니다.

```
$ aws ec2 describe-spot-price-history --instance-types m1.xlarge m1.medium
```

부울 – 옵션을 켜거나 끄는 이진 플래그입니다. 예를 들어, `ec2 describe-spot-price-history`에는 지정할 경우 실제 쿼리는 실행하지 않고 서비스에 대해 명령을 검증하는 테스트 실행 부울 파라미터가 있습니다.

```
$ aws ec2 describe-spot-price-history --dry-run
```

출력은 명령이 제대로 구성되었는지 여부를 나타냅니다. 또한 이 명령에는 기본 동작이기 때문에 반드시 포함할 필요는 없지만 명령을 정상적으로 실행해야 함을 명시적으로 표시하는 데 사용할 수 있는 no-dry-run 버전의 파라미터도 포함됩니다.

정수 – 부호가 없는 정수입니다.

```
$ aws ec2 describe-spot-price-history --max-items 5
```

Blob – 이진수 객체입니다. BLOB 파라미터는 이진 데이터를 포함하는 로컬 파일에 대한 경로를 갖고 있습니다. 경로에는 `http://` 또는 `file://` 같은 프로토콜 식별자가 포함되지 않아야 합니다.

`aws s3api put-object`의 `--body` 파라미터는 BLOB입니다.

```
$  aws s3api put-object --bucket my-bucket --key testimage.png --body /tmp/image.png
```

맵 – JSON 또는 간편 구문 (p. 54)으로 지정된 일련의 키 값 페어입니다. 다음 예제에서는 `--key` 맵 파라미터가 포함된 my-table이라는 DynamoDB 테이블에서 항목을 읽습니다. 파라미터는 중첩된 JSON 구조로 1 숫자 값을 가진 id라는 기본 키를 지정합니다.

```
$ aws dynamodb get-item --table-name my-table --key '{"id": {"N":"1"}}'
{
    "Item": {
        "name": {
            "S": "John"
        },
        "id": {
            "N": "1"
        }
    }
}
```

다음 단원에서는 JSON 인수를 더 자세히 설명합니다.

파라미터에 JSON 사용

JSON은 복잡한 명령줄 파라미터를 지정하는 데 유용합니다. 예를 들어, 다음 명령은 `m1.small` 가용 영역에도 있는 `m1.medium` 또는 `us-west-2c` 인스턴스 유형을 가진 모든 EC2 인스턴스를 나열합니다.

```
$ aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro,m1.medium"
 "Name=availability-zone,Values=us-west-2c"
```

다음 예제에서는 JSON 배열에 동등한 필터 목록을 지정합니다. 대괄호를 사용하여 쉼표로 구분된 JSON 객체 배열을 생성합니다. 각 객체는 쉼표로 구분된 키-값 페어 목록입니다("Name" 및 "Values"는 모두 이 인스턴스에서 키입니다).

"Values" 키 오른쪽에 있는 값은 배열입니다. 배열에 값 문자열이 하나만 포함된 경우에도 필수입니다.

```
[
  {
    "Name": "instance-type",
    "Values": ["t2.micro", "m1.medium"]
  },
  {
    "Name": "availability-zone",
    "Values": ["us-west-2c"]
  }
]
```

한편 가장 바깥쪽에 있는 괄호는 둘 이상의 필터를 지정한 경우에만 필요합니다. 위 명령의 단일 필터 버전(JSON으로 형식 지정)은 다음과 같습니다.

```
$ aws ec2 describe-instances --filters '{"Name": "instance-type", "Values": ["t2.micro",
 "m1.medium"]}'
```

일부 작업에서는 데이터 형식을 JSON으로 지정해야 합니다. 예를 들어, `--block-device-mappings` 명령의 `ec2 run-instances` 파라미터에 파라미터를 전달하려면 블록 디바이스 정보의 형식을 JSON으로 지정해야 합니다.

이 예제에서는 시작 인스턴스의 `/dev/sdb`에서 단일 20GiB Elastic Block Store(EBS) 디바이스가 매핑되도록 지정하는 JSON을 보여 줍니다.

```
{
  "DeviceName": "/dev/sdb",
  "Ebs": {
    "VolumeSize": 20,
    "DeleteOnTermination": false,
    "VolumeType": "standard"
  }
}
```

여러 객체를 연결하려면 다음 예제에서와 같은 배열로 객체를 나열합니다.

```
[
  {
    "DeviceName": "/dev/sdb",
    "Ebs": {
      "VolumeSize": 20,
      "DeleteOnTermination": false,
      "VolumeType": "standard"
    }
  },
  {
    "DeviceName": "/dev/sdc",
    "Ebs": {
      "VolumeSize": 10,
      "DeleteOnTermination": true,
      "VolumeType": "standard"
    }
  }
]
```

명령줄에 JSON을 직접 입력하거나(인용 문자열 (p. 43) 참조) 명령줄에서 참조하는 파일에 저장할 수 있습니다(파일에서 파라미터 로드 (p. 44) 참조).

대용량 데이터 블록을 전달하는 경우 JSON을 파일에 저장한 다음 명령줄에서 참조하는 것이 더 쉽다는 것을 알 수 있습니다. 파일의 JSON 데이터를 보다 쉽게 읽고, 편집 및 공유할 수 있습니다. 이 방법은 다음 단원에서 설명합니다.

JSON에 대한 자세한 내용은 Wikipedia - JSON 및 RFC4627 - The application/json Media Type for JSON을 참조하십시오.

인용 문자열

명령 줄에서 JSON 형식의 파라미터를 입력하는 방법은 운영 체제에 따라 다릅니다. Linux, macOS, or Unix 및 Windows PowerShell은 작은 따옴표(')를 사용하여 다음 예제와 같이 JSON 데이터 구조를 묶습니다.

```
$ aws ec2 run-instances --image-id ami-12345678 --block-device-mappings '[{"DeviceName":"/
dev/sdb","Ebs":{"VolumeSize":20,"DeleteOnTermination":false,"VolumeType":"standard"}}]'
```

반면, Windows 명령 프롬프트에서는 큰따옴표(")를 사용하여 JSON 데이터 구조를 묶습니다. 또한 다음 예제와 같이 JSON 데이터 구조 내 큰따옴표(")마다 백슬래시(\) 이스케이프 문자가 필요합니다.

```
> aws ec2 run-instances --image-id ami-12345678 --block-device-mappings "[{\"DeviceName
\":\"/dev/sdb\",\"Ebs\":{\"VolumeSize\":20,\"DeleteOnTermination\":false,\"VolumeType\":
\"standard\"}}]"
```

다음 예제와 같이 Windows PowerShell에서는 JSON 구조 내에서 각 큰따옴표(")를 이스케이프하는 백슬래시(\)뿐만 아니라 JSON 데이터 구조를 묶는 작은따옴표(')도 필요합니다.

```
> aws ec2 run-instances --image-id ami-12345678 --block-device-mappings '[{\"DeviceName
\":\"/dev/sdb\",\"Ebs\":{\"VolumeSize\":20,\"DeleteOnTermination\":false,\"VolumeType\":
\"standard\"}}]'
```

파라미터 값이 JSON 문서일 경우 포함된 JSON 문서에서 따옴표를 이스케이프합니다. 예를 들어, `attribute`의 `aws sqs create-queue` 파라미터는 `RedrivePolicy` 키를 가질 수 있습니다. `RedrivePolicy`의 값은 이스케이프해야 하는 JSON 문서입니다.

```
$ aws sqs create-queue --queue-name my-queue --
attributes '{ "RedrivePolicy":"{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
west-2:0123456789012:deadletter\", \"maxReceiveCount\":\"5\"}"}'
```

파일에서 파라미터 로드

명령줄에서 JSON 문자열을 이스케이프해야 할 필요성을 방지하려면 파일에서 JSON을 로드합니다. 다음 예제와 같이 `file://` 접두사로 파일 경로를 제공하여 로컬 파일에서 파라미터를 로드합니다.

Linux, macOS, or Unix

```
// Read from a file in the current directory
$ aws ec2 describe-instances --filters file://filter.json

// Read from a file in /tmp
$ aws ec2 describe-instances --filters file:///tmp/filter.json
```

Windows

```
// Read from a file in C:\temp
> aws ec2 describe-instances --filters file://C:\temp\filter.json
```

`file://` 접두사 옵션은 '~/', './', '../'를 포함한 Unix 스타일의 확장을 지원합니다. Windows에서는 '~/' 표현식이 %USERPROFILE% 환경 변수에 저장된 사용자 디렉터리로 확장합니다. 예를 들어, Windows 7에서는 일반적으로 `C:\Users\`*`User Name`*`\` 아래에 사용자 디렉터리가 있습니다.

파라미터 키 값으로 제공되는 JSON 문서는 계속 이스케이프해야 합니다.

```
$ aws sqs create-queue --queue-name my-queue --attributes file://attributes.json
```

attributes.json

```
{
  "RedrivePolicy":"{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
west-2:0123456789012:deadletter\", \"maxReceiveCount\":\"5\"}"
}
```

이진 파일

이진 데이터를 파라미터로 갖고 있는 명령의 경우 `fileb://` 접두사를 사용하여 데이터가 이진 콘텐츠임을 지정합니다. 이진 데이터를 수락하는 명령은 다음과 같습니다.

- **`aws ec2 run-instances`** – `--user-data` 파라미터.
- **`aws s3api put-object`** – `--sse-customer-key` 파라미터.
- **`aws kms decrypt`** – `--ciphertext-blob` 파라미터.

다음 예제에서는 Linux 명령줄 도구를 사용하여 이진 256비트 AES 키를 생성한 다음, 이를 Amazon S3에 제공하여 업로드된 파일 서버측을 암호화합니다.

```
$ dd if=/dev/urandom bs=1 count=32 > sse.key
32+0 records in
32+0 records out
32 bytes (32 B) copied, 0.000164441 s, 195 kB/s
$ aws s3api put-object --bucket my-bucket --key test.txt --body test.txt --sse-customer-key
 fileb://sse.key --sse-customer-algorithm AES256
{
    "SSECustomerKeyMD5": "iVg8oWa8sy714+FjtesrJg==",
    "SSECustomerAlgorithm": "AES256",
    "ETag": "\"a6118e84b76cf98bf04bbe14b6045c6c\""
}
```

원격 파일

또한 AWS CLI는 `http://` 또는 `https://` URL을 사용하여 인터넷에서 호스트되는 파일에서 파라미터 로드를 지원합니다. 다음 예제에서는 버킷의 파일을 참조합니다. 이렇게 하면 모든 컴퓨터에서 파라미터 파일에 액세스할 수 있지만, 공개적으로 액세스 가능한 위치에 파일을 저장해야 합니다.

```
$ aws ec2 run-instances --image-id ami-12345678 --block-device-mappings http://my-
bucket.s3.amazonaws.com/filename.json
```

위의 예제에서는 `filename.json` 파일에 다음 JSON 데이터가 포함되어 있습니다.

```
[
  {
    "DeviceName": "/dev/sdb",
    "Ebs": {
      "VolumeSize": 20,
      "DeleteOnTermination": false,
      "VolumeType": "standard"
    }
  }
]
```

더 복잡한 JSON 형식의 파라미터를 포함하는 파일을 참조하는 또 다른 예제는 IAM 사용자에 대한 IAM 정책 설정 (p. 75) 항목을 참조하십시오.

CLI 스켈레톤 및 CLI 입력 JSON 파라미터 생성

대부분의 AWS CLI 명령은 파라미터를 JSON으로 저장하고 파라미터를 명령 줄에 입력하는 대신 파일에서 읽기 위해 사용할 수 있는 `--generate-cli-skeleton` 및 `--cli-input-json` 파라미터를 지원합니다.

연산에 지정할 수 있는 모든 파라미터를 대략적으로 설명하는 CLI 스켈레톤 출력 JSON을 생성합니다.

aws ec2 run-instances으로 --generate-cli-skeleton을 사용하려면

1. run-instances 명령을 `--generate-cli-skeleton` 옵션과 함께 실행하여 JSON 스켈레톤을 보십시오.

   ```
   $ aws ec2 run-instances --generate-cli-skeleton
   {
       "DryRun": true,
       "ImageId": "",
       "MinCount": 0,
   ```

```
"MaxCount": 0,
"KeyName": "",
"SecurityGroups": [
    ""
],
"SecurityGroupIds": [
    ""
],
"UserData": "",
"InstanceType": "",
"Placement": {
    "AvailabilityZone": "",
    "GroupName": "",
    "Tenancy": ""
},
"KernelId": "",
"RamdiskId": "",
"BlockDeviceMappings": [
    {
        "VirtualName": "",
        "DeviceName": "",
        "Ebs": {
            "SnapshotId": "",
            "VolumeSize": 0,
            "DeleteOnTermination": true,
            "VolumeType": "",
            "Iops": 0,
            "Encrypted": true
        },
        "NoDevice": ""
    }
],
"Monitoring": {
    "Enabled": true
},
"SubnetId": "",
"DisableApiTermination": true,
"InstanceInitiatedShutdownBehavior": "",
"PrivateIpAddress": "",
"ClientToken": "",
"AdditionalInfo": "",
"NetworkInterfaces": [
    {
        "NetworkInterfaceId": "",
        "DeviceIndex": 0,
        "SubnetId": "",
        "Description": "",
        "PrivateIpAddress": "",
        "Groups": [
            ""
        ],
        "DeleteOnTermination": true,
        "PrivateIpAddresses": [
            {
                "PrivateIpAddress": "",
                "Primary": true
            }
        ],
        "SecondaryPrivateIpAddressCount": 0,
        "AssociatePublicIpAddress": true
    }
],
"IamInstanceProfile": {
    "Arn": "",
    "Name": ""
},
```

```
    "EbsOptimized": true
}
```

2. 출력을 파일로 전달하여 스켈레톤을 로컬로 저장합니다.

```
$ aws ec2 run-instances --generate-cli-skeleton > ec2runinst.json
```

3. 텍스트 편집기에서 스켈레톤을 열고 사용하지 않는 파라미터를 제거합니다.

```
{
    "DryRun": true,
    "ImageId": "",
    "KeyName": "",
    "SecurityGroups": [
        ""
    ],
    "InstanceType": "",
    "Monitoring": {
        "Enabled": true
    }
}
```

EC2의 테스트 실행 기능을 사용하려면 `DryRun` 파라미터를 true로 설정합니다. 이렇게 하면 리소스를 생성하지 않고 구성을 테스트할 수 있습니다.

4. 인스턴스 유형, 키 이름, 보안 그룹 및 기본 리전의 AMI를 입력합니다. 이 예제에서 `ami-dfc39aef`는 `us-west-2` 리전의 64비트 Amazon Linux 이미지입니다.

```
{
    "DryRun": true,
    "ImageId": "ami-dfc39aef",
    "KeyName": "mykey",
    "SecurityGroups": [
        "my-sg"
    ],
    "InstanceType": "t2.micro",
    "Monitoring": {
        "Enabled": true
    }
}
```

5. `file://` 접두사를 사용하여 JSON 구성을 `--cli-input-json` 파라미터에 전달합니다.

```
$ aws ec2 run-instances --cli-input-json file://ec2runinst.json
A client error (DryRunOperation) occurred when calling the RunInstances operation:
 Request would have succeeded, but DryRun flag is set.
```

테스트 실행 오류는 JSON이 올바르게 구성되었으며 파라미터 값이 유효함을 나타냅니다. 출력에서 다른 문제가 보고되는 경우 문제를 수정하고 테스트 실행 오류가 표시될 때까지 위의 단계를 반복합니다.

6. 테스트 실행 기능을 비활성화하려면 DryRun 파라미터를 false로 설정합니다.

```
{
    "DryRun": false,
    "ImageId": "ami-dfc39aef",
    "KeyName": "mykey",
    "SecurityGroups": [
        "my-sg"
    ],
    "InstanceType": "t2.micro",
    "Monitoring": {
        "Enabled": true
```

```
    }
}
```

7. `run-instances` 명령을 다시 실행하여 인스턴스를 시작합니다.

```
$ aws ec2 run-instances --cli-input-json file://ec2runinst.json
{
    "OwnerId": "123456789012",
    "ReservationId": "r-d94a2b1",
    "Groups": [],
    "Instances": [
...
```

AWS Command Line Interface의 명령 출력 제어

이 섹션에서는 AWS CLI의 출력을 제어할 수 있는 다양한 방법을 설명합니다.

항목

출력 형식을 선택하는 방법

AWS CLI는 세 가지 출력 형식을 지원합니다.

- JSON(json)
- 탭으로 구분된 텍스트(텍스트)
- ASCII 형식의 테이블(테이블)

구성 (p. 18) 주제에 설명된 대로 다음 세 가지 방법으로 출력 형식을 지정할 수 있습니다.

- 구성 파일에서 `output` 옵션 사용. 다음 예제에서는 출력을 `text`로 설정합니다.

```
[default]
output=text
```

- `AWS_DEFAULT_OUTPUT` 환경 변수 사용. 예:

```
$ export AWS_DEFAULT_OUTPUT="table"
```

- 명령줄에서 `--output` 옵션 사용. 예:

```
$ aws swf list-domains --registration-status REGISTERED --output text
```

Note

여러 방법으로 출력 형식을 지정한 경우 일반적인 AWS CLI 우선순위 규칙 (p. 20)이 적용됩니다. 예를 들어, `AWS_DEFAULT_OUTPUT` 환경 변수 사용은 `output`을 사용하여 config 파일에서 설정한 값

을 재정의하고, --output을 사용하여 AWS CLI 명령에 전달한 값은 환경 또는 config 파일에서 설정한 값을 재정의합니다.

JSON은 다양한 언어 또는 jq(명령줄 JSON 처리기)를 통해 프로그래밍 방식으로 출력을 처리하는 데 가장 적합합니다. 테이블 형식은 사람이 쉽게 읽을 수 있으며, 텍스트 형식은 sed, grep 및 awk와 같은 기존 Unix 텍스트 처리 도구 및 Windows PowerShell 스크립트에 효과적입니다.

--query 옵션을 사용하여 출력을 필터링하는 방법

AWS CLI는 --query 옵션을 사용하는 내장 출력 필터링 기능을 제공합니다. 작동 방식을 보여 주기 위해 먼저 아래와 같은 기본 JSON 출력으로 시작합니다. 여기서는 별도의 EC2 인스턴스에 연결된 두 개의 EBS(Elastic Block Storage) 볼륨을 설명합니다.

```
$ aws ec2 describe-volumes
{
    "Volumes": [
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-17T00:55:03.000Z",
                    "InstanceId": "i-a071c394",
                    "VolumeId": "vol-e11a5288",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-e11a5288",
            "State": "in-use",
            "SnapshotId": "snap-f23ec1c8",
            "CreateTime": "2013-09-17T00:55:03.000Z",
            "Size": 30
        },
        {
            "AvailabilityZone": "us-west-2a",
            "Attachments": [
                {
                    "AttachTime": "2013-09-18T20:26:16.000Z",
                    "InstanceId": "i-4b41a37c",
                    "VolumeId": "vol-2e410a47",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "VolumeType": "standard",
            "VolumeId": "vol-2e410a47",
            "State": "in-use",
            "SnapshotId": "snap-708e8348",
            "CreateTime": "2013-09-18T20:26:15.000Z",
            "Size": 8
        }
    ]
}
```

먼저 다음 명령을 사용하여 Volumes 목록의 첫 번째 볼륨만 표시할 수 있습니다.

```
$ aws ec2 describe-volumes --query 'Volumes[0]'
{
```

```
    "AvailabilityZone": "us-west-2a",
    "Attachments": [
        {
            "AttachTime": "2013-09-17T00:55:03.000Z",
            "InstanceId": "i-a071c394",
            "VolumeId": "vol-e11a5288",
            "State": "attached",
            "DeleteOnTermination": true,
            "Device": "/dev/sda1"
        }
    ],
    "VolumeType": "standard",
    "VolumeId": "vol-e11a5288",
    "State": "in-use",
    "SnapshotId": "snap-f23ec1c8",
    "CreateTime": "2013-09-17T00:55:03.000Z",
    "Size": 30
}
```

이제 와일드 카드 표기법 `[*]`를 사용하여 전체 목록을 반복하고 `VolumeId`, `AvailabilityZone`, `Size`와 같은 세 개의 요소를 필터링합니다. 사전 표기법을 사용하려면 {Alias1:Key1,Alias2:Key2}와 같이 각 키의 별칭을 제공해야 합니다. 사전은 본질적으로 순서가 지정되지 않으므로, 경우에 따라 구조 내에서 키 별칭의 순서가 일관되지 않을 수 있습니다.

```
$ aws ec2 describe-volumes --query 'Volumes[*].{ID:VolumeId,AZ:AvailabilityZone,Size:Size}'
[
    {
        "AZ": "us-west-2a",
        "ID": "vol-e11a5288",
        "Size": 30
    },
    {
        "AZ": "us-west-2a",
        "ID": "vol-2e410a47",
        "Size": 8
    }
]
```

또한 사전 표기법에서는 `key1.key2[0].key3`와 같은 체인 키를 사용하여 구조 내에 깊이 중첩된 요소를 필터링할 수 있습니다. 아래 예제에서는 간단히 `Attachments[0].InstanceId`라는 별칭이 지정된 `InstanceId` 키를 사용하여 이 작업을 보여 줍니다.

```
$ aws ec2 describe-volumes --query 'Volumes[*].
{ID:VolumeId,InstanceId:Attachments[0].InstanceId,AZ:AvailabilityZone,Size:Size}'
[
    {
        "InstanceId": "i-a071c394",
        "AZ": "us-west-2a",
        "ID": "vol-e11a5288",
        "Size": 30
    },
    {
        "InstanceId": "i-4b41a37c",
        "AZ": "us-west-2a",
        "ID": "vol-2e410a47",
        "Size": 8
    }
]
```

`[key1, key2]`와 같은 목록 표기법을 사용하여 여러 요소를 필터링할 수도 있습니다. 이렇게 하면 유형과 상관없이 필터링된 모든 속성이 객체당 단 하나의 순서가 지정된 목록으로 서식 지정됩니다.

```
$ aws ec2 describe-volumes --query 'Volumes[*].[VolumeId, Attachments[0].InstanceId,
 AvailabilityZone, Size]'
[
    [
        "vol-e11a5288",
        "i-a071c394",
        "us-west-2a",
        30
    ],
    [
        "vol-2e410a47",
        "i-4b41a37c",
        "us-west-2a",
        8
    ]
]
```

특정 필드의 값을 기준으로 결과를 필터링하려면 JMESPath "?" 연산자를 사용하십시오. 다음 예제 쿼리는 us-west-2a 가용 영역의 볼륨만 출력합니다.

```
$ aws ec2 describe-volumes --query 'Volumes[?AvailabilityZone==`us-west-2a`]'
```

> Note
>
> JMESPath 쿼리 표현식에서 위의 "us-west-2"와 같은 리터럴 값을 지정할 경우 제대로 읽을 수 있도록 값을 백틱(`)으로 묶어야 합니다.

다음 단원에서 자세히 설명할 세 가지 출력 형식과 결합되는 `--query` 옵션은 출력의 내용과 스타일을 사용자 지정하기 위해 사용할 수 있는 강력한 도구입니다. 기본 JSON 처리 라이브러리인 JMESPath의 추가 예와 전체 사양은 http://jmespath.org/specification.html을 참조하십시오.

JSON 출력 형식

JSON은 AWS CLI의 기본값 출력 형식입니다. 대부분은 언어는 내장 기능 또는 공개적으로 사용 가능한 라이브러리를 사용하여 JSON 문자열을 쉽게 디코딩할 수 있습니다. 이전 주제에서 출력 예제와 함께 살펴본 바와 같이, `--query` 옵션은 AWS CLI의 JSON 형식 출력을 필터링하고 서식 지정할 수 있는 강력한 방법을 제공합니다. `--query`로는 가능하지 않을 수 있는 추가 고급 기능이 필요한 경우 명령줄 JSON 처리기인 `jq`를 사용할 수 있습니다. http://stedolan.github.io/jq/에서 이 처리기를 다운로드하고 공식 자습서를 찾아볼 수 있습니다.

텍스트 출력 형식

텍스트 형식은 AWS CLI의 출력을 탭으로 구분된 줄로 구성합니다. 이 방식은 `grep`, `sed`, `awk`와 같은 기존 Unix 텍스트 도구 및 Windows PowerShell에 효과적입니다.

텍스트 출력 형식은 아래와 같은 기본 구조를 따릅니다. 열은 기본 JSON 객체의 해당 키 이름을 기준으로 알파벳 순서로 정렬됩니다.

```
IDENTIFIER  sorted-column1 sorted-column2
IDENTIFIER2 sorted-column1 sorted-column2
```

다음은 텍스트 출력의 예입니다.

```
$ aws ec2 describe-volumes --output text
VOLUMES us-west-2a      2013-09-17T00:55:03.000Z        30      snap-f23ec1c8   in-use
 vol-e11a5288    standard
ATTACHMENTS     2013-09-17T00:55:03.000Z        True    /dev/sda1       i-a071c394
 attached        vol-e11a5288
```

```
VOLUMES us-west-2a      2013-09-18T20:26:15.000Z        8       snap-708e8348   in-use
 vol-2e410a47    standard
ATTACHMENTS     2013-09-18T20:26:16.000Z        True    /dev/sda1       i-4b41a37c
 attached        vol-2e410a47
```

일관된 동작을 유지하려면 텍스트 출력을 *--query* 옵션과 함께 사용하는 것이 좋습니다. 이렇게 하는 이유는 텍스트 형식이 출력 열을 알파벳 순서로 정렬하며, 유사한 리소스에 항상 동일한 키 모음이 있는 것은 아니기 때문입니다. 예를 들어, Linux EC2 인스턴스의 JSON 표시에는 Windows 인스턴스의 JSON 표시에 없는 요소가 있을 수 있으며 반대의 경우도 마찬가지입니다. 또한 리소스에는 향후 업데이트에서 추가되거나 제거되어 열 순서를 변경하는 키-값 요소가 있을 수 있습니다. 이러한 경우 `--query`를 사용하면 출력 형식을 완전히 제어할 수 있도록 텍스트 출력의 기능이 향상됩니다. 아래 예제에서 명령은 표시할 요소를 미리 선택하고 목록 표기법 `[key1, key2, ...]`를 사용하여 열의 순서를 정의합니다. 이렇게 하면 예상 열에 올바른 키 값이 항상 표시된다는 완전한 확신을 사용자에게 제공할 수 있습니다. 마지막으로 가 존재하지 않는 키의 값으로 'None'을 출력하는 방식을 살펴보십시오.

```
$ aws ec2 describe-volumes --query 'Volumes[*].[VolumeId, Attachments[0].InstanceId,
 AvailabilityZone, Size, FakeKey]' --output text
vol-e11a5288    i-a071c394      us-west-2a      30      None
vol-2e410a47    i-4b41a37c      us-west-2a      8       None
```

다음 예제에서는 `aws ec2 describe-instances` 명령의 텍스트 출력과 함께 `grep`와 `awk`를 사용할 수 있는 방법을 보여 줍니다. 첫 번째 명령은 텍스트 출력에 각 인스턴스의 가용 영역, 시/도 및 인스턴스 ID를 표시합니다. 두 번째 명령은 `us-west-2a` 가용 영역에서 실행 중인 모든 인스턴스의 인스턴스 ID만 출력합니다.

```
$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text
us-west-2a      running i-4b41a37c
us-west-2a      stopped i-a071c394
us-west-2b      stopped i-97a217a0
us-west-2a      running i-3045b007
us-west-2a      running i-6fc67758

$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text | grep us-west-2a |
 grep running | awk '{print $3}'
i-4b41a37c
i-3045b007
i-6fc67758
```

다음 명령은 중지된 모든 인스턴스에 대한 유사한 예를 보여 주며 한 단계 더 나아가서 중지된 각 인스턴스의 인스턴스 유형 변경을 자동화합니다.

```
$ aws ec2 describe-instances --query 'Reservations[*].Instances[*].[State.Name,
 InstanceId]' --output text |
> grep stopped |
> awk '{print $2}' |
> while read line;
> do aws ec2 modify-instance-attribute --instance-id $line --instance-type '{"Value":
 "m1.medium"}';
> done
```

다음 출력은 Windows PowerShell에서도 유용합니다. AWS CLI'의 텍스트 출력은 탭으로 구분되기 때문에 `` `t `` 구분 기호를 사용하여 PowerShell에서 어레이로 쉽게 분할할 수 있습니다. 다음 명령은 첫 번째 열(`AvailabilityZone`)이 `us-west-2a`와 일치할 경우 세 번째 열(`InstanceId`)의 값을 표시합니다.

```
> aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text |
%{if ($_.split("`t")[0] -match "us-west-2a") { $_.split("`t")[2]; } }
```

```
i-4b41a37c
i-a071c394
i-3045b007
i-6fc67758
```

테이블 출력 형식

table 형식은 사람이 읽을 수 있는 AWS CLI 출력 표시를 생성합니다. 다음은 일례입니다.

```
$ aws ec2 describe-volumes --output table
-------------------------------------------------------------------------------------------------------------------------
|                                                  DescribeVolumes
                        |
+-----------------------------------------------------------------------------------------------------------------------
+
||                                                    Volumes
                       ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|| AvailabilityZone |        CreateTime         | Size  |  SnapshotId    |  State  |
 VolumeId     | VolumeType   ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
||  us-west-2a      |  2013-09-17T00:55:03.000Z |  30   |  snap-f23ec1c8 |  in-use |  vol-
e11a5288  |  standard    ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|||                                                  Attachments
                      |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
|||        AttachTime         |  DeleteOnTermination   |   Device    | InstanceId   |
 State    |   VolumeId     |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
|||  2013-09-17T00:55:03.000Z |  True                  |  /dev/sda1  |  i-a071c394  |
 attached  |  vol-e11a5288  |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
||                                                    Volumes
                       ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|| AvailabilityZone |        CreateTime         | Size  |  SnapshotId    |  State  |
 VolumeId     | VolumeType   ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
||  us-west-2a      |  2013-09-18T20:26:15.000Z |  8    |  snap-708e8348 |  in-use |
 vol-2e410a47  |  standard    ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|||                                                  Attachments
                      |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
|||        AttachTime         |  DeleteOnTermination   |   Device    | InstanceId   |
 State    |   VolumeId     |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
|||  2013-09-18T20:26:16.000Z |  True                  |  /dev/sda1  |  i-4b41a37c  |
 attached  |  vol-2e410a47  |||
||+--------------------------+----------------------+-------------+--------------
+------------+----------------+||
```

테이블 형식과 함께 `--query` 옵션을 사용하여 원시 출력에서 미리 선택한 요소 집합을 표시할 수 있습니다. 사전 표기법과 목록 표기법의 출력 차이에 주의하십시오. 첫 번째 예제에서는 열 이름이 알파벳 순서로 정렬되고 두 번째 예제에서는 이름 없는 열이 사용자가 정의하는 방식으로 정렬됩니다.

```
$ aws ec2 describe-volumes --query 'Volumes[*].
{ID:VolumeId,InstanceId:Attachments[0].InstanceId,AZ:AvailabilityZone,Size:Size}' --output
 table
-------------------------------------------------------
|                   DescribeVolumes                   |
+------------+----------------+--------------+-------+
|     AZ     |      ID        | InstanceId   | Size  |
+------------+----------------+--------------+-------+
|  us-west-2a|  vol-e11a5288  |  i-a071c394  |  30   |
|  us-west-2a|  vol-2e410a47  |  i-4b41a37c  |  8    |
+------------+----------------+--------------+-------+

$ aws ec2 describe-volumes --query 'Volumes[*].
[VolumeId,Attachments[0].InstanceId,AvailabilityZone,Size]' --output table
-----------------------------------------------------
|                  DescribeVolumes                  |
+--------------+--------------+--------------+-----+
|  vol-e11a5288|  i-a071c394  |  us-west-2a  |  30 |
|  vol-2e410a47|  i-4b41a37c  |  us-west-2a  |  8  |
+--------------+--------------+--------------+-----+
```

AWS Command Line Interface에서 간편 구문 사용

AWS Command Line Interface는 JSON 형식으로 스칼라가 아닌 옵션 파라미터를 가져올 수 있지만, 명령줄에 대규모 JSON 목록이나 구조를 입력하는 것은 지루한 일일 수 있습니다. 이 문제를 해결하기 위해 AWS CLI는 전체 JSON 형식을 사용하는 것보다 더 간단하게 옵션 파라미터를 표시할 수 있는 간편 구문을 지원합니다.

구조 파라미터

AWS CLI에서 간편 구문을 사용하면 사용자가 플랫(중첩되지 않은 구조) 파라미터를 더 쉽게 입력할 수 있습니다. 형식은 쉼표로 구분된 키 값 페어 목록입니다.

Linux, macOS, or Unix

```
--option key1=value1,key2=value2,key3=value3
```

Windows PowerShell

```
--option "key1=value1,key2=value2,key3=value3"
```

이 구문은 JSON 형식의 다음 예와 동등합니다.

```
--option '{"key1":"value1","key2":"value2","key3":"value3"}'
```

쉼표로 구분된 각 키/값 페어 사이에 공백이 없어야 합니다. 다음은 `--provisioned-throughput` 옵션이 간편 방식으로 지정되어 있는 DynamoDB `update-table` 명령입니다.

```
$ aws dynamodb update-table --provisioned-
throughput ReadCapacityUnits=15,WriteCapacityUnits=10 --table-name MyDDBTable
```

이 구문은 JSON 형식의 다음 예와 동등합니다.

```
$ aws dynamodb update-table --provisioned-
throughput '{"ReadCapacityUnits":15,"WriteCapacityUnits":10}' --table-name MyDDBTable
```

목록 파라미터

목록 형식의 입력 파라미터는 JSON과 간편 구문이라는 두 가지 방법으로 지정할 수 있습니다. AWS CLI의 간편 구문은 숫자, 문자열 또는 비중첩 구조가 있는 목록을 더 쉽게 입력할 수 있도록 하기 위해 설계되었습니다. 기본 형식은 여기에 표시됩니다. 여기서 목록의 값은 단일 공백으로 구분됩니다.

```
--option value1 value2 value3
```

이 구문은 JSON 형식의 다음 예와 동등합니다.

```
--option '[value1,value2,value3]'
```

앞에서 언급한 바와 같이, 숫자 목록, 문자열 목록 또는 비중첩 구조 목록을 간편 방식으로 지정할 수 있습니다. 다음은 Amazon EC2에 대한 `stop-instances` 명령의 예입니다. 여기서 `--instance-ids` 옵션에 대한 입력 파라미터(문자열 목록)는 간편 방식으로 지정됩니다.

```
$ aws ec2 stop-instances --instance-ids i-1486157a i-1286157c i-ec3a7e87
```

이 구문은 JSON 형식의 다음 예와 동등합니다.

```
$ aws ec2 stop-instances --instance-ids '["i-1486157a","i-1286157c","i-ec3a7e87"]'
```

다음은 Amazon EC2 `create-tags` 명령의 예입니다. 이 명령은 `--tags` 옵션에 대한 비중첩 구조 목록을 가져옵니다. `--resources` 옵션은 태깅할 인스턴스의 ID를 지정합니다.

```
$ aws ec2 create-tags --resources i-1286157c --tags Key=My1stTag,Value=Value1
 Key=My2ndTag,Value=Value2 Key=My3rdTag,Value=Value3
```

이 구문은 JSON 형식의 다음 예와 동등합니다. JSON 파라미터는 쉽게 읽을 수 있도록 여러 줄로 작성됩니다.

```
$ aws ec2 create-tags --resources i-1286157c --tags '[
  {"Key": "My1stTag", "Value": "Value1"},
  {"Key": "My2ndTag", "Value": "Value2"},
  {"Key": "My3rdTag", "Value": "Value3"}
]'
```

AWS Command Line Interface의 페이지 매김 옵션 사용

많은 항목 목록을 반환할 수 있는 명령의 경우 AWS CLI에서 서비스의 API를 호출하여 목록을 채울 때 CLI의 페이지 매김 동작을 수정하는 데 사용할 수 있는 세 가지 옵션을 추가합니다.

기본적으로 CLI는 1,000페이지 크기를 사용하고 사용 가능한 모든 항목을 검색합니다. 예를 들어, 3,500개 객체를 포함하는 Amazon S3 버킷에서 `aws s3api list-objects`를 실행할 경우 CLI는 백그라운드에서 서비스별 페이지 매김 로직을 처리하면서 Amazon S3에 대한 4개 호출을 작성합니다.

많은 리소스에서 list 명령을 실행할 때 문제가 발생할 경우, 기본값 페이지 크기 너무 높아서 AWS 서비스에 대한 호출 시간이 초과되었을 수 있습니다. `--page-size` 옵션을 사용하여 더 작은 페이지 크기를 지정하면 이 문제를 해결할 수 있습니다. CLI는 계속 전체 목록을 검색하지만, 백그라운드에서 더 많은 수의 호출을 수행하므로 각 호출마다 더 적은 수의 항목을 검색합니다.

```
$ aws s3api list-objects --bucket my-bucket --page-size 100
{
    "Contents": [
...
```

더 적은 항목을 검색하려면 `--max-items` 옵션을 사용합니다. CLI는 페이지 매김을 동일한 방식으로 처리하지만, 사용자가 지정한 항목 수만 출력합니다.

```
$ aws s3api list-objects --bucket my-bucket --max-items 100
{
    "NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==",
    "Contents": [
...
```

출력 항목 수(`--max-items`)가 전체 항목 수보다 적을 경우 사용자가 다음 항목 세트를 검색하기 위해 후속 명령에서 전달할 수 있도록 출력에 `NextToken`이 포함됩니다.

```
$ aws s3api list-objects --bucket my-bucket --max-items 100 --starting-token
 eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==
{
    "NextToken": "eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==",
    "Contents": [
...
```

서비스를 호출할 때마다 같은 순서로 항목이 반환되지는 않습니다. 페이지 중간에서 다음 토큰을 지정할 경우 예상한 것과 다른 결과가 표시될 수 있습니다. 이를 방지하려면 `--page-size` 및 `--max-items`에 동일한 번호를 사용하여 CLI의 페이지 매김을 서비스의 것과 동기화하십시오. 또한 전체 목록을 검색하고 필요한 구문 분석 작업을 로컬에서 수행할 수 있습니다.

Amazon Web Services 작업

이 단원에서는 AWS 서비스에 액세스하기 위한 AWS Command Line Interface 사용 예를 제공합니다. 이러한 예제는 를 사용하여 관리 작업을 수행하는 방법을 보여 주기 위한 것입니다.

각 서비스에 사용 가능한 모든 명령을 전체적으로 참조하려면 AWS CLI Command Reference을 확인하거나 내장 명령줄 도움말을 사용하십시오. 자세한 내용은 AWS Command Line Interface를 사용하여 도움말 보기 (p. 36) 항목을 참조하십시오.

항목

AWS Command Line Interface에서 Amazon DynamoDB 사용

AWS Command Line Interface(AWS CLI)은 Amazon DynamoDB를 지원합니다. 테이블 생성과 같이 특별 작업을 수행할 때 를 사용할 수 있습니다. 또한 이를 사용하여 유틸리티 스크립트 내에 작업을 포함할 수 있습니다.

명령줄 형식은 Amazon DynamoDB API 이름과 해당 API 파라미터 순서로 구성됩니다. AWS CLI는 파라미터 값의 간편 구문과 JSON을 지원합니다.

예를 들어, 다음 명령을 사용하면 이름이 `MusicCollection`인 테이블이 생성됩니다.

> **Note**
>
> 읽기 쉽도록 이 단원에서는 긴 명령이 여러 줄로 나누어져 있습니다. 백슬래시 문자를 사용하면 여러 줄을 Linux 터미널에 복사하고 붙여넣거나 입력할 수 있습니다. 문자를 이스케이프하는 데 백슬래시를 사용하지 않는 셸을 사용할 경우, 백슬래시를 다른 이스케이프 문자로 대체하거나 백슬래시를 제거하고 전체 명령을 한 줄에 넣으십시오.

```
$ aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

다음 명령을 통해 테이블에 새 항목이 추가됩니다. 본 예제에서는 간편 구문과 JSON이 함께 사용되고 있습니다.

```
$ aws dynamodb put-item \
    --table-name MusicCollection \
```

```
    --item '{
        "Artist": {"S": "No One You Know"},
        "SongTitle": {"S": "Call Me Today"} ,
        "AlbumTitle": {"S": "Somewhat Famous"} }' \
    --return-consumed-capacity TOTAL
{
    "ConsumedCapacity": {
        "CapacityUnits": 1.0,
        "TableName": "MusicCollection"
    }
}
$ aws dynamodb put-item \
    --table-name MusicCollection \
    --item '{
        "Artist": {"S": "Acme Band"},
        "SongTitle": {"S": "Happy Day"} ,
        "AlbumTitle": {"S": "Songs About Life"} }' \
    --return-consumed-capacity TOTAL
{
    "ConsumedCapacity": {
        "CapacityUnits": 1.0,
        "TableName": "MusicCollection"
    }
}
```

명령줄에서는 유효한 JSON을 작성하기가 어려울 수 있지만 AWS CLI는 JSON 파일을 읽을 수 있습니다. 예를 들어 `expression-attributes.json`이라는 파일에 저장된 JSON 코드 조각을 고려해 보십시오.

Example expression-attributes.json

```
{
  ":v1": {"S": "No One You Know"},
  ":v2": {"S": "Call Me Today"}
}
```

이제 AWS CLI를 사용하여 `Query` 요청을 발행할 수 있습니다. 이 예제에서는 `expression-attributes.json` 파라미터에 `--expression-attribute-values` 파일의 콘텐츠가 사용됩니다.

```
$ aws dynamodb query --table-name MusicCollection \
    --key-condition-expression "Artist = :v1 AND SongTitle = :v2" \
    --expression-attribute-values file://expression-attributes.json
{
    "Count": 1,
    "Items": [
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "SongTitle": {
                "S": "Call Me Today"
            },
            "Artist": {
                "S": "No One You Know"
            }
        }
    ],
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

AWS CLI와 DynamoDB를 함께 사용하는 방법에 대한 더 많은 설명서를 보려면 https://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html로 이동하십시오.

DynamoDB 외에도 DynamoDB Local과 함께 AWS CLI를 사용할 수 있습니다. DynamoDB Local은 서비스를 모방하는 클라이언트 측 소형 데이터베이스 및 서버입니다. DynamoDB Local을 사용하면 DynamoDB의 테이블 또는 데이터를 실제로 조작하지 않고서도 DynamoDB API를 사용하는 애플리케이션을 쓸 수 있습니다. 그 대신 모든 API 작업이 DynamoDB Local로 재라우팅됩니다. 애플리케이션에서 테이블을 생성하거나 데이터를 수정하면 변경 사항이 로컬 데이터베이스에 기록됩니다. 이를 통해 프로비저닝된 처리량, 데이터 스토리지 및 데이터 전송 요금을 절감할 수 있습니다.

DynamoDB Local 및 AWS CLI와 함께 이를 사용하는 방법에 대한 자세한 내용은 Amazon DynamoDB 개발자 안내서의 다음 섹션을 참조하십시오.

- DynamoDB Local
- DynamoDB Local과 함께 AWS CLI 사용

AWS Command Line Interface를 통해 Amazon EC2 사용

AWS CLI를 사용하여 Amazon EC2 기능에 액세스할 수 있습니다. Amazon EC2에 대한 AWS CLI 명령을 나열하려면 다음 명령을 사용하십시오.

```
aws ec2 help
```

명령을 실행하기 전에 기본 자격 증명을 설정합니다. 자세한 내용은 AWS CLI 구성 (p. 18) 항목을 참조하십시오.

Amazon EC2에 대한 일반적인 작업의 예는 다음 주제를 참조하십시오.

항목

키 페어 사용

AWS CLI를 사용하여 키 페어를 생성, 표시 및 삭제할 수 있습니다. 인스턴스를 시작하거나 해당 인스턴스에 연결할 때 키 페어를 지정해야 합니다.

> **Note**
>
> 예제 명령을 시도하기 전에 기본 자격 증명을 설정하십시오.

항목

키 페어 만들기

`MyKeyPair`라는 키 페어를 생성하려면 create-key-pair 명령을 사용하고, `--query` 옵션 및 `--output text` 옵션을 사용하여 프라이빗 키를 직접 파일에 파이프합니다.

```
aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text >
 MyKeyPair.pem
```

Windows PowerShell의 경우 > file 리디렉션이 UTF-8 인코딩으로 기본 지정되는데, 일부 SSH 클라이언트에서는 이 인코딩을 사용할 수 없습니다. 따라서 out-file 명령에서 ASCII 인코딩을 명시적으로 지정해야 합니다.

```
aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text | out-file
 -encoding ascii -filepath MyKeyPair.pem
```

결과로 나온 MyKeyPair.pem 파일은 다음과 같습니다.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEKEYKCAQEAy7WZhaDsrA1W3mRlQtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
vBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7iO5dSrvC7dQkW2duV5QuUdE0QW
Z/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
G50TCFeOzfl8dqqCP6GzbPaIjiU19xX/azOR9V+tpUOzEL+wmXnZt3/nHPQ5xvD2OJH67km6SuPW
oPzev/D8V+x4+bHthfSjR9Y7DvQFjfBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrqu
/uler7vgIn5m7lN5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/ufGxbL1
mb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2
bahyWyJNfjLe4M86yd2YK3V2CmK+X/BOsShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/tJWSD9
81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA3OzdXzMQexXVJ1TLZVEH0E7bhlY9d8O1ozR
oQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfql+lIp1
YkriL0DbLXlvRAH+yHPRit2hHOjtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh+GTfbjsXsOxQx/x
p9otyVwc7hsQ5TA5PZb+mvkJ5OBEKzet9XcKwONBYELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
ayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWVOEihvm+xTtmaZlSp//lkq75XDwnU
WA8gkn6O3QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
gYBjbO+OZk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
oMo7yykABY7Ozd5wQewBQ4AdSlWSX4nGDtsiFxWiI5sKuAAeOCbTosy1s8w8fxoJ5Tz1sdoxNeGs
Arq6Wv/G16zQuAE9zK9vvwKBgF+09VI/1wJBirsDGz9whVWfFPrTkJNvJZzYt69qezxlsjgFKshy
WBhd4xHZtmCqpBPlAymEjr/TOlbxyARmXMnIOWIAnNXMGB4KGSyl1mzSVAoQ+fqR+cJ3d0dyPl1j
jjb0Ed/NY8frlNDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0iOegLDa
NWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
VRkAKKKYeGjkpUfVTrW0YFjXkfcrR/V+QFL5OndHAKJXjW7a4ejJLncTzmZSpYzwApc=
-----END RSA PRIVATE KEY-----
```

프라이빗 키가 AWS에 저장되지 않고 이 키가 생성될 때만 검색할 수 있습니다.

Linux 컴퓨터에서 SSH 클라이언트를 사용하여 인스턴스에 연결하려면 사용자만 프라이빗 키 파일을 읽을 수 있도록 다음 명령으로 해당 권한을 설정합니다.

```
chmod 400 MyKeyPair.pem
```

키 페어 표시

키 페어에서 지문이 생성되고 이 지문을 사용하여 로컬 시스템에 있는 프라이빗 키가 AWS에 저장된 퍼블릭 키와 일치하는지 확인할 수 있습니다. 지문은 프라이빗 키의 DER 인코딩된 사본에서 가져온 SHA1 해시입니다. 이 값은 AWS에 저장되고 EC2 관리 콘솔에서 보거나 aws ec2 describe-key-pairs를 호출하여 볼 수 있습니다. 예를 들어, 다음 명령을 사용하여 MyKeyPair에 대한 지문을 볼 수 있습니다.

```
aws ec2 describe-key-pairs --key-name MyKeyPair
{
    "KeyPairs": [
        {
            "KeyName": "MyKeyPair",
            "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f"
        }
```

```
    ]
}
```

키 및 지문에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 페이지를 참조하십시오.

키 페어 삭제

`MyKeyPair`를 삭제하려면 다음과 같이 delete-key-pair 명령을 사용합니다.

```
aws ec2 delete-key-pair --key-name MyKeyPair
```

보안 그룹 사용

EC2-Classic 또는 EC2-VPC에서 사용할 보안 그룹을 생성합니다. EC2-Classic 및 EC2-VPC에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 지원되는 플랫폼을 참조하십시오.

AWS CLI를 사용하여 보안 그룹을 생성 및 삭제하고 이 그룹에 규칙을 추가할 수 있습니다

> Note
>
> 예제 명령을 시도하기 전에 기본 자격 증명을 설정하십시오.

항목

보안 그룹 생성

`my-sg`라는 보안 그룹을 생성하려면 create-security-group 명령을 사용합니다.

EC2-VPC

다음 명령은 지정된 VPC에 대한 `my-sg`라는 보안 그룹을 생성합니다.

```
aws ec2 create-security-group --group-name my-sg --description "My security group" --vpc-
id vpc-1a2b3c4d
{
    "GroupId": "sg-903004f8"
}
```

`my-sg`에 대한 초기 정보를 보려면 다음과 같이 describe-security-groups 명령을 사용합니다. 이름으로 EC2-VPC에 대한 보안 그룹을 참조할 수 없습니다.

```
aws ec2 describe-security-groups --group-ids sg-903004f8
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
```

```
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "UserIdGroupPairs": []
                }
            ],
            "Description": "My security group"
            "IpPermissions": [],
            "GroupName": "my-sg",
            "VpcId": "vpc-1a2b3c4d",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

EC2-Classic

다음 명령은 EC2-Classic에 보안 그룹을 생성합니다.

```
aws ec2 create-security-group --group-name my-sg --description "My security group"
{
    "GroupId": "sg-903004f8"
}
```

`my-sg`에 대한 초기 정보를 보려면 다음과 같이 describe-security-groups 명령을 사용합니다.

```
aws ec2 describe-security-groups --group-names my-sg
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [],
            "Description": "My security group"
            "IpPermissions": [],
            "GroupName": "my-sg",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

보안 그룹에 규칙 추가

Windows 인스턴스를 시작할 경우 TCP 포트 3389(RDP)에서 인바운드 트래픽을 허용하는 규칙을 추가해야 합니다. Linux 인스턴스를 시작할 경우 TCP 포트 22(SSH)에서 인바운드 트래픽을 허용하는 규칙을 추가해야 합니다. authorize-security-group-ingress 명령을 사용하여 보안 그룹에 규칙을 추가합니다. 이 명령의 필수 파라미터 중 하나가 컴퓨터의 퍼블릭 IP 주소(CIDR 표기법 사용)입니다.

> **Note**
>
> 서비스를 사용하여 로컬 컴퓨터의 퍼블릭 IP 주소를 확인할 수 있습니다. 예를 들어 https://checkip.amazonaws.com/ 서비스를 제공합니다. IP 주소를 제공하는 다른 서비스를 찾으려면 "what is my IP address"로 검색하십시오. 고정 IP 주소 없이 ISP 또는 방화벽을 경유하여 연결하는 경우에는 클라이언트 컴퓨터가 사용하는 IP 주소의 범위를 알아내야 합니다.

EC2-VPC

다음 명령은 sg-903004f8 ID를 가진 보안 그룹에 RDP 규칙을 추가합니다.

```
aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 3389
 --cidr 203.0.113.0/24
```

다음 명령은 sg-903004f8 ID를 가진 보안 그룹에 SSH 규칙을 추가합니다.

```
aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 22 --
cidr 203.0.113.0/24
```

my-sg에 대한 변경 사항을 보려면 다음과 같이 describe-security-groups 명령을 사용합니다.

```
aws ec2 describe-security-groups --group-ids sg-903004f8
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "UserIdGroupPairs": []
                }
            ],
            "Description": "My security group"
            "IpPermissions": [
                {
                    "ToPort": 22,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "203.0.113.0/24"
                        }
                    ]
                    "UserIdGroupPairs": [],
                    "FromPort": 22
                }
            ],
            "GroupName": "my-sg",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

EC2-Classic

다음 명령은 RDP에 대한 규칙을 my-sg보안 그룹에 추가합니다.

```
aws ec2 authorize-security-group-ingress --group-name my-sg --protocol tcp --port 3389 --
cidr 203.0.113.0/24
```

다음 명령은 SSH에 대한 규칙을 my-sg 보안 그룹에 추가합니다.

```
aws ec2 authorize-security-group-ingress --group-name my-sg --protocol tcp --port 22 --
cidr 203.0.113.0/24
```

my-sg에 대한 변경 사항을 보려면 다음과 같이 describe-security-groups 명령을 사용합니다.

```
aws ec2 describe-security-groups --group-names my-sg
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [],
            "Description": "My security group"
            "IpPermissions": [
                {
                    "ToPort": 22,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "203.0.113.0/24"
                        }
                    ]
                    "UserIdGroupPairs": [],
                    "FromPort": 22
                }
            ],
            "GroupName": "my-sg",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

보안 그룹 삭제

보안 그룹을 삭제하려면 delete-security-group 명령을 사용합니다. 환경에 연결되어 있는 경우 보안 그룹을 삭제할 수 없습니다.

EC2-VPC

다음 명령은 sg-903004f8 ID를 가진 보안 그룹을 삭제합니다.

```
aws ec2 delete-security-group --group-id sg-903004f8
```

EC2-Classic

다음 명령은 `my-sg`라는 보안 그룹을 삭제합니다.

```
aws ec2 delete-security-group --group-name my-sg
```

Amazon EC2 인스턴스 사용

AWS CLI를 사용하여 인스턴스를 시작, 나열 및 종료할 수 있습니다. 키 페어와 보안 그룹이 필요합니다. AWS CLI를 통해 이를 생성하는 방법에 대한 내용은 키 페어 사용 (p. 59) 및 보안 그룹 사용 (p. 61) 단원을 참조하십시오. 또한 Amazon 머신 이미지(AMI)를 선택하고 AMI ID를 적어 두어야 합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 적합한 AMI 찾기를 참조하십시오.

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

> Note
>
> 예제 명령을 시도하기 전에 기본 자격 증명을 설정하십시오.

항목

인스턴스 시작하기

선택한 AMI를 사용하여 단일 Amazon EC2 인스턴스를 시작하려면 run-instances 명령을 사용합니다. 계정에서 지원하는 플랫폼을 따라 인스턴스를 EC2-Classic 또는 EC2-VPC에서 시작할 수 있습니다.

처음에는 인스턴스가 `pending` 상태이지만 몇 분 후에 `running` 상태가 됩니다.

EC2-VPC

다음 명령은 지정된 서브넷에서 `t2.micro` 인스턴스를 시작합니다.

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-
name MyKeyPair --security-group-ids sg-xxxxxxxx --subnet-id subnet-xxxxxxxx
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "PrivateIpAddress": "10.0.1.114",
            "ProductCodes": [],
            "VpcId": "vpc-1a2b3c4d",
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": ip-10-0-1-114.ec2.internal,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "SubnetId": "subnet-6e7f829e",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
```

```
                    "Status": "in-use",
                    "SourceDestCheck": true,
                    "VpcId": "vpc-1a2b3c4d",
                    "Description": "Primary network interface",
                    "NetworkInterfaceId": "eni-a7edb1c9",
                    "PrivateIpAddresses": [
                        {
                            "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                            "Primary": true,
                            "PrivateIpAddress": "10.0.1.114"
                        }
                    ],
                    "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                    "Attachment": {
                        "Status": "attached",
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "AttachmentId": "eni-attach-52193138",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    },
                    "Groups": [
                        {
                            "GroupName": "my-sg",
                            "GroupId": "sg-903004f8"
                        }
                    ],
                    "SubnetId": "subnet-6e7f829e",
                    "OwnerId": "123456789012",
                    "PrivateIpAddress": "10.0.1.114"
                }
            ],
            "SourceDestCheck": true,
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "RootDeviceName": "/dev/sda1",
            "VirtualizationType": "hvm",
            "RootDeviceType": "ebs",
            "Tags": [
                {
                    "Value": "MyInstance",
                    "Key": "Name"
                }
            ],
            "AmiLaunchIndex": 0
        }
    ]
```

```
}
```

EC2-Classic

다음 명령은 EC2-Classic에서 `t1.micro` 인스턴스를 시작합니다.

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t1.micro --key-
name MyKeyPair --security-groups my-sg
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "ProductCodes": [],
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": null,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "InstanceType": "t1.micro",
            "NetworkInterfaces": [],
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
```

```
                "Code": "pending"
            },
            "RootDeviceName": "/dev/sda1",
            "VirtualizationType": "hvm",
            "RootDeviceType": "ebs",
            "Tags": [
                {
                    "Value": "MyInstance",
                    "Key": "Name"
                }
            ],
            "AmiLaunchIndex": 0
        }
    ]
}
```

인스턴스에 블록 디바이스 매핑 추가

실행된 각 인스턴스에는 연관된 루트 디바이스 볼륨이 있습니다. 블록 디바이스 매핑을 사용하면 실행될 때 인스턴스에 연결할 추가 EBS 볼륨 또는 인스턴스 스토어 볼륨을 지정할 수 있습니다.

인스턴스에 블록 디바이스 매핑을 추가하려면 `run-instances`를 사용할 때 `--block-device-mappings` 옵션을 지정합니다.

다음 예제에서는 `/dev/sdf`에 매핑된 20GB 크기의 표준 Amazon EBS 볼륨을 추가합니다.

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"VolumeSize\":20,
\"DeleteOnTermination\":false}}]"
```

다음 예제에서는 스냅샷을 기반으로 `/dev/sdf`에 매핑된 Amazon EBS 볼륨을 추가합니다. 스냅샷을 지정할 때 볼륨 크기를 지정할 필요는 없지만, 지정할 경우 스냅샷 크기보다 크거나 같아야 합니다.

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":
\"snap-xxxxxxxx\"}}]"
```

다음 예제에서는 인스턴스 스토어 볼륨 2개를 추가합니다. 인스턴스에 사용 가능한 인스턴스 스토어 볼륨 수는 인스턴스 유형에 따라 다릅니다.

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdf\",\"VirtualName\":\"ephemeral0\"},
{\"DeviceName\":\"/dev/sdg\",\"VirtualName\":\"ephemeral1\"}]"
```

다음 예제에서는 인스턴스(`/dev/sdj`)를 시작하기 위해 AMI에서 지정한 디바이스에 대한 매핑을 생략합니다.

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdj\",\"NoDevice\":\"\"}]"
```

자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 블록 디바이스 매핑을 참조하십시오.

인스턴스에 이름 태그 추가

인스턴스에 `Name=MyInstance` 태그를 추가하려면 다음과 같이 create-tags 명령을 사용합니다.

```
aws ec2 create-tags --resources i-xxxxxxxx --tags Key=Name,Value=MyInstance
```

자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 리소스에 태그 지정을 참조하십시오.

인스턴스에 연결

인스턴스가 실행되는 동안 실행 중인 인스턴스에 연결하여 바로 앞에 있는 컴퓨터를 사용하는 것처럼 인스턴스를 사용할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 인스턴스에 연결을 참조하십시오.

인스턴스 나열

AWS CLI를 사용하여 인스턴스를 나열하고 정보를 확인할 수 있습니다. 모든 인스턴스를 나열하거나 관심이 있는 인스턴스에 따라 결과를 필터링할 수 있습니다.

> Note
>
> 예제 명령을 시도하기 전에 기본 자격 증명을 설정하십시오.

다음 예제에서는 describe-instances 명령을 사용하는 방법을 보여 줍니다.

Example 1: 지정한 인스턴스 유형을 가진 인스턴스 나열

다음 명령은 `t2.micro` 인스턴스를 나열합니다.

```
aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro" --query
 Reservations[].Instances[].InstanceId
```

Example 2: 지정한 태그를 가진 인스턴스 나열

다음 명령은 Name=MyInstance 태그를 가진 인스턴스를 나열합니다.

```
aws ec2 describe-instances --filters "Name=tag:Name,Values=MyInstance"
```

Example 3: 지정된 이미지를 사용하여 시작된 인스턴스 나열

다음 명령은 다음 AMI에서 시작된 인스턴스를 나열합니다(`ami-x0123456`, `ami-y0123456` 및 `ami-z0123456`).

```
aws ec2 describe-instances --filters "Name=image-id,Values=ami-x0123456,ami-y0123456,ami-
z0123456"
```

인스턴스 종료

인스턴스를 종료하면 인스턴스가 실제로 삭제되므로 인스턴스를 종료한 후에는 인스턴스에 다시 연결할 수 없습니다. 인스턴스 상태가 `shutting-down` 또는 `terminated`로 변경되는 즉시 해당 인스턴스에 대한 반복적인 요금 부과가 중단됩니다.

인스턴스가 완료되면 다음과 같이 terminate-instances 명령을 사용합니다.

```
aws ec2 terminate-instances --instance-ids i-5203422c
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-5203422c",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
```

```
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 인스턴스 종료를 참조하십시오.

AWS Command Line Interface에서 Amazon S3 Glacier 사용

대용량 파일을 분할하고 명령줄에서 업로드하여 Glacier로 업로드할 수 있습니다. 이 주제에서는 AWS CLI를 사용하여 볼트를 생성하고, 파일을 분할하며, 멀티파트를 구성하여 Glacier로 업로드하는 프로세스에 대해 설명합니다.

> **Note**
>
> 이 자습서에서는 Linux 및 OS X를 포함한 Unix 같은 운영 체제에서 일반적으로 사전 설치되어 제공되는 여러 명령줄 도구를 사용합니다. Windows 사용자는 Cygwin을 설치하고 Cygwin 터미널에서 명령을 실행하여 동일한 도구를 사용할 수 있습니다. 동일한 기능을 수행하는 Windows 기본 명령 및 유틸리티가 표시되어 있습니다(사용 가능한 경우).

항목

Glacier 볼트 생성

`aws glacier create-vault` 명령을 사용하여 볼트를 생성합니다. 다음 명령은 `myvault`라는 볼트를 생성합니다.

```
$ aws glacier create-vault --account-id - --vault-name myvault
{
    "location": "/123456789012/vaults/myvault"
}
```

> **Note**
>
> 모든 glacier 명령에는 계정 ID 파라미터가 필요합니다. 하이픈을 사용하여 현재 계정을 지정합니다.

파일 업로드 준비

테스트 업로드를 위한 파일을 생성합니다. 다음 명령은 3MiB(3x1024x1024바이트)의 임의 데이터를 포함하는 파일을 생성합니다.

Linux, macOS, or Unix

```
$ dd if=/dev/urandom of=largefile bs=3145728 count=1
1+0 records in
1+0 records out
3145728 bytes (3.1 MB) copied, 0.205813 s, 15.3 MB/s
```

dd는 입력 파일에서 출력 파일로 많은 바이트를 복사하는 유틸리티입니다. 위의 예제에서는 디바이스 파일 /dev/urandom을 임의 데이터 소스로 사용합니다. fsutil은 Windows에서 유사한 함수를 수행합니다.

Windows

```
C:\temp>fsutil file createnew largefile 3145728
File C:\temp\largefile is created
```

그런 다음 파일을 1MiB(1,048,576바이트) 조각으로 분할합니다.

```
$ split --bytes=1048576 --verbose largefile chunk
creating file `chunkaa'
creating file `chunkab'
creating file `chunkac'
```

> **Note**
>
> HJ-Split는 Windows 및 많은 다른 플랫폼에서 사용할 수 있는 무료 파일 분할기입니다.

멀티파트 업로드 및 파일 업로드 시작

aws glacier initiate-multipart-upload 명령을 사용하여 Glacier에 멀티파트 업로드를 생성하십시오.

```
$ aws glacier initiate-multipart-upload --account-id - --archive-description "multipart
 upload test" --part-size 1048576 --vault-name myvault
{
    "uploadId": "19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ",
    "location": "/123456789012/vaults/myvault/multipart-
uploads/19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ"
}
```

멀티파트 업로드를 구성하기 위해 Glacier에서 각 파트의 크기(바이트, 이 예제에서는 1MiB), 볼트 이름 및 계정 ID를 요구합니다. 작업이 완료되면 에서 업로드 ID를 출력합니다. 나중에 사용하기 위해 업로드 ID를 셸 변수에 저장합니다.

Linux, macOS, or Unix

```
$ UPLOADID="19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ"
```

Windows

```
C:\temp> set UPLOADID="19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ"
```

그런 다음 aws glacier upload-multipart-part 명령을 사용하여 각 파트를 업로드합니다.

```
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkaa --range 'bytes
 0-1048575/*' --account-id - --vault-name myvault
{
    "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"
}
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkab --range 'bytes
 1048576-2097151/*' --account-id - --vault-name myvault
{
    "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"
}
$ aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkac --range 'bytes
 2097152-3145727/*' --account-id - --vault-name myvault
{
    "checksum": "e1f2a7cd6e047fa606fe2f0280350f69b9f8cfa602097a9a026360a7edc1f553"
}
```

Note

위의 예에서는 달러 기호("$")를 사용하여 `UPLOADID` 셸 변수를 역참조합니다. Windows 명령줄에서는 두 개의 퍼센트 기호(예: `%UPLOADID%`)를 사용합니다.

Glacier에서 올바른 순서로 다시 수집할 수 있도록 각 파트를 업로드할 때 각 파트의 바이트 범위를 지정해야 합니다. 각 조각은 1048576바이트입니다. 따라서 첫 번째 조각은 0-1048575바이트, 두 번째 조각은 1048576-2097151바이트, 세 번째 조각은 2097152-3145727바이트를 차지합니다.

업로드 완료

업로드된 모든 조각이 AWS에 변동 없이 도달했는지 확인하기 위해 Glacier에는 원본 파일의 트리 해시가 필요합니다. 트리 해시를 계산하려면 파일을 1MiB 파트로 분할하고 각 조각의 이진 SHA-256 해시를 계산합니다. 그런 다음 해시 목록을 쌍으로 분할하고, 2개의 이진 해시를 각 쌍으로 결합하며, 결과의 해시를 가져옵니다. 하나의 해시만 남을 때까지 이 프로세스를 반복합니다. 임의 레벨에서 홀수 해시가 있을 경우 수정하지 않고 다음 레벨로 승격시킵니다.

명령줄 유틸리티를 사용할 때 트리 해시를 올바로 계산하는 핵심은 각 해시를 이진 형식으로 저장하고 마지막 단계에서만 16진수로 변환하는 것입니다. 트리의 16진수 버전 해시를 결합하거나 해시할 경우 잘못된 결과가 발생할 수 있습니다.

Note

Windows 사용자는 `cat` 대신 `type` 명령을 사용할 수 있습니다. OpenSSL은 OpenSSL.org에서 Windows용으로 제공됩니다.

트리 해시를 계산하려면

1. 원본 파일을 1MiB로 분할합니다(아직 분할하지 않은 경우).

```
$ split --bytes=1048576 --verbose largefile chunk
creating file `chunkaa'
creating file `chunkab'
creating file `chunkac'
```

2. 각 청크의 이진 SHA-256 해시를 계산 및 저장합니다.

```
$ openssl dgst -sha256 -binary chunkaa > hash1
$ openssl dgst -sha256 -binary chunkab > hash2
$ openssl dgst -sha256 -binary chunkac > hash3
```

3. 처음 2개 해시를 결합하고 결과의 이진 해시를 가져옵니다.

```
$ cat hash1 hash2 > hash12
$ openssl dgst -sha256 -binary hash12 > hash12hash
```

4. 청크 aa 및 ab의 상위 해시를 청크 ac의 해시와 결합하고 결과를 해시합니다. 이때는 16진수가 출력됩니다. 결과를 셸 변수에 저장합니다.

```
$ cat hash12hash hash3 > hash123
$ openssl dgst -sha256 hash123
SHA256(hash123)= 9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67
$ TREEHASH=9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67
```

마지막으로 `aws glacier complete-multipart-upload` 명령을 사용하여 업로드를 완료합니다. 이 명령에서는 원본 파일의 크기(바이트), 최종 트리 해시 값(16진수) 및 계정 ID와 볼트 이름을 사용합니다.

```
$ aws glacier complete-multipart-upload --checksum $TREEHASH --archive-size 3145728 --
upload-id $UPLOADID --account-id - --vault-name myvault
{
    "archiveId": "d3AbWhE0YE1m6f_fI1jPG82F8xzbMEEZmrAlLGAAONJAzo5QdP-
N83MKqd96Unspoa5H5lItWX-sK8-QS0ZhwsyGiu9-R-kwWUyS1dSBlmgPPWkEbeFfqDSav053rU7FvVLHfRc6hg",
    "checksum": "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
    "location": "/123456789012/vaults/myvault/archives/
d3AbWhE0YE1m6f_fI1jPG82F8xzbMEEZmrAlLGAAONJAzo5QdP-N83MKqd96Unspoa5H5lItWX-sK8-
QS0ZhwsyGiu9-R-kwWUyS1dSBlmgPPWkEbeFfqDSav053rU7FvVLHfRc6hg"
}
```

`aws glacier describe-vault`를 사용하여 볼트 상태를 확인할 수도 있습니다.

```
$ aws glacier describe-vault --account-id - --vault-name myvault
{
    "SizeInBytes": 3178496,
    "VaultARN": "arn:aws:glacier:us-west-2:123456789012:vaults/myvault",
    "LastInventoryDate": "2015-04-07T00:26:19.028Z",
    "NumberOfArchives": 1,
    "CreationDate": "2015-04-06T21:23:45.708Z",
    "VaultName": "myvault"
}
```

> **Note**
>
> 볼트 상태는 매일 한 번 정도 업데이트됩니다. 자세한 내용은 볼트 작업을 참조하십시오.

이제 생성한 파트 및 해시 파일을 안전하게 제거할 수 있습니다.

```
$ rm chunk* hash*
```

멀티파트 업로드에 대한 자세한 내용은 Amazon S3 Glacier 개발자 안내서의 파트로 대용량 아카이브 업로드 및 체크섬 컴퓨팅을 참조하십시오.

AWS Command Line Interface의 AWS Identity and Access Management

이 섹션에서는 AWS Identity and Access Management (IAM)와 관련된 몇 가지 일반적인 작업 및 AWS Command Line Interface를 사용하여 이를 수행하는 방법에 대해 설명합니다.

여기에 표시된 명령은 기본 자격 증명 및 기본 리전을 설정했다고 가정합니다.

항목

새 IAM 사용자 및 그룹 생성

이 섹션에서는 새 IAM 그룹과 새 IAM 사용자를 생성한 다음 사용자를 그룹에 추가하는 방법을 설명합니다.

IAM 그룹을 생성하고 이 그룹에 새 IAM 사용자를 추가하려면

1. 먼저, `create-group` 명령을 사용하여 그룹을 생성합니다.

```
$ aws iam create-group --group-name MyIamGroup
{
    "Group": {
        "GroupName": "MyIamGroup",
        "CreateDate": "2012-12-20T03:03:52.834Z",
        "GroupId": "AKIAI44QH8DHBEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    }
}
```

2. 다음에는 `create-user` 명령을 사용하여 사용자를 생성합니다.

```
$ aws iam create-user --user-name MyUser
{
    "User": {
        "UserName": "MyUser",
        "Path": "/",
        "CreateDate": "2012-12-20T03:13:02.581Z",
        "UserId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/MyUser"
    }
}
```

3. 마지막으로 `add-user-to-group` 명령을 사용하여 사용자를 그룹에 추가합니다.

```
$ aws iam add-user-to-group --user-name MyUser --group-name MyIamGroup
```

4. `MyIamGroup` 그룹에 `MyUser`가 포함되어 있는지 확인하려면 `get-group` 명령을 사용합니다.

```
$ aws iam get-group --group-name MyIamGroup
{
    "Group": {
        "GroupName": "MyIamGroup",
        "CreateDate": "2012-12-20T03:03:52Z",
        "GroupId": "AKIAI44QH8DHBEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    },
    "Users": [
        {
            "UserName": "MyUser",
```

```
            "Path": "/",
            "CreateDate": "2012-12-20T03:13:02Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/MyUser"
        }
    ],
    "IsTruncated": "false"
}
```

또한 AWS Management 콘솔을 사용하여 IAM 사용자 및 그룹을 볼 수 있습니다.

IAM 사용자에 대한 IAM 정책 설정

다음 명령은 IAM 정책을 IAM 사용자에게 할당하는 방법을 보여 줍니다. 여기에 지정된 정책은 사용자에게 "파워 유저 액세스"를 제공합니다. 이 정책은 IAM 콘솔에서 제공하는 파워 유저 액세스 정책 템플릿과 동일합니다. 이 예제에서 정책은 파일 `MyPolicyFile.json`에 저장됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

정책을 지정하려면 `put-user-policy` 명령을 사용합니다.

```
$ aws iam put-user-policy --user-name MyUser --policy-name MyPowerUserRole --policy-
document file://C:\Temp\MyPolicyFile.json
```

`list-user-policies` 명령을 사용하여 사용자에게 정책이 할당되었는지 확인합니다.

```
$ aws iam list-user-policies --user-name MyUser
{
    "PolicyNames": [
        "MyPowerUserRole"
    ],
    "IsTruncated": "false"
}
```

추가 리소스

자세한 내용은 권한 및 정책 학습을 위한 리소스를 참조하십시오. 이 주제에서는 권한 및 정책 개요에 대한 링크와 Amazon S3, Amazon EC2 및 기타 서비스에 액세스하기 위한 정책 예에 대한 링크를 제공합니다.

IAM 사용자의 초기 암호 설정

다음 예제에서는 `create-login-profile` 명령을 사용하여 IAM 사용자의 초기 암호를 설정하는 방법을 보여 줍니다.

```
$ aws iam create-login-profile --user-name MyUser --password My!User1Login8P@ssword
{
```

```
    "LoginProfile": {
        "UserName": "MyUser",
        "CreateDate": "2013-01-02T21:10:54.339Z",
        "MustChangePassword": "false"
    }
}
```

`update-login-profile` 명령을 사용하여 IAM 사용자의 암호를 업데이트하십시오.

IAM 사용자의 보안 자격 증명 생성

다음 예제에서는 `create-access-key` 명령을 사용하여 IAM 사용자의 보안 자격 증명을 생성합니다. 보안 자격 증명 세트는 액세스 키 ID와 보안 키로 구성됩니다. 사용자는 어떤 시점에서도 자격 증명 세트를 두 개 이상 가질 수 없습니다. 세 번째 세트를 생성하려고 시도하면 `create-access-key` 명령에서 "LimitExceeded" 오류가 반환됩니다.

```
$ aws iam create-access-key --user-name MyUser
{
    "AccessKey": {
        "SecretAccessKey": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "Status": "Active",
        "CreateDate": "2013-01-02T22:44:12.897Z",
        "UserName": "MyUser",
        "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
    }
}
```

`delete-access-key` 명령을 사용하여 IAM 사용자의 자격 증명 세트를 삭제하십시오. 액세스 키 ID를 사용하여 삭제할 자격 증명을 지정합니다.

```
$ aws iam delete-access-key --user-name MyUser --access-key-id AKIAI44QH8DHBEXAMPLE
```

AWS Command Line Interface에서 Amazon S3 사용

AWS CLI는 Amazon S3에 액세스하기 위한 2가지 티어 명령을 제공합니다.

- s3라는 제1 티어는 객체 및 버킷 생성, 조작 및 삭제와 같이 자주 사용하는 작업을 위한 상위 수준 명령으로 구성됩니다.
- s3api라는 제2 티어에는 버킷 ACL(액세스 제어 목록) 수정, CORS(cross-origin 리소스 공유) 사용, 로깅 정책을 포함한 모든 Amazon S3 작업이 표시됩니다. 이 계층에서는 상위 수준 명령만으로는 가능하지 않을 수 있는 고급 작업을 수행할 수 있습니다.

각 티어에서 사용 가능한 모든 명령의 목록을 보려면 `aws s3` 또는 `aws s3api` 명령과 함께 `help` 인수를 사용합니다.

```
$ aws s3 help
```

또는

```
$ aws s3api help
```

> **Note**
>
> AWS CLI는 Amazon S3에서 Amazon S3으로 복사, 이동 및 동기화를 지원합니다. 이러한 작업은 Amazon S3에서 제공하는 service-side COPY 작업을 사용합니다. 파일은 클라우드에 보관되며 클라이언트 컴퓨터에 다운로드된 다음 Amazon S3로 백업되지 않습니다.
> 이러한 작업을 완전히 클라우드에서 수행할 수 있으면 HTTP 요청 및 응답에 필요한 대역폭만 사용됩니다.

Amazon S3 사용량 예제는 이 섹션의 다음 주제를 참조하십시오.

항목

AWS Command Line Interface에서 상위 수준 s3 명령 사용

이 섹션에서는 상위 수준 `aws s3` 명령을 사용하여 Amazon S3 버킷과 객체를 관리할 수 있는 방법을 설명합니다.

버킷 관리

상위 수준 `aws s3` 명령은 버킷 생성, 제거 및 나열과 같이 일반적으로 사용되는 버킷 작업을 지원합니다.

버킷 생성

`aws s3 mb` 명령을 사용하여 새 버킷을 생성합니다. 버킷 이름은 고유해야 하며 DNS를 준수해야 합니다. 버킷 이름에는 소문자, 숫자, 하이픈, 마침표가 포함될 수 있습니다. 버킷 이름은 문자나 숫자로만 시작하고 끝날 수 있으며 하이픈이나 다른 마침표 옆에 마침표가 포함될 수 없습니다.

```
$ aws s3 mb s3://bucket-name
```

버킷 제거

버킷을 제거하려면 `aws s3 rb` 명령을 사용합니다.

```
$ aws s3 rb s3://bucket-name
```

기본적으로 작업에 성공하려면 버킷이 비어 있어야 합니다. 비어 있지 않은 버킷을 제거하려면 `--force` 옵션을 포함시켜야 합니다.

```
$ aws s3 rb s3://bucket-name --force
```

그러면 먼저 버킷에서 모든 객체와 하위 폴더가 삭제된 다음 버킷이 제거됩니다.

> **Note**
>
> 이전에 삭제했지만 보관된 객체가 포함되어 있는 버전 지정된 버킷을 사용할 경우 이 명령을 사용하여 버킷을 제거할 수 없습니다.

버킷 나열

모든 버킷 또는 해당 콘텐츠를 나열하려면 `aws s3 ls` 명령을 사용합니다. 다음은 일반적인 사용 예입니다.

다음 명령은 모든 버킷을 나열합니다.

```
$ aws s3 ls
2013-07-11 17:08:50 my-bucket
2013-07-24 14:55:44 my-bucket2
```

다음 명령은 버킷에 있는 모든 객체와 폴더(접두사)를 나열합니다.

```
$ aws s3 ls s3://bucket-name
                           PRE path/
2013-09-04 19:05:48          3 MyFile1.txt
```

다음 명령은 *bucket-name*/path에 있는 객체(다시 말해서 접두사 path/를 기준으로 필터링된 *bucket-name*에 있는 객체)를 나열합니다.

```
$ aws s3 ls s3://bucket-name/path/
2013-09-06 18:59:32          3 MyFile2.txt
```

객체 관리

상위 수준 aws s3 명령을 사용하면 Amazon S3 객체도 편리하게 관리할 수 있습니다. 객체 명령에는 aws s3 cp, aws s3 ls, aws s3 mv, aws s3 rm 및 sync가 포함됩니다. cp, ls, mv 및 rm 명령은 Unix의 해당 명령과 유사하게 작동하며 이러한 명령을 사용하여 로컬 디렉터리와 Amazon S3 버킷 전반에 걸쳐 원활하게 작업할 수 있습니다. sync 명령은 버킷과 디렉터리 또는 두 버킷의 내용을 동기화합니다.

> **Note**
>
> 객체를 Amazon S3 버킷에 업로드하는 작업과 관련된 모든 상위 수준 명령(aws s3 cp, aws s3 mv 및 aws s3 sync)은 객체가 큰 경우 멀티파트 업로드를 자동으로 수행합니다.
> 이러한 명령을 사용할 때는 실패한 업로드를 재개할 수 없습니다. 시간 초과로 인해 멀티파트 업로드가 실패하거나 CTRL+C를 눌러 멀티파트 업로드를 수동으로 취소할 경우 AWS CLI는 생성된 모든 파일을 정리하고 업로드를 중단합니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.
> kill 명령 또는 시스템 결함으로 인해 프로세스가 중단될 경우 진행 중인 멀티파트 업로드는 Amazon S3에서 그대로 유지되며, AWS Management 콘솔에서 또는 s3api abort-multipart-upload 명령을 사용하여 수동으로 정리해야 합니다.

cp, mv 및 sync 명령에는 지정된 사용자 또는 그룹에게 객체에 대한 권한을 부여하기 위해 사용할 수 있는 --grants 옵션이 포함됩니다. 다음 구문을 사용하여 --grants 옵션을 권한 목록으로 설정합니다.

```
--grants Permission=Grantee_Type=Grantee_ID
         [Permission=Grantee_Type=Grantee_ID ...]
```

각 값에는 다음 요소가 포함됩니다.

- *##* – 부여된 권한을 지정하며 read, readacl, writeacl 또는 full로 설정할 수 있습니다.
- *Grantee_Type* – 피부여자 식별 방법을 지정하며 uri, emailaddress 또는 id로 설정할 수 있습니다.
- *Grantee_ID* – *Grantee_Type*을 기준으로 피부여자를 지정합니다.
  - uri – 그룹의 URI입니다. 자세한 내용은 피부여자란?을 참조하십시오.
  - emailaddress – 계정의 이메일 주소입니다.
  - id – 계정의 정식 ID입니다.

Amazon S3 액세스 제어에 대한 자세한 내용은 액세스 제어를 참조하십시오.

다음 예제에서는 버킷에 객체를 복사합니다. 여기서는 모든 사람에게 객체에 대한 read 권한을 부여하고 user@example.com과 연결된 계정에 full 권한(read, readacl 및 writeacl)을 부여합니다.

```
$ aws s3 cp file.txt s3://my-bucket/ --grants read=uri=http://acs.amazonaws.com/groups/
global/AllUsers full=emailaddress=user@example.com
```

Amazon S3에 업로드하는 객체에 대해 기본값이 아닌 스토리지 클래스(REDUCED_REDUNDANCY 또는 STANDARD_IA)를 지정하려면 --storage-class 옵션을 사용하십시오.

```
$ aws s3 cp file.txt s3://my-bucket/ --storage-class REDUCED_REDUNDANCY
```

sync 명령의 형식은 다음과 같습니다. 가능한 원본-대상 조합은 다음과 같습니다.

- 로컬 파일 시스템에서 Amazon S3으로
- Amazon S3에서 로컬 파일 시스템으로
- Amazon S3 Amazon S3

```
$ aws s3 sync <source> <target> [--options]
```

다음 예제에서는 my-bucket에 있는 경로라는 이름의 Amazon S3 폴더 내용을 현재 작업 중인 디렉터리와 동기화합니다. s3 sync는 대상에서 이름이 같은 파일보다 크기나 수정된 시간이 다른 모든 파일을 업데이트합니다. 출력에는 동기화 중에 수행된 특정 작업이 표시됩니다. 이 작업은 하위 디렉터리 MySubdirectory와 해당 내용을 s3://my-bucket/path/MySubdirectory와 반복적으로 동기화합니다.

```
$ aws s3 sync . s3://my-bucket/path
upload: MySubdirectory\MyFile3.txt to s3://my-bucket/path/MySubdirectory/MyFile3.txt
upload: MyFile2.txt to s3://my-bucket/path/MyFile2.txt
upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt
```

일반적으로 sync는 원본과 대상 간에 누락되거나 오래된 파일 또는 객체만 복사합니다. 하지만 --delete 옵션을 제공하여 원본에 없는 파일이나 객체를 대상에서 제거할 수 있습니다.

다음 예제에서는 이전 예제를 확장하여 이 옵션이 작동하는 방식을 보여 줍니다.

```
// Delete local file
$ rm ./MyFile1.txt

// Attempt sync without --delete option - nothing happens
$ aws s3 sync . s3://my-bucket/path

// Sync with deletion - object is deleted from bucket
$ aws s3 sync . s3://my-bucket/path --delete
delete: s3://my-bucket/path/MyFile1.txt

// Delete object from bucket
$ aws s3 rm s3://my-bucket/path/MySubdirectory/MyFile3.txt
delete: s3://my-bucket/path/MySubdirectory/MyFile3.txt

// Sync with deletion - local file is deleted
$ aws s3 sync s3://my-bucket/path . --delete
delete: MySubdirectory\MyFile3.txt

// Sync with Infrequent Access storage class
$ aws s3 sync . s3://my-bucket/path --storage-class STANDARD_IA
```

--exclude 및 --include 옵션을 사용하면 동기화 작업 중에 복사할 파일이나 객체를 필터링하기 위한 규칙을 지정할 수 있습니다. 기본적으로 지정된 디렉터리에 있는 모든 항목이 동기화에 포함됩니다. 따라서 --include 옵션에 예외를 지정할 경우 --exclude만 있으면 됩니다(예: --include는 실질적으로 "제외하지 않음"을 의미함). 다음 예제와 같이 이 옵션은 지정된 순서로 적용됩니다.

```
Local directory contains 3 files:
MyFile1.txt
MyFile2.rtf
MyFile88.txt
'''
$ aws s3 sync . s3://my-bucket/path --exclude '*.txt'
upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf
'''
$ aws s3 sync . s3://my-bucket/path --exclude '*.txt' --include 'MyFile*.txt'
upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt
upload: MyFile88.txt to s3://my-bucket/path/MyFile88.txt
upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf
'''
$ aws s3 sync . s3://my-bucket/path --exclude '*.txt' --include 'MyFile*.txt' --exclude
 'MyFile?.txt'
upload: MyFile2.rtf to s3://my-bucket/path/MyFile2.rtf
upload: MyFile88.txt to s3://my-bucket/path/MyFile88.txt
```

또한 `--exclude` 및 `--include` 옵션을 `--delete` 옵션과 함께 사용하면 동기화 작업 중에 삭제할 파일이나 객체를 필터링할 수 있습니다. 이 경우 파라미터 문자열은 대상 디렉터리 또는 버킷의 맥락에서 삭제에서 제외하거나 삭제를 위해 포함할 파일을 지정해야 합니다. 다음은 그 한 예입니다.

```
Assume local directory and s3://my-bucket/path currently in sync and each contains 3 files:
MyFile1.txt
MyFile2.rtf
MyFile88.txt
'''
// Delete local .txt files
$ rm *.txt

// Sync with delete, excluding files that match a pattern. MyFile88.txt is deleted, while
 remote MyFile1.txt is not.
$ aws s3 sync . s3://my-bucket/path --delete --exclude 'my-bucket/path/MyFile?.txt'
delete: s3://my-bucket/path/MyFile88.txt
'''
// Delete MyFile2.rtf
$ aws s3 rm s3://my-bucket/path/MyFile2.rtf

// Sync with delete, excluding MyFile2.rtf - local file is NOT deleted
$ aws s3 sync s3://my-bucket/path . --delete --exclude './MyFile2.rtf'
download: s3://my-bucket/path/MyFile1.txt to MyFile1.txt
'''
// Sync with delete, local copy of MyFile2.rtf is deleted
$ aws s3 sync s3://my-bucket/path . --delete
delete: MyFile2.rtf
```

또한 `sync` 명령에서는 `--acl` 옵션을 적용하여 Amazon S3에 복사되는 파일에 대한 액세스 권한을 설정할 수 있습니다. 이 옵션에는 `private`, `public-read` 및 `public-read-write` 값을 적용할 수 있습니다.

```
$ aws s3 sync . s3://my-bucket/path --acl public-read
```

앞에서 언급한 바와 같이, `s3` 명령 집합에는 `cp`, `mv`, `ls` 및 `rm`이 포함되며 이러한 명령은 해당 Unix 명령과 유사한 방식으로 작동합니다. 다음은 몇 가지 예입니다.

```
// Copy MyFile.txt in current directory to s3://my-bucket/path
$ aws s3 cp MyFile.txt s3://my-bucket/path/

// Move all .jpg files in s3://my-bucket/path to ./MyDirectory
$ aws s3 mv s3://my-bucket/path ./MyDirectory --exclude '*' --include '*.jpg' --recursive

// List the contents of my-bucket
```

```
$ aws s3 ls s3://my-bucket

// List the contents of path in my-bucket
$ aws s3 ls s3://my-bucket/path/

// Delete s3://my-bucket/path/MyFile.txt
$ aws s3 rm s3://my-bucket/path/MyFile.txt

// Delete s3://my-bucket/path and all of its contents
$ aws s3 rm s3://my-bucket/path --recursive
```

`--recursive` 옵션을 `cp`, `mv` 또는 `rm`과 함께 디렉터리/폴더에 사용하면 이 명령은 모든 하위 디렉터리를 포함한 디렉터리 트리를 표시합니다. 이 명령에는 `--exclude` 명령과 마찬가지로 `--include`, `--acl` 및 `sync` 옵션을 적용할 수 있습니다.

AWS Command Line Interface에서 API 수준(s3api)명령 사용

API 수준 명령(`s3api` 명령 집합에 포함됨)을 사용하면 Amazon S3 API에 직접 액세스할 수 있으며 상위 수준 명령에 표시되지 않는 일부 작업을 활성화할 수 있습니다. 이 단원에서는 API 수준 명령에 대해 설명하고 몇 가지 예제를 제공합니다. 추가 Amazon S3 예제는 s3api 명령줄 참조를 확인하고 목록에서 사용 가능한 명령을 선택하십시오.

사용자 지정 ACL

상위 수준 명령에서는 `--acl` 옵션을 사용하여 Amazon S3 객체에 대해 미리 정의된 ACL(액세스 제어 목록)을 적용할 수 있지만 버킷 전체 ACL을 설정할 수 없습니다. API 수준의 명령인 `put-bucket-acl`을 사용하여 이 작업을 수행할 수 있습니다. 다음 예제에서는 두 명의 AWS 사용자(user1@example.com 및 user2@example.com)에게 전체 제어 권한을 부여하고 모든 사람에게 읽기 권한을 부여합니다.

```
$ aws s3api put-bucket-acl --bucket MyBucket --grant-full-control
 'emailaddress="user1@example.com",emailaddress="user2@example.com"' --grant-read
 'uri="http://acs.amazonaws.com/groups/global/AllUsers"'
```

사용자 지정 ACL에 대한 자세한 내용은 PUT 버킷 ACL을 참조하십시오. `put-bucket-acl`과 같은 `s3api` ACL 명령은 동일한 간편 인수 표기법을 사용합니다.

로깅 정책

API 명령인 `put-bucket-logging`은 버킷 로깅 정책을 구성합니다. 다음 예제에서는 MyBucket에 대한 로깅 정책을 설정합니다. AWS 사용자 user@example.com은 로그 파일을 완전히 제어할 수 있으며 모든 사용자는 이 파일에 액세스할 수 있습니다. Amazon S3의 로그 전달 시스템에 필수 권한(write 및 read-acp)을 부여하려면 `put-bucket-acl` 명령이 필요하다는 점을 유의하십시오.

```
$ aws s3api put-bucket-acl --bucket MyBucket --grant-write 'URI="http://acs.amazonaws.com/
groups/s3/LogDelivery"' --grant-read-acp 'URI="http://acs.amazonaws.com/groups/s3/
LogDelivery"'
$ aws s3api put-bucket-logging --bucket MyBucket --bucket-logging-status file://
logging.json
```

logging.json

```
{
  "LoggingEnabled": {
    "TargetBucket": "MyBucket",
```

```
    "TargetPrefix": "MyBucketLogs/",
    "TargetGrants": [
      {
        "Grantee": {
          "Type": "AmazonCustomerByEmail",
          "EmailAddress": "user@example.com"
        },
        "Permission": "FULL_CONTROL"
      },
      {
        "Grantee": {
          "Type": "Group",
          "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
        },
        "Permission": "READ"
      }
    ]
  }
}
```

Amazon SNS와 함께 AWS Command Line Interface 사용

이 섹션에서는 Amazon Simple Notification Service (Amazon SNS)와 관련된 몇 가지 일반적인 작업 및 AWS Command Line Interface를 사용하여 이를 수행하는 방법에 대해 설명합니다.

항목

주제 생성

다음 명령은 **my-topic**라는 주제를 생성합니다.

```
$ aws sns create-topic --name my-topic
{
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

나중에 메시지를 게시할 때 사용하게 될 `TopicArn`을 적어 둡니다.

주제 구독

다음 명령은 알림 엔드포인트에 대한 이메일 프로토콜 및 이메일 주소를 사용하여 주제를 구독합니다.

```
$ aws sns subscribe --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --protocol
 email --notification-endpoint emailusername@example.com
{
    "SubscriptionArn": "pending confirmation"
```

```
}
```

구독 명령에 나열된 이메일 주소로 이메일 메시지가 전송됩니다. 이메일 메시지에는 다음 텍스트가 포함됩니다.

```
You have chosen to subscribe to the topic:
arn:aws:sns:us-west-2:123456789012:my-topic
To confirm this subscription, click or visit the following link (If this was in error no
 action is necessary):
Confirm subscription
```

구독 확인을 클릭하면 '구독이 확인되었습니다!'라는 알림 메시지가 브라우저에 다음과 유사한 정보와 함께 나타납니다.

```
Subscription confirmed!

You have subscribed emailusername@example.com to the topic:my-topic.

Your subscription's id is:
arn:aws:sns:us-west-2:123456789012:my-topic:1328f057-de93-4c15-512e-8bb2268db8c4

If it was not your intention to subscribe, click here to unsubscribe.
```

주제 게시

다음 명령은 주제에 메시지를 게시합니다.

```
$ aws sns publish --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --message "Hello
 World!"
{
    "MessageId": "4e41661d-5eec-5ddf-8dab-2c867a709bab"
}
```

'Hello World!' 문구가 포함된 이메일 메시지가 emailusername@example.com에 전송됩니다

주제에서 구독 취소

다음 명령은 주제에서 구독을 취소합니다.

```
$ aws sns unsubscribe --subscription-arn arn:aws:sns:us-west-2:123456789012:my-
topic:1328f057-de93-4c15-512e-8bb2268db8c4
```

주제에 대한 구독 취소를 확인하려면 다음을 입력합니다.

```
$ aws sns list-subscriptions
```

주제 삭제

다음 명령은 주제를 삭제합니다.

```
$ aws sns delete-topic --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic
```

주제의 삭제를 확인하려면 다음을 입력합니다.

```
$ aws sns list-topics
```

AWS Command Line Interface에서 Amazon Simple Workflow Service 사용

AWS CLI를 사용하여 Amazon Simple Workflow Service(Amazon SWF) 기능에 액세스할 수 있습니다.

명령 목록 및 Amazon SWF의 도메인과 작동하는 방법을 보려면 다음 주제를 참조하십시오.

항목

범주별 Amazon SWF 명령 목록

이 섹션에서는 AWS CLI의 Amazon SWF 명령에 대한 참조 항목을 나열합니다. 여기에서 명령은 기능 범주별로 나열됩니다.

명령의 알파벳 목록은 AWS CLI Command Reference의 Amazon SWF 섹션을 참조하거나 다음 명령을 사용하십시오.

```
$ aws swf help
```

특정 명령에 대한 도움말을 보려면 명령 이름 다음에 `help` 명령을 사용합니다. 다음은 그 한 예입니다.

```
$ aws swf register-domain help
```

항목

활동 관련 명령

활동 작업자는 `poll-for-activity-task`를 사용하여 새 활동 작업을 가져옵니다. 작업자가 Amazon SWF에서 활동 작업을 받은 후 작업을 수행하고 성공한 경우에는 `respond-activity-task-completed`, 실패한 경우에는 `respond-activity-task-failed`를 사용하여 응답합니다.

활동 작업자가 수행하는 명령은 다음과 같습니다.

- poll-for-activity-task
- respond-activity-task-completed
- respond-activity-task-failed
- respond-activity-task-canceled

- record-activity-task-heartbeat

결정자 관련 명령

결정자는 `poll-for-decision-task`를 사용하여 의사 결정 작업을 가져옵니다. 결정자가 Amazon SWF에서 의사 결정 작업을 받은 후 워크플로 실행 내역을 검사하고 다음에 수행할 작업을 결정합니다. 그리고 `respond-decision-task-completed`를 호출하여 의사 결정 작업을 완료하고 0개 이상의 다음 의사 결정을 제공합니다.

결정자가 수행하는 명령은 다음과 같습니다.

- poll-for-decision-task
- respond-decision-task-completed

워크플로우 실행 관련 명령

워크플로우 실행에서 작동하는 명령은 다음과 같습니다.

- request-cancel-workflow-execution
- start-workflow-execution
- signal-workflow-execution
- terminate-workflow-execution

관리 관련 명령

Amazon SWF 콘솔에서 관리 작업을 수행할 수 있지만 이 섹션의 명령을 사용하여 기능을 자동화하거나 고유한 관리 도구를 개발할 수 있습니다.

활동 관리

- register-activity-type
- deprecate-activity-type

워크플로우 관리

- register-workflow-type
- deprecate-workflow-type

도메인 관리

- register-domain
- deprecate-domain

도메인 관리 명령에 대한 예제 및 자세한 내용은 AWS Command Line Interface를 사용하여 Amazon SWF 도메인 작업 (p. 86) 항목을 참조하십시오.

워크플로우 실행 관리

- request-cancel-workflow-execution
- terminate-workflow-execution

가시성 명령

Amazon SWF 콘솔에서 가시성 작업을 수행할 수 있지만 이 섹션의 명령을 사용하여 고유한 콘솔 또는 관리 도구를 개발할 수 있습니다.

활동 가시성

- list-activity-types
- describe-activity-type

워크플로우 가시성

- list-workflow-types
- describe-workflow-type

워크플로우 실행 가시성

- describe-workflow-execution
- list-open-workflow-executions
- list-closed-workflow-executions
- count-open-workflow-executions
- count-closed-workflow-executions
- get-workflow-execution-history

도메인 가시성

- list-domains
- describe-domain

이러한 도메인 가시성 명령에 대한 예제 및 자세한 내용은 AWS Command Line Interface를 사용하여 Amazon SWF 도메인 작업 (p. 86) 항목을 참조하십시오.

작업 목록 가시성

- count-pending-activity-tasks
- count-pending-decision-tasks

AWS Command Line Interface를 사용하여 Amazon SWF 도메인 작업

이 섹션에서는 AWS CLI를 사용하여 범용 Amazon SWF 도메인 작업을 수행하는 방법을 설명합니다.

항목

도메인 나열

계정에 등록한 Amazon SWF 도메인을 나열하려면 `swf list-domains`를 사용할 수 있습니다. 필수 파라미터가 단 하나 있으며 `--registration-status`입니다. 이 파라미터를 `REGISTERED` 또는 `DEPRECATED`로 설정할 수 있습니다.

다음은 최소 예입니다.

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

Note

`DEPRECATED` 사용 예는 도메인 사용 안 함 (p. 89) 단원을 참조하십시오. 이 명령은 가지고 있는 도메인 중에서 사용되지 않는 도메인을 반환합니다.

페이지 크기 설정을 통한 결과 제한

도메인이 많은 경우 `--maximum-page-size` 파라미터를 설정하여 반환되는 결과의 수를 제한할 수 있습니다. 지정한 최대 수보다 많은 결과가 나오면 `nextPageToken`에 대한 다음 호출로 전송하여 추가 항목을 검색할 수 있는 `list-domains`을 수신합니다.

다음은 `--maximum-page-size` 사용 예입니다.

```
$ aws swf list-domains --registration-status REGISTERED --maximum-page-size 1
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        }
    ],
    "nextPageToken": "ANeXAMPLEtOKENiSpRETTYlONG=="
}
```

Note

반환되는 `nextPageToken`은 훨씬 더 깁니다. 이 값은 설명을 돕기 위한 예제일 뿐입니다.

`nextPageToken` 인수에 `--next-page-token` 값을 입력하여 호출을 다시 수행하면 다른 결과 페이지가 나옵니다.

```
$ aws swf list-domains --registration-status REGISTERED --maximum-page-size 1 --next-page-
token "ANeXAMPLEtOKENiSpRETTYlONG=="
{
    "domainInfos": [
        {
            "status": "REGISTERED",
```

```
            "name": "mytest"
        }
    ]
}
```

더 이상 가져올 결과 페이지가 없으면 `nextPageToken`이 결과에 반환됩니다.

도메인에 대한 정보 보기

특정 도메인에 대한 자세한 정보를 보려면 `swf describe-domain`을 사용합니다. 필수 파라미터가 하나 있으며 `--name`입니다. 이 파라미터는 정보를 보려고 하는 도메인의 이름을 가져옵니다. 예:

```
$ aws swf describe-domain --name ExampleDomain
{
    "domainInfo": {
        "status": "REGISTERED",
        "name": "ExampleDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "1"
    }
}
```

도메인 등록

새 도메인을 등록하려면 `swf register-domain`을 사용합니다. 필수 파라미터 두 개가 있습니다. `--name`은 도메인 이름을 가져오고 `--workflow-execution-retention-period-in-days`는 이 도메인에서 워크플로우 실행 데이터를 유지하는 기간(일)을 최대 90일까지 지정할 정수를 가져옵니다(자세한 내용은 Amazon SWF FAQ 참조). 이 값에 0을 지정하면 유지 기간이 최대 기간으로 지동 설정됩니다. 그렇지 않으면 지정된 기간이 경과한 후에는 워크플로우 실행 데이터가 유지되지 않습니다.

다음은 새 도메인을 등록하는 예입니다.

```
$ aws swf register-domain --name MyNeatNewDomain --workflow-execution-retention-period-in-
days 0
```

도메인을 등록하면 아무 것도 반환되지 않지만("") `swf list-domains` 또는 `swf describe-domain`을 사용하여 새 도메인을 볼 수 있습니다. 예:

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "MyNeatNewDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

다음은 `swf describe-domain` 사용 예입니다.

```
$ aws swf describe-domain --name MyNeatNewDomain
{
    "domainInfo": {
        "status": "REGISTERED",
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
    }
}
```

도메인 사용 안 함

도메인을 사용하지 않도록 설정하려면(도메인을 여전히 볼 수 있지만 새 워크플로우 실행을 생성하거나 유형을 등록할 수 없음) `swf deprecate-domain`을 사용합니다. 필수 파라미터가 하나 있으며 `--name`입니다. 이 파라미터는 사용하지 않도록 설정할 도메인의 이름을 가져옵니다.

```
$ aws swf deprecate-domain --name MyNeatNewDomain
```

`register-domain`과 마찬가지로 출력이 반환되지 않습니다. 하지만 `list-domains`를 사용하여 등록된 도메인을 보면 그 중에 해당 도메인이 더 이상 나타나지 않는 것을 알 수 있습니다.

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

`--registration-status DEPRECATED`를 `list-domains`와 함께 사용하여 사용되지 않는 도메인을 확인할 수 있습니다.

```
$ aws swf list-domains --registration-status DEPRECATED
{
    "domainInfos": [
        {
            "status": "DEPRECATED",
            "name": "MyNeatNewDomain"
        }
    ]
}
```

또한 `describe-domain`을 사용하여 사용되지 않는 도메인에 대한 정보를 가져올 수 있습니다.

```
$ aws swf describe-domain --name MyNeatNewDomain
{
    "domainInfo": {
        "status": "DEPRECATED",
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
```

```
    }
}
```

참고 항목

- AWS CLI Command Reference의 deprecate-domain
- AWS CLI Command Reference의 describe-domain
- AWS CLI Command Reference의 list-domains
- AWS CLI Command Reference의 register-domain

AWS CLI 오류 문제 해결

pip를 사용하여 설치한 후 aws 실행 파일을 OS의 PATH 환경 변수에 추가하거나 모드를 변경하여 이 파일을 실행 파일로 만들어야 할 수 있습니다.

오류: aws: 명령을 찾을 수 없음

aws 실행 파일을 OS의 PATH 환경 변수에 추가해야 할 수 있습니다.

- Windows – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 11)
- macOS – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 13)
- Linux – 명령줄 경로에 AWS CLI 실행 파일 추가 (p. 7)

aws가 PATH에 있는데도 이 오류가 표시되는 경우, 올바른 파일 모드가 아닐 수 있습니다. 직접 실행해 보십시오.

```
$ ~/.local/bin/aws --version
```

오류: 권한 거부

aws 스크립트가 실행 가능한 파일 모드인지 확인합니다. 예, 755.

chmod +x를 실행하여 파일을 실행 파일로 만듭니다.

```
$ chmod +x ~/.local/bin/aws
```

오류: AWS에서 제공된 자격 증명을 검증할 수 없음

AWS CLI가 예상한 것과 다른 위치에서 자격 증명을 읽어올 수 있습니다. aws configure list를 실행하여 올바른 자격 증명을 사용하고 있는지 확인합니다.

```
$ aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************XYVA shared-credentials-file
secret_key     ****************ZAGY shared-credentials-file
    region                 us-west-2      config-file    ~/.aws/config
```

올바른 자격 증명을 사용 중이라면 클록이 동기화되지 않았을 수 있습니다. Linux, macOS, or Unix에서는 data를 실행하여 시간을 확인하십시오.

```
date
```

시스템 클록이 꺼져 있는 경우, ntpd로 동기화합니다 it.

```
sudo service ntpd stop
sudo ntpdate time.nist.gov
sudo service ntpd start
ntpstat
```

Windows에서는 제어판의 날짜 및 시간 옵션을 사용하여 시스템 클록을 구성합니다.

오류: *CreateKeyPair* 작업 호출 중 오류 발생(UnauthorizedOperation): 이 작업을 수행할 수 있는 권한이 없습니다.

AWS CLI에서 실행하는 명령에 해당하는 API 작업을 호출할 권한이 IAM 사용자 또는 역할에게 있어야 합니다. 대부분의 명령은 명령 이름에 해당하는 이름으로 한 가지 작업을 호출합니다. 그러나 `aws s3 sync` 같은 사용자 지정 명령은 API 여러 개를 호출합니다. `--debug` 옵션으로 명령이 어떤 API를 호출하는지 확인할 수 있습니다.