AWS Command Line Interface

ユーザーガイド

AWS Command Line Interface: ユーザーガイド

Table of Contents

AWS Command Line Interface とは

AWS Command Line Interface (AWS CLI) は、コマンドラインシェルでコマンドを使用して AWS サービスとやり取りするためのオープンソースツールです。最小限の設定で、使い慣れたターミナルプログラムのコマンドプロンプトから、ブラウザベースの AWS マネジメントコンソール で提供される機能と同等の機能の使用を開始できます。

- Linux シェル – `bash`、`zsh`、`tsch` などの一般的なシェルプログラムを使用して、Linux, macOS, or Unix でコマンドを実行します。
- Windows コマンドライン – Windows で、PowerShell または Windows コマンドプロンプトでコマンドを実行します。
- リモート – PuTTY や SSH などのリモートターミナルを通じて、または AWS Systems Manager システムマネージャーを使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスでコマンドを実行します。

AWS マネジメントコンソール で使用できる AWS IaaS (infrastructure as a Service) の管理、運用、アクセスの全機能は、AWS API と CLI でも使用できます。AWS マネジメントコンソール で使用できる AWS IaaS の新しい機能およびサービスは、開始時、または開始から 180 日以内に、API と CLI からも使用できます。

AWS CLI では、AWS のサービスのパブリック API に直接アクセスできます。AWS CLI を使用してサービスの機能を調べ、シェルスクリプトを開発してリソースを管理できます。または、AWSSDK を使用して、他の言語でのプログラム開発で得た経験を活用することもできます。

低レベルの同等の API コマンドに加えて、複数の AWS サービスでは AWS CLI のカスタマイズを提供します。カスタマイズには、複雑な API によるサービスの使用を簡略化する高レベルのコマンドが含まれます。たとえば、Amazon Simple Storage Service (Amazon S3) の一連のコマンドでは、使い慣れた構文を使用して `aws s3` のファイルを管理できます。

Example Amazon S3 へのファイルのアップロード

`aws s3 cp` はシェルに類似したコピーコマンドを提供し、マルチパートアップロードを自動的に実行して、大きなファイルをすばやく復元力の高い方法で転送します。

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

同じタスクを低レベルのコマンド (`aws s3api` で使用可能) で実行すると、はるかに多くの労力がかかります。

ユースケースによっては、AWS SDK のいずれか、または AWS Tools for PowerShell を使用することをお勧めします。

- AWS Tools for PowerShell
- AWS SDK for Java
- AWS SDK for .NET

  AWS SDK for JavaScript
- AWS SDK for Ruby
- AWS SDK for Python (Boto)

- AWS SDK for PHP
- AWS SDK for Go
- AWS Mobile SDK for iOS
- AWS Mobile SDK for Android

GitHub の aws-cli レポジトリで、AWS CLI のソースコードを表示 (およびフォーク) できます。GitHub でユーザーのコミュニティに参加して、フィードバックを提供したり、機能をリクエストしたり、独自の投稿を提出したりしてください。

このガイドにある例を使用する

このガイドの例は、次の規則に従った形式となります。

- プロンプト – コマンドプロンプトはドル記号にスペースを追加した形式 (「'$ '」) で表示されます。コマンドを入力した場合はプロンプトを含めないでください。
- ディレクトリ – 特定のディレクトリからコマンドを実行する必要がある場合は、プロンプト記号の前にディレクトリ名が表示されます。
- ユーザー入力 – コマンドラインに入力する必要があるコマンドテキストは、**`user input`** の形式となります。
- 置き換え可能なテキスト – 選択したリソースの名前、またはコマンドに含める必要のある AWS のサービスによって生成された ID を含む変更可能なテキストで、*`replaceable text`* の形式となります。複数行のコマンドまたは特定のキーボード入力が必要なコマンドの場合、キーボードコマンドも置き換え可能なテキストとして表示できます。
- 出力 – AWS サービスによって返される出力は、`computer output` の形式でユーザー入力の下に表示されます。

たとえば、次のコマンドには、ユーザー入力、置き換え可能なテキスト、出力が含まれています。

```
$ 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
```

この例を使用するには、コマンドラインに「**`aws configure`**」と入力し、Enter キーを押します。**`aws configure`** はコマンドです。このコマンドはインタラクティブであるため、AWS CLI はテキストの行を出力し、追加情報の入力が求められます。各アクセスキーを順に入力して、[Enter] キーを押します。その後、表示されている形式で AWS リージョン名を入力して Enter キーを押し、最後に Enter キーをもう一度押して出力形式の設定をスキップします。最後の Enter コマンドは、その行にユーザー入力がないため、置き換え可能なテキストとして表示されます。それ以外の場合は、暗黙的に示されます。

以下の例では、JSON 形式でサービスからの出力がある単純な非インタラクティブコマンドを示します。

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

この例を使用するには、コマンドのフルテキスト (プロンプトの後の強調表示されたテキスト) を入力し、Enter キーを押します。セキュリティグループの名前 *`my-sg`* は、置き換え可能です。この場合は、表示されているようにグループ名を使用できますが、分かりやすい名前を使用する方がよいかもしれません。

Note

必ず置き換える必要のある引数 (AWS Access Key ID など) と、置き換えるべき引数 (group name など) はいずれも、###############として表示されます。引数を置き換える必要がある場合は、例を示すテキストにその旨が記載されています。

JSON ドキュメントは中括弧を含めて出力です。CLI を TEXT または TABLE 形式で出力するように設定した場合、出力は異なる形式になります。JSON がデフォルトの出力形式です。

アマゾン ウェブ サービスについて

アマゾン ウェブ サービス (AWS) は、開発者がアプリケーションの開発時に利用できるデジタルインフラストラクチャサービスの集合体です。サービスには、演算能力、ストレージ、データベース、アプリケーション同期 (メッセージングとキューイング) があります。AWS は従量制サービスモデルを採用しています。料金が発生するのは、ユーザー (すなわちユーザのアプリケーション) が実際に使用したサービスの分のみです。また、AWS をプロトタイプと実験用のプラットフォームとして利用しやすくするために、AWS には無料利用枠も用意されています。この枠では、サービスを利用しても一定のレベル以下であれば無料です。AWS のコストと無料利用枠に関する詳細については、「無料利用枠による AWS のテスト」を参照してください。AWS アカウントを取得するには、AWS ホームページを開き、[Sign Up] をクリックします。

AWS CLI のインストール

AWS Command Line Interface (AWS CLI) をインストールする方法

前提条件

- Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+
- Windows、Linux, macOS, or Unix

> **Note**
>
> 以前のバージョンの Python は、AWS のすべてのサービスで動作しない可能性があります。AWS CLI をインストールまたは使用するときに、`InsecurePlatformWarning` または廃止の通知が表示された場合は、新しいバージョンに更新します。

pip を使用した AWS CLI のインストール

Linux、Windows、および MacOS 上の AWS CLI の主なディストリビューションは `pip` です。これは、Python パッケージとその依存関係を簡単にインストール、アップグレード、および削除するための Python パッケージマネージャです。

現行の AWS CLI バージョン

AWS CLI 新しいサービスおよびコマンドをサポートするよう頻繁に更新されます。最新バージョンがあるかどうかを確認するには、「GitHub のリリースページ」を参照してください。

`pip` およびサポートされるバージョンの Python がすでにインストールされている場合は、以下のコマンドを使用して、AWS CLI をインストールできます。

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

`--upgrade` オプションでは、すでにインストールされている要件をアップグレードするよう に指示します。`pip--user` オプションでは、オペレーティングシステムによって使用されるライブラリの変更を避けるために、ユーザーディレクトリのサブディレクトリにプログラムをインストールするよう `pip` に指示します。

仮想環境への AWS CLI のインストール

`pip` で AWS CLI のインストールを試みたときに依存関係の問題が発生した場合は、仮想環境に AWS CLI をインストール (p. 15)してツールとその依存関係を隔離します。または、通常使用しているものと異なるバージョンの Python を使用することもできます。

インストーラを使用した AWS CLI のインストール

Linux, macOS, or Unix でオフラインまたは自動インストールをするには、バンドルされたインストーラ (p. 16)を試してください。バンドルされたインストーラには、AWS CLI、その依存関係、およびインストールを実行するシェルスクリプトが含まれます。

Windows では、MSI インストーラ (p. 10)も使用できます。これらの方法のいずれも、初回のインストールを簡単にします。ただし、AWS CLI の新しいバージョンがリリースされた際のアップグレードはより難しくなります。

インストール後に実行する手順

AWS CLI をインストールしたら、`PATH` 変数への実行可能ファイルのパスの追加が必要になる場合があります。プラットフォーム固有の手順については、以下のトピックを参照してください。

- Linux – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 7)
- Windows – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 12)
- macOS – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 15)

`aws --version` を実行することで、AWS CLI が正しくインストールされたことを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59-amzn2.x86_64 botocore/1.12.106
```

AWS CLI は新しいサービスとコマンドのサポートを追加するために、定期的に更新されます。AWS CLI を最新バージョンに更新するには、インストールコマンドを再び実行します。AWS CLI の最新バージョンについては、「AWS CLI リリースノート」を参照してください。

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

AWS CLI をアンインストールする必要がある場合は、`pip uninstall` を使用します。

```
$ pip uninstall awscli
```

Python と `pip` がインストールされていない場合は、使用している環境に応じた手順に従ってください。

各環境の詳細な手順

- Linux に AWS CLI をインストールする (p. 5)
- Windows に AWS CLI をインストールする (p. 10)
- macOS に AWS CLI をインストールする (p. 12)
- 仮想環境に AWS CLI をインストールする (p. 15)
- バンドルされたインストーラを使用して AWS CLI をインストールする (Linux, macOS, or Unix) (p. 16)

Linux に AWS CLI をインストールする

AWS Command Line Interface (AWS CLI) とその依存関係は、Python 用のパッケージマネージャーである `pip` を使用して、ほとんどの Linux ディストリビューションでインストールできます。

Important

awscli パッケージは apt や yum など他のパッケージマネージャー用にリポジトリで利用可能ですが、pip から入手するか、バンドルインストーラ (p. 16)を使用しない限り、最新バージョンであることは保証されません。

pip がすでに存在する場合は、メインのインストールに関するトピック (p. 4)の手順に従います。pip --version を実行して、Linux のバージョンにすでに Python と pip が含まれているかどうか確認します。

```
$ pip --version
```

pip がない場合は、インストールされている Python のバージョンを確認します。

```
$ python --version
```

または

```
$ python3 --version
```

まだ Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+ を持っていない場合は、まず、Python をインストールする (p. 8)必要があります。Python がすでにインストールされている場合は、pip および AWS CLI のインストールに進みます。

セクション

- pip のインストール (p. 6)
- pip を使用して AWS CLI をインストールします。 (p. 7)
- AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 7)
- Linux での Python のインストール (p. 8)
- Amazon Linux に AWS CLI をインストールする (p. 9)

pip のインストール

pip がインストールされていない場合は、Python Packaging Authority が提供するスクリプトを使用してインストールできます。

pip をインストールするには

1. curl コマンドを使用してインストールスクリプトをダウンロードします。

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

2. Python でスクリプトを実行して、pip の最新バージョンとその他の必要なサポートパッケージをダウンロードしてインストールします。

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

   --user スイッチを含めると、スクリプトは pip をパス ~/.local/bin にインストールします。

3. pip のパスが、PATH 変数の一部であることを確認します。

   a. ユーザーフォルダーでシェルのプロファイルスクリプトを見つけます。現在使用しているシェルが不明な場合は、echo $SHELL を実行します。

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

- Bash – .bash_profile、.profile、または .bash_login。
- Zsh – .zshrc。
- Tcsh – .tcshrc、.cshrc、または .login。

b. 次の例のように、プロファイルスクリプトの末尾にエクスポートコマンドを追加します。

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

このコマンドでは、パス (この例では ~/.local/bin) が、既存の PATH 変数の前に挿入されます。

c. 変更が有効になるように、プロファイルを現在のセッションに再ロードします。

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

4. これで、pip が正しくインストールされたことを確認するためにテストすることができます。

```
$ pip --version
pip 19.0.3 from ~/.local/lib/python3.7/site-packages (python 3.7)
```

pip を使用して AWS CLI をインストールします。

pip を使用して AWS CLI をインストールします。

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

--user スイッチを使用すると、pip は AWS CLI を ~/.local/bin にインストールします。

AWS CLI が正しくインストールされたことを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59-amzn2.x86_64 botocore/1.12.106
```

エラーが表示される場合は、「AWS CLI エラーのトラブルシューティング (p. 99)」を参照してください。

最新バージョンにアップグレードするには、インストールコマンドを再び実行します。

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

AWS CLI 実行ファイルをコマンドラインパスに追加する

pip を使用してインストールした後は、オペレーティングシステムの aws 環境変数への PATH 実行ファイルの追加が必要になる場合があります。

pip によって AWS CLI をインストールしたフォルダを確認するには、次のコマンドを実行します。

```
$ which aws
/home/username/.local/bin/aws
```

/home/username は Linux では ~ に対応するため、これを ~/.local/bin/ として参照できます。

--user スイッチを省いて、ユーザーモードでインストールしなかった場合、実行可能ファイルは Python のインストール先の bin フォルダにあります。Python がインストールされた場所が不明な場合は、次のコマンドを実行します。

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

出力は、実際の実行可能ファイルではなく symlink へのパスになる場合があります。ls -al を実行して、その参照先を確認します。

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

pip のインストール (p. 6) のステップ 3 でパスに追加したのと同じフォルダである場合、作業は完了です。それ以外の場合は、ステップ 3a – 3c を再び実行して、この追加フォルダをパスに追加します。

Linux での Python のインストール

ご使用のディストリビューションに Python が付属していないか、以前のバージョンである場合は、pip および AWS CLI をインストールする前に Python をインストールします。

Linux に Python 3 をインストールするには

1. Python がインストールされているかどうかを確認します。

   ```
   $ python --version
   ```

   Note

   ご使用の Linux ディストリビューションに Python が付属している場合、拡張機能のコンパイルや AWS CLI のインストールで必要となるヘッダーとライブラリを取得するために、Python 開発者パッケージのインストールが必要になることがあります。パッケージマネージャーを使用して、開発者パッケージ (名前は通常 python-dev または python-devel) をインストールします。
2. Python 2.7 以降がインストールされていない場合は、ご使用のディストリビューションのパッケージマネージャーを使用して Python をインストールします。コマンドとパッケージ名は、場合によって異なります。

   - Debian から派生した OS (Ubuntu など) では、apt を使用します。

     ```
     $ sudo apt-get install python3
     ```
   - Red Hat およびそれから派生した OS では、yum を使用します。

     ```
     $ sudo yum install python
     ```
   - SUSE およびそれから派生した OS では、zypper を使用します。

     ```
     $ sudo zypper install python3
     ```

3. コマンドプロンプトまたはシェルを開き、次のコマンドを実行して、Python が正しくインストールされたことを確認します。

```
$ python --version
Python 3.6.1
```

Amazon Linux に AWS CLI をインストールする

AWS Command Line Interface (AWS CLI) は Amazon Linux と Amazon Linux 2 にプリインストールされています。以下のコマンドを使用して、現在インストールされているバージョンを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59.amzn2.x86_64 botocore/1.12.106
```

`sudo yum update` を使用して、yum レポジトリで使用可能な最新バージョンを取得できますが、これは最新バージョンではない場合があります。代わりに、`pip` を使用して最新バージョンを取得することをお勧めします。

前提条件

Python と `pip` がすでにインストールされていることを確認します。詳細については、「Linux に AWS CLI をインストールする (p. 5)」を参照してください。

Amazon Linux で AWS CLI をアップグレードするには (root)

1. `pip install` を使用して、AWS CLI の最新バージョンをインストールします。

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

2. `aws --version` で新しいバージョンを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59.amzn2.x86_64 botocore/1.12.106
```

ルート権限がない場合、ユーザーモードで AWS CLI をインストールします。

Amazon Linux で AWS CLI をアップグレードするには (user)

1. `pip install` を使用して、AWS CLI の最新バージョンをインストールします。

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

2. インストール場所を `PATH` 変数の先頭に追加します。

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

`~/.bashrc` の末尾にこのコマンドを追加し、セッション間の変更を維持します。

3. `aws --version` で新しいバージョンを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59.amzn2.x86_64 botocore/1.12.106
```

Windows に AWS CLI をインストールする

Windows に AWS Command Line Interface (AWS CLI) をインストールするには、スタンドアロンインストーラ、または Python のパッケージマネージャーである `pip` を使用します。`pip` がすでに存在する場合は、メインのインストールに関するトピック (p. 4)の手順に従います。

セクション

MSI インストーラを使用して AWS CLI をインストールする

AWS CLI は、Microsoft Windows XP 以降でサポートされています。Windows ユーザーの場合、MSI インストールパッケージを使用すれば、その他の必要なものをインストールすることなく、使い慣れた便利な方法で AWS CLI をインストールできます。

更新がリリースされたときは、最新バージョンの AWS CLI を取得するため、インストールプロセスを繰り返す必要があります。頻繁に更新する場合は、更新を容易にするために pip の使用 (p. 11)を検討してください。

MSI インストーラを使用して AWS CLI をインストールするには

1. 適切な MSI インストーラをダウンロードします。

   - Windows (64 ビット) 用 の AWS CLI MSI インストーラのダウンロード
   - Windows (32 ビット) 用 の AWS CLI MSI インストーラのダウンロード
   - AWS CLI セットアップファイルのダウンロード (32 ビットと 64 ビット両方の MSI インストーラが含まれており、適切なバージョンが自動的にインストールされます)

   > **Note**
   >
   > AWS CLI の MSI インストーラは Windows Server 2008 (バージョン 6.0.6002) では機能しません。このバージョンの Windows Server にインストールするには、pip (p. 11) を使用します。

2. ダウンロードした MSI インストーラまたは設定ファイルを実行します。
3. 画面上の指示に従います。

CLI はデフォルトでは `C:\Program Files\Amazon\AWSCLI`(64 ビットバージョン) または `C:\Program Files (x86)\Amazon\AWSCLI` (32 ビットバージョン) にインストールされます。インストールを確認するには、コマンドプロンプトで `aws --version` コマンドを使用します (コマンドプロンプトを開始するには、[スタート] メニューを開いて "cmd" を検索します)。

```
C:\> aws --version
aws-cli/1.16.116 Python/3.6.8 Windows/10 botocore/1.12.106
```

コマンドを入力するときは、プロンプト記号 (上記の「`C:\>`」) を含めないでください。これらは、入力したコマンドと CLI から返された出力を区別するためにプログラムのリストに含まれています。このガイドの残りの部分では、コマンドが Windows 固有の場合を除き、一般的なプロンプト記号 "$" を使用します。

Windows がプログラムを見つけることができない場合、パスを読み込み直すためにコマンドプロンプトを閉じて再度開くか、または手動でインストールディレクトリを PATH 環境変数に追加 (p. 12)する必要が生じることもあります。

MSI のインストールを更新する

AWS CLI は定期的に更新されます。GitHub のリリースページで、最新バージョンがいつリリースされたかを確認してください。最新バージョンに更新するには、前述のように、再度 MSI インストーラをダウンロードして実行します。

AWS CLI のアンインストール

AWS CLI をアンインストールするには、コントロールパネルを開き、[プログラムと機能] を選択します。[AWS Command Line Interface] という名前のエントリを選択後、[アンインストール] を選択してアンインストーラを起動します。プロンプトが表示されたら、AWS CLI をアンインストールすることを確認します。

次のコマンドを使用してコマンドラインから [プログラムと機能] プログラムを起動することもできます。

```
C:\> appwiz.cpl
```

Windows で Python と pip を使用して AWS CLI をインストールする

Python Software Foundation は、pip を含む Windows 用インストーラを提供しています。

Python と pip をインストールするには (Windows)

1. Python.org のダウンロードページから Python Windows x86-64 のインストーラをダウンロードします。
2. インストーラを実行します。
3. [Add Python 3 to PATH] を選択します。
4. [Install Now] を選択します。

インストーラはユーザーフォルダに Python をインストールし、プログラムフォルダをユーザーパスに追加します。

pip を使用して AWS CLI をインストールするには (Windows)

1. [スタート] メニューから [コマンドプロンプト] を開きます。
2. 次のコマンドを使用して、Python と pip のいずれも正しくインストールされたことを確認します。

   ```
   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 を使用して AWS CLI をインストールする

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

4. AWS CLI が正しくインストールされたことを確認します。

   ```
   C:\Windows\System32> aws --version
   ```

```
aws-cli/1.16.116 Python/3.6.8 Windows/10 botocore/1.12.106
```

最新バージョンにアップグレードするには、インストールコマンドを再び実行します。

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

AWS CLI 実行ファイルをコマンドラインパスに追加する

pip を使用して AWS CLI インストールしたら、オペレーティングシステムの PATH 環境変数への aws プログラムの追加が必要になる場合があります。MSI がインストールされている場合、これは自動的に行われますが、インストール後に aws コマンドが実行しない場合は、手動で設定しなければならないことがあります。

次のコマンドを実行すると、aws プログラムがインストールされた場所を確認できます。

```
C:\> where aws
C:\Program Files\Python37\Scripts\aws
```

このコマンドで結果が返らない場合は、手動でパスを追加する必要があります。コマンドラインまたは Windows Explorer を使用して、コンピュータのインストールされている場所を見つけます。一般的なパスは、次のとおりです。

- Python 3 および **pip** – `C:\Program Files\Python37\Scripts\`
- Python 3 および **pip** --user オプション (以前の Windows バージョン) – `%USERPROFILE%\AppData\Local\Programs\Python\Python37\Scripts`
- Python 3 および **pip** --user オプション (Windows 10) – `%USERPROFILE%\AppData\Roaming\Python\Python37\Scripts`
- MSI インストーラ (64 ビット) – `C:\Program Files\Amazon\AWSCLI`
- MSI インストーラ (32 ビット) – `C:\Program Files (x86)\Amazon\AWSCLI`

> **Note**
>
> バージョン番号が含まれるフォルダ名は、異なる場合があります。上記の例は、Python37 を示します。必要に応じて、使用しているバージョン番号に置き換えます。

PATH 変数を変更するには (Windows)

1. Windows キーを押し、「**environment variables**」と入力します。
2. [Edit environment variables for your account] を選択します。
3. PATH を選択して、編集 を選択します。
4. このパスを [変数値] フィールドに追加します。以下に例を示します。***C:\new\path***
5. [OK] を 2 回選択して、新しい設定を適用します。
6. 実行中のコマンドプロンプトを閉じ、コマンドプロンプトウィンドウを再度開きます。

macOS に AWS CLI をインストールする

macOS に AWS Command Line Interface (AWS CLI) をインストールする方法としては、バンドルされたインストーラを使用することが推奨されます。バンドルされたインストーラにはすべての依存関係が含まれており、オフラインで使用できます。

Important

バンドルされたインストーラでは、スペースを含むパスへのインストールはサポートされていません。

セクション

前提条件

- Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+

Python のインストールを確認します。

```
$ python --version
```

ご使用のコンピュータに Python がインストールされていない場合、または別のバージョンの Python をインストールする場合は、「Linux に AWS CLI をインストールする (p. 5)」の手順に従います。

バンドルされたインストーラを使用して AWS CLI をインストールする

コマンドラインから以下の手順に従い、バンドルされたインストーラを使用して AWS CLI をインストールします。

バンドルされたインストーラを使用して AWS CLI をインストールするには

1. AWS CLI のバンドルされたインストーラをダウンロードします。

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

2. パッケージを解凍します。

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

   Note

   `unzip` がない場合は、Linux ディストリビューションに組み込まれたパッケージマネージャーを使用してインストールします。

3. インストールプログラムを実行します。

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

   Note

   デフォルトでは、インストールスクリプトはシステムのデフォルトバージョンの Python で実行されます。インストールされている別のバージョンの Python を使用して AWS CLI をイン

ストールする場合は、Python アプリケーションへの絶対パスを使用してそのバージョンを指定することにより、インストールスクリプトを実行します。以下に例を示します。

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

このコマンドは AWS CLI を `/usr/local/aws` にインストールし、シンボリックリンク `aws` を `/usr/local/bin` ディレクトリに作成します。`-b` オプションを使用してシンボリックリンクを作成すると、ユーザーの `$PATH` 変数にインストールディレクトリを指定する必要がなくなります。これにより、すべてのユーザーが任意のディレクトリから `aws` を入力して AWS CLI を呼び出せるようになります。

`-i` オプションおよび `-b` オプションの説明を表示するには、`-h` オプションを使用します。

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

pip を使用して macOS に AWS CLI をインストールする

`pip` を直接使用して AWS CLI をインストールすることもできます。`pip` がない場合は、メインの「インストールに関するトピック (p. 4)」の手順に従います。`pip --version` を実行して、macOS のバージョンにすでに Python と `pip` が含まれているかどうか確認します。

```
$ pip --version
```

macOS で AWS CLI をインストールするには

1. Python.org のダウンロードページから Python 3.6 をダウンロードしてインストールします。
2. Python Packaging Authority が提供する `pip` インストールスクリプトをダウンロードして実行します。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   $ python3 get-pip.py --user
   ```

3. 新しくインストールした `pip` を使用して AWS CLI をインストールします。

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

4. AWS CLI が正しくインストールされたことを確認します。

   ```
   $ aws --version
   AWS CLI 1.16.116 (Python 3.7.1)
   ```

   プログラムが見つからない場合は、そのプログラムをコマンドラインパスに追加 (p. 15)します。

最新バージョンにアップグレードするには、インストールコマンドを再び実行します。

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

AWS CLI 実行ファイルをコマンドラインパスに追加する

pip を使用してインストールした後は、オペレーティングシステムの PATH 環境変数への aws プログラムの追加が必要になる場合があります。プログラムの場所は、Python のインストール先によって異なります。

Example AWS CLI のインストール場所 - Python 3.7 および **pip** のある macOS (ユーザーモード)

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

Python のインストール先がわからない場合は、which python を実行します。

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

出力は、実際のプログラムではなくシンボリックリンクへのパスになる場合があります。ls -al を実行して、その参照先を確認します。

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

pip は、Python プログラムが含まれている同じフォルダに、アプリケーションをインストールします。このフォルダを PATH 変数に追加します。

PATH 変数を変更するには (Linux, macOS, or Unix)

1. ユーザーフォルダーでシェルのプロファイルスクリプトを見つけます。現在使用しているシェルが不明な場合は、echo $SHELL を実行します。

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

   - Bash – .bash_profile、.profile、または .bash_login。
   - Zsh – .zshrc。
   - Tcsh – .tcshrc、.cshrc、または .login。
2. プロファイルスクリプトにエクスポートコマンドを追加します。

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

   このコマンドは、現在の PATH 変数にパス (この例では ~/.local/bin) を追加します。
3. 現在のセッションにプロファイルをロードします。

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

仮想環境に AWS CLI をインストールする

仮想環境に AWS Command Line Interface (AWS CLI) をインストールすることで、他の pip パッケージとのバージョンに関する要件の競合を回避できます。

仮想環境に AWS CLI をインストールするには

1. pip を使用して virtualenv をインストールします。

```
$ pip install --user virtualenv
```

2. 仮想環境を作成して名前を付けます。

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

または、-p オプションを使用してデフォルト以外のバージョンの Python を指定できます。

```
$ virtualenv -p /usr/bin/python3.4 ~/cli-ve
```

3. 新しい仮想環境をアクティブ化します。

Linux, macOS, or Unix

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

Windows

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

4. 仮想環境に AWS CLI をインストールします。

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

5. AWS CLI が正しくインストールされたことを確認します。

```
$ aws --version
aws-cli/1.16.116 Python/3.6.8 Linux/4.14.77-81.59-amzn2.x86_64 botocore/1.12.106
```

deactivate コマンドを使用して、仮想環境を終了できます。新しいセッションを開始するたびに、環境を再度アクティブ化する必要があります。

最新バージョンにアップグレードするには、インストールコマンドを再び実行します。

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

バンドルされたインストーラを使用して AWS CLI をインストールする (Linux, macOS, or Unix)

Linux, macOS, or Unix では、バンドルされたインストーラを使用して AWS Command Line Interface (AWS CLI) をインストールすることができます。バンドルされたインストーラにはすべての依存関係が含まれており、オフラインで使用できます。

> Important
>
> バンドルされたインストーラでは、スペースを含むパスへのインストールはサポートされていません。

セクション

前提条件

- Linux, macOS, or Unix
- Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+

Python のインストールを確認します。

```
$ python --version
```

ご使用のコンピュータに Python がインストールされていない場合、または別のバージョンの Python をインストールする場合は、「Linux に AWS CLI をインストールする (p. 5)」の手順に従います。

バンドルされたインストーラを使用して AWS CLI をインストールする

コマンドラインから以下の手順に従い、バンドルされたインストーラを使用して AWS CLI をインストールします。

バンドルされたインストーラを使用して AWS CLI をインストールするには

1. AWS CLI のバンドルされたインストーラをダウンロードします。

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

2. パッケージを解凍します。

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

   Note

   `unzip` がない場合は、Linux ディストリビューションに組み込まれたパッケージマネージャーを使用してインストールします。

3. インストール実行可能ファイルを実行します。

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

   Note

   デフォルトでは、インストールスクリプトはシステムのデフォルトバージョンの Python で実行されます。別のバージョンの Python がインストールされており、それを使用して AWS CLI をインストールする場合は、Python の実行可能ファイルへの絶対パスを指定してそのバージョンでインストールスクリプトを実行します。例:

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

インストーラは AWS CLI を /usr/local/aws にインストールし、シンボリックリンク aws を /usr/local/bin ディレクトリに作成します。-b オプションを使用してシンボリックリンクを作成すると、ユーザーの $PATH 変数にインストールディレクトリを指定する必要がなくなります。これにより、すべてのユーザーが任意のディレクトリから aws を入力して AWS CLI を呼び出せるようになります。

-i オプションおよび -b オプションの説明を表示するには、-h オプションを使用します。

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

Sudo を使用せずに AWS CLI をインストールする (Linux, macOS, or Unix)

sudo を使用するアクセス許可がない場合、または現在のユーザー向けに AWS CLI のみをインストールする場合は、前述のコマンドの修正バージョンを使用できます。

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

これにより、AWS CLI がデフォルトの場所 (~/.local/lib/aws) にインストールされ、シンボリックリンクが ~/bin/aws に作成されます。シンボリックリンクが動作するように、~/bin が PATH 環境変数にあることを確認してください。

```
$ 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
>
> $PATH 設定がセッション間で維持されるように、export 行をシェルプロファイル (~/.profile、~/.bash_profile など) に追加します。

AWS CLI のアンインストール

バンドルされたインストーラでは、オプションのシンボリックリンク以外は、インストールディレクトリの外に何も置きません。そのため、アンインストールはこの 2 つの項目を削除するだけでシンプルです。

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

AWS CLI の設定

このセクションでは、お客様のセキュリティ認証情報、デフォルトの出力形式、デフォルトの AWS リージョンなど、AWS Command Line Interface (AWS CLI) が AWS とやり取りする際に使用する設定を指定する方法について説明します。

> Note
>
> AWS では、すべての着信リクエストは暗号で署名される必要があります。これは、AWS CLI によって行われます。「署名」には日時スタンプが含まれます。したがって、コンピュータの日付と時刻が正しく設定されていることを確認する必要があります。日時が正しく設定されていない場合、署名の日時が AWS サービスによって認識された日間と大きく異なることがあり、その場合は AWS によってリクエストが却下されます。

セクション

AWS CLI のかんたん設定

一般的な使用の場合、`aws configure` コマンドが、AWS CLI のインストールをセットアップするための最も簡単な方法です。

```
$ 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
```

このコマンドを入力すると、AWS CLI によって 4 種類の情報 (アクセスキー、シークレットアクセスキー、AWS リージョン、出力形式) の入力が求められ、`default` という名前の プロファイル (設定のコレクション) に保存されます。したがって、このプロファイルは、使用するプロファイルを明示的に指定しない AWS CLI コマンドを実行する場合はいつでも使用されます。

アクセスキーとシークレットアクセスキー

`AWS Access Key ID` および `AWS Secret Access Key` は、ユーザーの AWS 認証情報です。これらは、付与されるアクセス許可を決定する AWS Identity and Access Management (IAM) ユーザーまたはロールに関連付けられています。IAM サービスを使用してユーザーを作成する方法のチュートリアルに関して

は、『IAM ユーザーガイド』の「最初の IAM 管理者のユーザーおよびグループの作成」を参照してください。

IAM ユーザーのアクセスキー ID およびシークレットアクセスキーを取得するには

アクセスキーはアクセスキー ID とシークレットアクセスキーから成り、AWS に対するプログラムによるリクエストに署名するときに使用されます。アクセスキーがない場合は、AWS マネジメントコンソールから作成することができます。AWS アカウントのルートユーザー のアクセスキーの代わりに、IAM のアクセスキーを使用することをお勧めします。IAM では、AWS アカウントでの AWS サービスとリソースへのアクセスを安全に制御できます。

シークレットアクセスキーを表示またはダウンロードできるのは、このキーを作成するときのみです。アクセスキーを後で復元することはできません。ただし、新しいアクセスキーはいつでも作成できます。必要な IAM アクションを実行するためのアクセス許可も必要です。詳細については、『IAM ユーザーガイド』の「他の IAM リソースにアクセスするのに必要なアクセス権限」を参照してください。

1. IAM コンソールを開きます。
2. コンソールのナビゲーションペインで、[Users] を選択します。
3. IAM のユーザー名 (チェックボックスではありません) を選択します。
4. [Security credentials] タブを選択し、次に [Create access key] を選択します。
5. 新しいアクセスキーを表示するには、[Show] を選択します。認証情報は以下のようになります。
   - アクセスキー ID: AKIAIOSFODNN7EXAMPLE
   - シークレットアクセスキー: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. キーペアをダウンロードするには、[Download .csv file] を選択します。このキーは安全な場所に保存してください。

   AWS アカウントを保護するためにキーは機密にしておき、メールでも送信しないでください。また、所属している組織外にこの情報を公開してはいけません。AWS または Amazon.com を名乗る人物から問い合わせがあった場合でも、この情報は開示しないでください。Amazon のスタッフまたは関係者がこの情報を尋ねることは決してありません。

関連トピック

- IAM とは」 (IAM ユーザーガイド)
- AWS セキュリティ認証情報 (AWS General Reference)

リージョン

`Default region name` は、デフォルトでリクエストを送信するサーバーの AWS リージョンを特定します。通常、お客様の最寄りのリージョンですが、どのリージョンでもかまいません。たとえば、`us-west-2` を入力して 米国西部 (オレゴン) を使用できます。これは、個別のコマンドで指定されない限り、今後のすべてのリクエストが送信されるリージョンです。

> **Note**
>
> AWS CLI を使用する際は、明示的に、またはデフォルトリージョンを設定して、AWS リージョンを指定する必要があります。利用可能なリージョンのリストについては、「リージョンとエンドポイント」を参照してください。AWS CLI で使用されるリージョン識別子は、AWS マネジメントコンソール の URL およびサービスエンドポイントに表示されるのと同じ名前です。

出力形式

`Default output format` は、結果の形式を指定します。値は、次下のリストのいずれかの値を指定できます。出力形式を指定しない場合、`json` がデフォルトとして使用されます。

- **json**: JSON 文字列形式で出力されます。
- **text**: 複数行のタブ区切り文字列値の形式で出力されます。出力をテキストプロセッサ (grep、sed、awk など) に渡す場合に便利です。
- **table**: セルの罫線を形成する文字列 +|- を使用して表形式で出力されます。通常、情報は他の形式よりも読みやすい「わかりやすい」形式で表示されますが、プログラムとしては役立ちません。

クイック設定と複数のプロファイル

前に示したコマンドを使用した場合、default という名前の 1 つのプロファイルが出力されます。--profile オプションを使用してプロファイルの名前を指定し、追加設定を作成することもできます。

```
$ 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
```

これで、コマンドを実行する際、--profile オプションを省略して、default プロファイルに保存されている設定を使用することができます。

```
$ aws s3 ls
```

または、--profile *profilename* を指定して、その名前で保存されている設定を使用することもできます。

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

設定を更新するには、aws configure を再度実行し (更新するプロファイルに応じて --profile パラメータを指定するか)、必要に応じて、新しい値を入力するだけです。次のセクションでは、aws configure で作成するファイル、追加の設定、名前付きプロファイルについて説明します。

構成設定と優先順位

AWS CLI では、認証情報プロバイダーを使用して、AWS 認証情報を検索します。各認証情報プロバイダーは、システムまたはユーザーの環境変数やローカルの AWS 設定ファイルなどの場所、またはコマンドラインでパラメータとして明示的に宣言されたいくつかの場所で認証情報を検索します。AWS CLI では、次の順序でプロバイダーを呼び出して、認証情報と構成設定を検索し、使用する認証情報セットが見つかると停止します。

1. コマンドラインオプション (p. 25) – コマンドラインのパラメータとして --region、--output、および --profile を指定できます。
2. 環境変数 (p. 24) – 環境変数 (AWS_ACCESS_KEY_ID、AWS_SECRET_ACCESS_KEY、および AWS_SESSION_TOKEN) に値を保存できます。存在する場合は、これらの環境変数が使用されます。
3. CLI 認証ファイル (p. 22) – これは、コマンド aws configure を実行すると更新されるファイルです。ファイルは、~/.aws/credentials (Linux, macOS, or Unix の場合) または C:\Users*USERNAME*\.aws\credentials (Windows の場合) にあります。このファイルには、default プロファイルと任意の名前付きプロファイルの認証情報の詳細が含まれています。
4. CLI 設定ファイル (p. 22) – aws configure コマンドを実行すると更新されるもう 1 つのファイルです。ファイルは、~/.aws/config (Linux, macOS, or Unix の場合) または C:\Users*USERNAME*\.aws\config (Windows の場合) にあります。このファイルには、 デフォルトプロファイルとあらゆる名前付きプロファイルの設定が含まれています。

5. コンテナ認証情報 – IAM ロールと各 Amazon Elastic Container Service (Amazon ECS) タスク定義を関連付けることができます。関連付けられると、そのロールの一時認証情報は、そのタスクのコンテナで使用できるようになります。詳細については、『Amazon Elastic Container Service Developer Guide』の「タスク用の IAM ロール」を参照してください。
6. インスタンスプロファイル認証情報 – IAM ロールと各 Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを関連付けることができます。関連付けられると、そのロールの一時認証情報は、インスタンスで実行中のコードで使用できるようになります。認証情報は、Amazon EC2 メタデータサービスから提供されます。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EC2 の IAM ロール」および『IAM ユーザーガイド』の「インスタンスプロファイルの使用」を参照してください。

設定ファイルと認証情報ファイル

AWS CLI は `aws configure` で指定された認証情報を、ホームディレクトリの `.aws` という名前のフォルダにある `credentials` という名前のローカルファイルに保存します。`aws configure` で指定されたもう 1 つの設定オプションは、`config` という名前のローカルファイルに保存されるだけでなく、ホームディレクトリの `.aws` フォルダにも保存されます。

ホームディレクトリの場所はオペレーティングシステムによって異なりますが、環境変数 `%UserProfile%` (Windows の場合) および `$HOME` またはチルド ~ (Unix ベースのシステムの場合) を使用して参照されます。

たとえば、次のコマンドは `.aws` フォルダの内容を一覧表示します。

Linux, macOS, or Unix

```
$ ls  ~/.aws
```

Windows

```
C:\> dir "%UserProfile%\.aws"
```

AWS CLI では、2 つのファイルを使用して、機密性の高い認証情報 (`~/.aws/credentials` 内) と機密性の低い設定オプション (`~/.aws/config` 内) を区別しています。

`AWS_CONFIG_FILE` 環境変数を別のローカルパスに設定することで、`config` ファイルの場所をデフォルト以外の場所に指定できます。詳細については、「Environment Variables (p. 24)」を参照してください。

認証情報を設定ファイルに保存する

また、AWS CLI は `config` ファイルから認証情報を読み取ることができます。すべてのプロファイル設定を 1 つのファイルに保管することもできます。両方の場所にプロファイルの認証情報がある場合 (たとえば、プロファイルのキーを更新するために `aws configure` を使用した場合など)、認証情報ファイルのキーが優先されます。
AWS CLI に加えていずれかの SDK を使用するとき、認証情報が独自のファイルに保存されていない場合は追加の警告が表示される場合があります。

前のセクションで設定されたプロファイルのために CLI で生成されたファイルは次のようになります。

`~/.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
>
> 前述の例では、1 つのデフォルトプロファイルを含むファイルを示しています。複数の名前付きプロファイルを含むファイルの例については、「名前付きプロファイル (p. 23)」を参照してください。

以下の設定がサポートされています。

aws_access_key_id – AWS アクセスキー。

aws_secret_access_key – AWS シークレットキー。

aws_session_token – AWS セッショントークン。セッショントークンは、一時的なセキュリティ認証情報を使用している場合にのみ必要です。

region – このプロファイルからリクエストを送信するデフォルトの AWS リージョン。

output (p. 20) – このプロファイルのデフォルトの出力形式。

名前付きプロファイル

AWS CLI では、`config` ファイルおよび `credentials` ファイルに保存された複数の名前付きプロファイルのいずれかを使用することができます。追加のプロファイルを設定するには、`--profile` オプションで `aws configure` を使用するか、`config` ファイルと `credentials` ファイルにエントリを追加します。

2 つのプロファイルのある `credentials` ファイルの例を以下に示します。1 つ目は CLI コマンドをプロファイルなしで実行するときに使用します。2 つ目は CLI コマンドに `--profile user1` パラメータを指定して実行するときに使用します。

~/.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
```

各プロファイルは異なる認証情報 (異なる IAM ユーザーのもの) を使用します。別の AWS リージョンおよび出力形式を使用することもできます。

~/.aws/config

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

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

```
output=text
```

Important

`credentials` ファイルでは、CLI `config` ファイルの名前付きプロファイルとは別の命名形式を使用します。`config` ファイルで名前付きプロファイルを設定する場合のみ、プレフィックス "`profile`" を含めてください。`credentials` ファイルを設定するときは、`profile` を使用しないでください。

AWS CLI でのプロファイルの使用

名前付きプロファイルを使用するには、コマンドに `--profile` *`profile-name`* オプションを追加します。次の例では、前のファイル例の `user1` プロファイルを使用して、すべての Amazon EC2 インスタンスを一覧表示します。

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

複数のコマンドで名前付きプロファイルを使用するには、コマンドラインに `AWS_PROFILE` 環境変数を設定することで、すべてのコマンドでプロファイルを指定せずにすみます。

Linux, macOS, or Unix

```
$ export AWS_PROFILE=user2
```

Windows

```
C:\> set AWS_PROFILE=user2
```

環境変数を設定すると、シェルセッションの終了時まで、または変数に別の値を設定するまで、デフォルトのプロファイルが変更されます。詳細については、「Environment Variables (p. 24)」を参照してください。

Environment Variables

環境変数を使用すると、別の方法で設定オプションと認証情報を指定できます。このため、スクリプト処理や、名前付きプロファイルを一時的にデフォルトとして設定する場合に便利です。

オプションの優先順位

- このトピックで示されている環境変数のいずれかを使用してオプションを指定した場合、設定ファイルのプロファイルからロードされた値は上書きされます。
- CLI コマンドラインでパラメータを使用してオプションを指定した場合、対応する環境変数か、設定ファイルのプロファイルからロードされた値は上書きされます。

サポートされている環境変数

AWS CLI は次の環境変数をサポートしています。

- `AWS_ACCESS_KEY_ID` – IAM ユーザーまたはロールに関連付けられる AWS アクセスキーを指定します。
- `AWS_SECRET_ACCESS_KEY` – アクセスキーに関連付けられるシークレットキーを指定します。これは、基本的にアクセスキーの「パスワード」です。

- `AWS_SESSION_TOKEN` – 一時的なセキュリティ認証情報を使用している場合にのみ必要となるセッショントークン値を指定します。詳細については、『AWS CLI Command Reference』の「★Output section of the assume-role command」を参照してください。
- `AWS_DEFAULT_REGION` – リクエストを送信する AWS リージョン (p. 20)を指定します。
- `AWS_DEFAULT_OUTPUT` – 使用する出力形式を指定します。
- `AWS_PROFILE` – CLI プロファイル (p. 23)の名前と、使用する認証情報およびオプションを指定します。これは、`credentials` ファイルまたは `config` ファイルに保存されているプロファイルの名前、または、デフォルトプロファイルを使用する場合は値 `default` となります。この環境変数を指定した場合、設定ファイルの `[default]` という名前のプロファイルを使用する動作は上書きされます。
- `AWS_CA_BUNDLE` – HTTPS 証明書の検証に使用する、証明書バンドルへのパスを指定します。
- `AWS_SHARED_CREDENTIALS_FILE` – AWS CLI で使用してアクセスキーを保存するファイルの場所を変更します (デフォルトは `~/.aws/credentials`)。
- `AWS_CONFIG_FILE` – AWS CLI で使用して設定プロファイルを保存するファイルの場所を変更します (デフォルトは `~/.aws/config`)。

次の例では、デフォルトのユーザーの環境変数を設定する方法を示します。これらの値は、名前付きプロファイルにある値またはインスタンスメタデータより優先されます。設定後にこれらの値を上書きするには、CLI コマンドラインでパラメータを指定するか、環境変数を変更または削除します。

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

```
C:\> set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
C:\> set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> set AWS_DEFAULT_REGION=us-west-2
```

コマンドラインオプション

次のコマンドラインオプションを使用して、1 つのコマンドのデフォルトの設定を上書きできます。使用するプロファイルを指定することはできますが、コマンドラインオプションを使用して認証情報を直接指定することはできません。

--profile

このコマンドに使用する named profile (p. 23) を指定します。追加の名前付きプロファイルを設定するには、`--profile` オプションを指定して `aws configure` コマンドを使用します。

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

--region

このコマンドの AWS リクエストを送信する AWS リージョンを指定します。指定できるすべてのリージョンのリストについては、『アマゾン ウェブ サービス全般のリファレンス』の「AWS リージョンとエンドポイント」を参照してください。

--出力

このコマンドに使用する出力形式を指定します。次の値のいずれかを指定できます。

- **`json`**: JSON 文字列形式で出力されます。

- **text**: 複数行のタブ区切り文字列値の形式で出力されます。出力をテキストプロセッサ (grep、sed、awk など) に渡す場合に便利です。
- **table**: セルの罫線を形成する文字列 +|- を使用して表形式で出力されます。通常、情報は他の形式よりも読みやすい「わかりやすい」形式で表示されますが、プログラムとしては役立ちません。

--endpoint-url

リクエストを送信する URL。ほとんどのコマンドでは、AWS CLI により、選択したサービスと AWS リージョンに基づいて URL が自動的に決定されます。ただし、一部のコマンドでは、アカウント固有の URL を指定する必要があります。一部の AWS サービスでは、プライベート VPC 内で直接エンドポイントをホストすることもできますが、URL を指定する必要がある場合があります。

各リージョンで使用できる標準的なサービスエンドポイントについては、『アマゾン ウェブ サービス全般のリファレンス』の「AWS リージョンとエンドポイント」を参照してください。

これらのオプションを 1 つ以上コマンドラインパラメータとして指定すると、デフォルト設定またはその 1 つのコマンドに対応するいずれかのプロファイル設定が上書きされます。それぞれのオプションの文字列引数にはスペースまたは等号 (=) が付いていて、オプション名から引数を分離しています。引数値にスペースが含まれている場合は、引数を引用符で囲む必要があります。

コマンドラインオプションの一般的な使用方法には、複数の AWS リージョンでのリソースの確認、および、スクリプティングでの読みやすさや使いやすさのための出力形式の変更が含まれます。たとえば、インスタンスが実行されているリージョンが不明な場合は、以下に示すように、わかるまで各リージョンに対して describe-instances コマンドを実行できます。

```
$ 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                                        |||
...
```

各コマンドラインオプションの引数の型 (文字列、ブールなど) については、「パラメータ値の指定 (p. 43)」で詳しく説明しています。

インスタンスメタデータ

Amazon EC2 インスタンス内から AWS CLI を実行すると、コマンドへの認証情報の提供を簡素化できます。各 Amazon EC2 インスタンスにはメタデータが含まれており、AWS CLI は一時的な認証情報に対してクエリを直接実行できます。これらの機能を使用するには、必要なリソースへのアクセス権を持つ AWS Identity and Access Management (IAM) ロールを作成し、そのロールを Amazon EC2 インスタンスにその起動時に割り当てます。

インスタンスを起動し、AWS CLI がすでにインストールされているかどうかを確認します (Amazon Linux ではプリインストールされています)。必要に応じて、AWS CLI をインストールします。すべてのコマンドで指定しなくてもいいようにデフォルトのリージョンを指定する必要があります。

最初の 2 つのプロンプトを Enter キーを 2 回押してスキップすることで、認証情報を指定せずに `aws configure` を実行してリージョンとデフォルトの出力形式を設定できます。

```
$ 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 ロールがインスタンスにアタッチされている場合、AWS CLI はインスタンスのメタデータから認証情報を自動的かつ安全に取得します。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに、AWS リソースへのアクセスを付与する」を参照してください。

HTTP プロキシを使用する

プロキシサーバーを使用して AWS にアクセスするには、`HTTP_PROXY` および `HTTPS_PROXY` 環境変数を、プロキシサーバーが使用する IP アドレスとポート番号を使用して設定することができます。

Linux, macOS, or Unix

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

Windows

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

これらの例では、`http://a.b.c.d:n` と `http://w.x.y.z:m` は HTTP および HTTPS のプロキシの IP アドレスおよびポート番号です。

プロキシを認証する

AWS CLI は HTTP Basic 認証をサポートしています。次のように、プロキシ URL にユーザー名とパスワードを指定します。

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

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

> **Note**
>
> AWS CLI では NTLM プロキシはサポートされていません。NTLM または Kerberos プロトコルプロキシを使用する場合は、Cntlm などの認証プロキシを介して接続できることがあります。

Amazon EC2 インスタンスでのプロキシの使用

アタッチされた IAM ロールで起動した Amazon EC2 インスタンスにプロキシを設定する場合は、インスタンスメタデータへのアクセスに使用されたアドレスを除外するようにします。そのためには、`NO_PROXY` 環境変数をインスタンスメタデータサービスの IP アドレス 169.254.169.254 に設定します。

Linux, macOS, or Unix

```
$ export NO_PROXY=169.254.169.254
```

Windows

```
C:\> set NO_PROXY=169.254.169.254
```

AWS CLI で IAM ロールを引き受ける

AWS Identity and Access Management (IAM) ロールは、追加の (または異なる) アクセス許可や、別の AWS アカウントでアクションを実行するためのアクセス許可を IAM ユーザーが取得できるようにする認証ツールです。

IAM ロールを使用するように AWS Command Line Interface (AWS CLI) を設定するには、`~/.aws/config` ファイルでロールのプロファイルを定義します。

次の例では、`marketingadmin` プロファイルを指定するコマンドの実行時に引き受ける `marketingadmin` という名前のロールプロファイルを示しています。

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

このロールは、IAM ユーザー認証情報に加えてロールを引き受けるためのアクセス許可を含む別個の名前付きプロファイルにリンクする必要があります。前の例の `marketingadmin` プロファイルは `source-profile` フィールドを使用して `user1` プロファイルにリンクされます。AWS CLI コマンドでのプロファイル `marketingadmin` の使用を指定すると、CLI はリンクされた `user1` プロファイルの認証情報を自動的に検索し、その認証情報を使用して、指定された IAM ロールの一時的な認証情報を要求します。これらの一時的な認証情報は、次に CLI コマンドを実行するために使用されます。指定されたロールには、CLI コマンドを実行できる IAM アクセス許可ポリシーがアタッチされている必要があります。

セクション

ロールの設定と使用

IAM ロールを指定するプロファイルを使用してコマンドを実行すると、AWS CLI はソースプロファイルの認証情報を使用して AWS Security Token Service (AWS STS) を呼び出し、指定したロールの一時的な認証情報を要求します。ソースプロファイルのユーザーは、指定されたプロファイルのロール用の `sts:assume-role` を呼び出すアクセス許可を持っている必要があります。ロールにはソースプロファイ

ルのユーザーがこのロールを引き受けることができる信頼関係が必要です。ロールの一時的な認証情報を取得して使用するプロセスを、一般にロールを引き受けると呼びます。

『AWS Identity and Access Management ユーザーガイド』の「IAM ユーザーにアクセス許可を委任するロールの作成」の手順に従って、ユーザーが引き受けるアクセス許可を使用して、IAM で新しいロールを作成できます。ロールとソースプロファイルの IAM ユーザーが同じアカウントに存在する場合、ロールの信頼関係を設定するときに、独自のアカウント ID を入力することができます。

ロールを作成した後、IAM ユーザー (または AWS アカウントのユーザー) が引き受けることを許可するように信頼関係を変更します。

次の例は、アカウント 123456789012 の任意の IAM ユーザーによるロールの引き受けを許可する信頼関係を示しています。ただし、これはそのアカウントの管理者が明示的に `sts:assumerole` アクセス許可をそのユーザーに付与した場合です。

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

信頼ポリシーは、実際にはアクセス許可を付与しません。アカウントの管理者は、適切なアクセス許可を持つポリシーをアタッチすることによって、ロールを引き受けるアクセス許可を個々のユーザーに委任する必要があります。次の例で、アタッチされた IAM ユーザーは `marketingadmin` ロールのみを引き受けることができます。

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

IAM ユーザーには、ロールプロファイルを使用して CLI コマンドを実行するための追加のアクセス許可は必要ありません。代わりに、コマンドの実行に必要なアクセス許可としてそのロールにアタッチされたアクセス許可が使用されます。しかし、ロールを使用せずに AWS リソースにユーザーがアクセスできるようにするには、追加のインラインポリシーまたは管理ポリシーを IAM ユーザーにアタッチしてそのリソースに対するアクセス許可を付与する必要があります。

ロールプロファイル、ロールのアクセス許可、ロールの信頼関係およびユーザーアクセス許可が適切に設定されたので、コマンドラインで `--profile` オプションを呼び出してロールを使用できます。たとえば、次のコマンドは、このトピックの冒頭の例で定義された `marketingadmin` ロールにアタッチされたアクセス許可を使用して Amazon S3 `ls` コマンドを呼び出します。

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

いくつかの呼び出しにロールを使用するには、コマンドラインから、現在のセッションに対して `AWS_PROFILE` 環境変数を設定することができます。この環境変数が定義されている場合、各コマンドで `--profile` オプションを指定する必要はありません。

Linux, macOS, or Unix

```
$ export AWS_PROFILE=marketingadmin
```

Windows

```
C:\> set AWS_PROFILE=marketingadmin
```

IAM ユーザーおよびロールの設定の詳細については、『IAM ユーザーガイド』の「ID (ユーザー、グループ、ロール)」および「IAM ロール」を参照してください。

多要素認証を使用する

セキュリティを高めるには、ロールプロファイルを使用して呼び出しを試みるときに、多要素認証 (MFA) デバイスから生成された一回限りのキー、U2F デバイス、またはモバイルアプリケーションを指定するようにユーザーに要求することができます。

最初に、IAM ロールで信頼関係を変更して MFA を要求します。例として、次の例の `Condition` 行を参照してください。

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

次に、ユーザーの MFA デバイスの ARN を指定する、ロールプロファイルに行を追加します。

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

この `mfa_serial` 設定では、次に示すように、ARN またはハードウェア MFA トークンのシリアル番号を使用できます。

クロスアカウントロール

クロスアカウントロールとしてロールを設定することにより、IAM ユーザーが別のアカウントに属しているロールを引き受けることができるようにします。「IAM ユーザーにアクセス許可を委任するロールの作成」の説明に従って、ロールの作成中にロールタイプを [別の AWS アカウント] に設定します。必要に応じて、[MFA が必要] を選択します。[MFA が必要] オプションは、「多要素認証を使用する (p. 30)」で説明されているように、信頼関係の適切な条件を設定します。

外部 ID を使用して、アカウント間で誰がロールを引き受けるかをさらに制御する場合、ロールプロファイルに `external_id` パラメータを追加する必要もあります。これは通常、もう一方のアカウントが社外または組織外のユーザーによって制御される場合にのみ使用します。

```
[profile crossaccountrole]
```

```
role_arn = arn:aws:iam::234567890123:role/SomeRole
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/saanvi
external_id = 123456
```

キャッシュされた認証情報のクリア

AWS CLI は、ロールを引き受けると、有効期限が切れるまで一時認証情報をキャッシュします。ロールの一時認証情報が取り消された場合、キャッシュを削除して、新しい認証情報の取得を AWS CLI に強制できます。

Linux, macOS, or Unix

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

Windows

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

コマンド補完

Unix に似たシステムでは、AWS CLI に含まれるコマンド補完機能によって、Tab キーを使用して部分的に入力されたコマンドを補完します。ほとんどのシステムでは、この機能は自動的にインストールされないため、手動で設定する必要があります。

コマンド補完を設定するには、2 つの情報として、使用しているシェルの名前と `aws_completer` スクリプトの場所が必要です。

Amazon Linux

コマンド補完は自動的に設定され、Amazon Linux を実行する Amazon EC2 インスタンス上でデフォルトで有効化されます。

セクション

シェルを識別する

使用しているシェルが不明な場合は、次のいずれかのコマンドを使用して識別できます。

echo $SHELL – シェルのプログラムファイル名が表示されます。これは通常、ログイン後に別のシェルを起動しない限り、使用中のシェルの名前と一致します。

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

ps – 現在のユーザーで実行中のプロセスが表示されます。シェルはそのうちの 1 つです。

```
$ ps
```

```
  PID TTY          TIME CMD
 2148 pts/1    00:00:00 bash
 8756 pts/1    00:00:00 ps
```

AWS コンプリータを見つける

AWS コンプリータの場所は、使用するインストール方法によって異なります。

パッケージマネージャー – pip、yum、brew、apt-get などのプログラムでは通常、AWS コンプリータ (またはシンボリックリンク) は標準のパスの場所にインストールされます。この場合、which コマンドを実行して、コンプリータを見つけることができます。

--user コマンドを使用せずに pip を使用した場合は、次のパスが表示される場合があります。

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

pip install コマンドで --user パラメータを使用した場合、コンプリータは通常、$HOME フォルダの下の local/bin フォルダにあります。

```
$ which aws_completer
/home/username/.local/bin/aws_completer
```

バンドルされたインストーラ – 前のセクションの手順に従ってバンドルされたインストーラを使用した場合、AWS コンプリータはインストールディレクトリの bin サブフォルダに配置されます。

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

他のすべてが失敗した場合、find を使用して、AWS コンプリータのファイルシステム全体を検索します。

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

コマンド補完を有効にする

コマンドを実行して、コマンドの完了を有効にします。補完を有効にするために使用するコマンドは、使用しているシェルに依存します。コマンドをシェルの RC ファイルに追加して、新しいシェルを開くたびに実行できます。各コマンドで、/usr/local/bin パスを、以前のセクションのシステム上にあるパスと置き換えます。

- **bash** – 組み込みのコマンド complete を使用します。

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

  コマンドを ~/.bashrc に追加して、新しいシェルを開くたびに実行します。~/.bash_profile はソースとして ~/.bashrc を使用して、コマンドがログインシェルでも実行されるようにできます。

- **tcsh** – tcsh の補完は、補完の振る舞いを定義するためのワードタイプとパターンを取ります。

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

  コマンドを ~/.tschrc に追加して、新しいシェルを開くたびに実行します。
- **zsh** – source bin/aws_zsh_completer.sh。

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

  AWS CLI は、zsh サポートのために bash 互換性自動補完 (bashcompinit) を使用します。詳細については、aws_zsh_completer.sh の上部を参照してください。

  コマンドを ~/.zshrc に追加して、新しいシェルを開くたびに実行します。

コマンド補完のテスト

コマンド補完を有効にしたら、コマンドの一部を入力し、Tab を押して使用可能なコマンドを表示します。

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

チュートリアル: AWS CLI を使用して Amazon EC2 開発環境をデプロイする

このチュートリアルでは、AWS Command Line Interface (AWS CLI) を使用して Amazon Elastic Compute Cloud (Amazon EC2) で開発環境をセットアップする方法について説明します。インストールおよび設定手順のショートバージョンが含まれています。Windowsや Linux, macOS, or Unix で最初から最後まで実行できます。

ステップ

ステップ 1: AWS CLI をインストールする

AWS CLI のインストールには、インストーラ (Windows) または Python のパッケージマネージャーである `pip` を使用できます。

Windows

1. MSI インストーラをダウンロードします。
   - Windows (64 ビット) 用 の AWS CLI MSI インストーラのダウンロード
   - Windows (32 ビット) 用 の AWS CLI MSI インストーラのダウンロード
2. ダウンロードした MSI インストーラを実行します。
3. 表示される手順に従います。

Linux, macOS, or Unix

これらのステップには、Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+ がインストールされ動作している必要があります。以下の手順を使用して問題が発生した場合は、AWS Command Line Interface ユーザーガイド の完全インストールの手順を参照してください。

1. `pip` をインストール済みの場合は、ステップ 2 へスキップしてください。`pip` をまだインストールしていない場合は、pip のウェブサイトからインストールスクリプトをダウンロードして実行します。

```
$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1622k  100 1622k    0     0  14.9M      0 --:--:-- --:--:-- --:--:-- 14.9M
$ python3 get-pip.py --user
```

```
Collecting pip
  Using cached https://files.pythonhosted.org/packages/c2/
d7/90f34cb0d83a6c5631cf71dfe64cc1054598c843a92b400e55675cc2ac37/pip-18.1-py2.py3-none-
any.whl
Collecting setuptools
  Using cached https://files.pythonhosted.org/packages/e7/16/
da8cb8046149d50940c6110310983abb359bbb8cbc3539e6bef95c29428a/setuptools-40.6.2-py2.py3-
none-any.whl
Collecting wheel
  Using cached https://files.pythonhosted.org/packages/
ff/47/1dfa4795e24fd6f93d5d58602dd716c3f101cfd5a77cd9acbe519b44a0a9/wheel-0.32.3-
py2.py3-none-any.whl
Installing collected packages: pip, setuptools, wheel
  The script wheel is installed in '/home/myusername/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning,
 use --no-warn-script-location.
```

`--user` スイッチは、ユーザーのローカルホームディレクトリに `pip` をインストールすることを指定します。これは、root/administrator アクセス許可がない場合に必要です。`--user` を指定しなかった場合、`pip` はすべてのユーザーについてシステムフォルダへのインストールを試みます。

2. `pip` インストールフォルダが `PATH` 環境変数にあることを確認してください。ない場合は通常、前の例に示されているように、インストーラにその旨が表示されます。これに対処するには、次のようなステートメントをシェルの RC スクリプトの末尾に追加します。これは、`pip` がホームディレクトリの `.local/bin` フォルダにインストールされている場合に適切です。

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

3. これで、`pip` を使用して AWS CLI をインストールできます。

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

繰り返しになりますが、`--user` スイッチを使用して、root/administrator アクセス許可が不要なローカルホームフォルダに AWS CLI をインストールすることをお勧めします。

ステップ 2: AWS CLI の設定

まず、認証情報とデフォルト設定で AWS CLI を設定します。

```
$ 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 CLI は、以下の情報を要求します。

- AWS アクセスキー ID と AWS シークレットアクセスキー – これらは、ユーザーまたはアカウントの認証情報です。キーを持っていない場合は、アマゾン ウェブ サービス全般のリファレンス の アクセスキー (アクセスキー ID とシークレットアクセスキー) を参照してください。
- デフォルトのリージョン名 – CLI がデフォルトでリクエストを送信する AWS リージョンの名前です。
- デフォルトの出力形式 – これは、CLI がデフォルトで使用する出力形式を指定します。値には、`json`、`text`、`table` などがあります。出力形式を指定しない場合、`json` が使用されます。

では、単純なコマンドを試して、認証情報が正しく設定されていることと、AWS に接続できることを確認します。

```
$ 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        ||
|+-----------------------------------+-------------------+|
```

ステップ 3: Amazon EC2 インスタンスのセキュリティグループおよびキーペアを作成する

このステップでは、Secure Shell (SSH) プロトコルを使用して接続された端末エミュレータでアクセスできる Amazon EC2 インスタンスを起動するための前提条件をセットアップします。Amazon EC2 と機能の詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイド を参照してください。

Amazon EC2 が EC2 インスタンスと通信するためには、以下の前提条件が必要です。

- セキュリティグループ – インスタンスへの進入と退出が許されるネットワークトラフィックを決定します。セキュリティグループは、そのグループにアタッチされているインスタンスの仮想ファイアウォールとして考えることができます。セキュリティグループには、インバウンドおよびアウトバウンドネットワークトラフィックを制御するルールが含まれます。
- キーペア – パブリックキー暗号は、パブリックキーを使用してパスワードなどのデータを暗号化し、受信者はプライベートキーを使用してデータを復号化します。Amazon EC2 は指定されたキーペアを使用して、インスタンスへのアクセスに使用される認証情報を暗号化します。

セキュリティグループとキーペアを作成するには

1. インスタンスを起動する VPC のセキュリティグループを作成します。アカウントのデフォルトの VPC を使用している場合は、`--vpc-id` パラメータを省略できます。その他の場合は、インスタンスを起動する VPC の ID を指定します。出力には、新しいセキュリティグループの識別子が示されます。

```
$ aws ec2 create-security-group --group-name devenv-sg --vpc-id vpc-xxxxxxxx --
description "Security group for development environment"
{
    "GroupId": "sg-b018ced5"
}
```

2. インスタンスへの接続に使用する CIDR ネットワークアドレス範囲からポート 22 へのインバウンドネットワークトラフィックを可能にするセキュリティグループのルールを作成します。

   Important

   この例は、インターネットのあらゆる場所からのインバウンドトラフィックを有効にする `0.0.0.0/0` CIDR 範囲を示しています。これをインスタンスに接続する (AWS によって見られる) ネットワークの public CIDR 範囲に置き換えることを強くお勧めします。

```
$ aws ec2 authorize-security-group-ingress --group-name devenv-sg --protocol tcp --port
 22 --cidr 0.0.0.0/0
```

   セキュリティグループ ID を書き留めます。後でインスタンスを起動するときに必要になります。

3. インスタンスへの接続に使用する SSH 暗号キーペアを作成します。この例のコマンドは、キーの内容を `devenv-key.pem` という名前のファイルに保存する方法を示しています。

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

   `--query "KeyMaterial"` パラメータは、必要な出力部分だけを `.pem` ファイルに抽出します。

   `--query` パラメータの二重引用符

   このトピックの例のうち、`--query` パラメータを含んでいるものはすべて、二重引用符を使用しています。Linux では、`--query` パラメータに単一引用符を使用しますが、Windows コマンドプロンプトでは、単一引用符ではなく、二重引用符を使用する必要があります。

4. Linux では、新しいキーファイルのアクセスを変更して、ユーザーだけがアクセスできるようにしてください。

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

ステップ 4: インスタンスを起動し接続する

これで、インスタンスを起動して接続する準備ができました。

インスタンスを起動し接続するには

1. 次のコマンドを、前のステップで作成したセキュリティグループの ID を使用して実行します。`--image-id` パラメータでは、Amazon EC2 がインスタンスをブートストラップするために使用する Amazon マシンイメージ (AMI) を指定します。AWS リージョンおよびオペレーティングシステムのイメージ ID は、Amazon EC2 コンソールを使用して確認できます。デフォルト VPC のデフォルトのサブネットを使用している場合は、`--subnet-id` パラメータを省略できます。その他の場合は、インスタンスを起動するサブネットの ID を指定します。

   Note

   この例は、Linux の「\」行連結文字を使用して複数行に分割されたコマンドを示しています。もちろん、すべてを 1 行で送信することもできます。Windows コマンドラインでは、「\」を「^」に置き換えます。

```
$ 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. インスタンスが起動するまで、少し時間がかかります。インスタンスの起動後、インスタンスへの接続に必要なインスタンスのパブリック IP アドレスは、以下のコマンドを使用して取得できます。

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

3. インスタンスに接続するには、パブリック IP アドレスとプライベートキーの `.pem` ファイルを任意のターミナルプログラムで使用します。Linux, macOS, or Unix では、コマンドラインから次のコマンドを使用して実行できます。

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

   インスタンスへの接続で、「Permission denied(publickey)」などのエラーが発生した場合は、以下が正しいことを確認します。

   - キー – 指定されたキーが指定されたパスにあり、プライベートキーである (パブリックキーではない)。キーのアクセス権限が所有者のみに制限されている。
   - ユーザー – ユーザー名がインスタンスの起動に使用された AMI に関連付けられているデフォルトのユーザー名と一致している。Ubuntu AMI の場合は `ubuntu` です。Amazon Linux AMI の場合は、`ec2-user` です。
   - インスタンス – インスタンスのパブリック IP アドレスまたは DNS 名。アドレスがパブリックであること、およびポート 22 がインスタンスのセキュリティグループのローカルマシンに対して開かれていることを確認してください。

   `-v` オプションを使用してエラーに関連する追加情報を表示することもできます。

   Windows の SSH

   Windows では、ここで入手できる PuTTY ターミナルアプリケーションを使用できます。ダウンロードページから `putty.exe` および `puttygen.exe` を入手します。
   `puttygen.exe` を使用して、プライベートキーを PuTTY で必要な `.ppk` ファイルに変換します。`putty.exe` を起動し、インスタンスのパブリック IP アドレスを [Host Name] フィールドに入力して、接続タイプを SSH に設定します。
   [Category (カテゴリ)] パネルで、[Connection (接続)]、[SSH]、[Auth] の順に移動し、[Browse] をクリックして `.ppk` ファイルを選択します。次に、[Open] をクリックして接続します。

4. ターミナルはサーバーのパブリックキーを受け入れるように要求します。`yes` と入力して、Enter キーを押し、接続を完了します。

これで、セキュリティグループの構成、キーペアの作成、EC2 インスタンスの起動、およびインスタンスへの接続が、コマンドラインを離れることなく完了しました。

AWS CLI の使用

このセクションでは、AWS Command Line Interface (AWS CLI) で使用できる多くの一般的な機能とオプションを紹介します。

> Note
>
> デフォルトでは、AWS CLI は TCP ポート 443 で HTTPS を使用することによって、リクエストを AWS に送信します。AWS CLI を正常に使用するには、TCP ポート 443 でのアウトバウンド接続が可能である必要があります。

トピック

AWS CLI のヘルプ

AWS Command Line Interface (AWS CLI) を使用している場合は、どのコマンドのヘルプも表示できます。そのためには、コマンド名の末尾に `help` と入力するだけです。

たとえば、次のコマンドは、一般的な AWS CLI オプションと使用可能な最上位レベルのコマンドに関するヘルプを表示します。

```
$ aws help
```

次のコマンドでは、使用可能な Amazon Elastic Compute Cloud (Amazon EC2) 固有のコマンドが表示されます。

```
$ aws ec2 help
```

次の例は、Amazon EC2 `DescribeInstances` 操作に関する詳細なヘルプを表示します。ヘルプには、入力パラメータ、使用可能なフィルター、および出力に含まれるものについての説明があります。コマンドの一般的なバリエーションを入力する方法を示す例も含まれています。

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

各コマンドのヘルプは 6 つのセクションに分かれています。

名前

コマンドの名前。

```
NAME
       describe-instances -
```

説明

コマンドが呼び出す API 操作の説明。

```
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.
...
```

概要

コマンドとそのオプションを使用するための基本的な構文。オプションが角括弧で示されている場合は、そのオプションが任意である、デフォルト値がある、または代わりに使用できる代替オプションがあることを意味しています。

```
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]
```

たとえば、describe-instances のデフォルトの動作では、現在のアカウントおよび AWS リージョン内のすべてのインスタンスを記述します。必要に応じて instance-ids のリストを指定して、1 つ以上のインスタンスを定義することもできます。dry-run は値を取らないオプションのブールフラグです。ブールフラグを使用するには、表示される値のいずれかを指定します。この場合は --dry-run または --no-dry-run です。同様に、--generate-cli-skeleton も値を取りません。オプションの使用に条件がある場合は、OPTIONS セクションで説明されるか、例に示されます。

オプション

Synopsis に示される各オプションの説明。

```
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.
...
```

例

コマンドとそのオプションの使用方法を示す例。必要なコマンドまたはユースケースについて例がない場合は、このページまたはコマンドのヘルプページの AWS CLI コマンドリファレンスにあるフィードバックリンクを使用してリクエストしてください。

```
    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 an Owner tag

    Command:

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

出力

AWS からの応答に含まれる各フィールドとデータタイプの説明。

describe-instances の場合は、出力は予約オブジェクトのリストであり、それぞれのオブジェクトに、関連付けられたインスタンスに関する情報を含む複数のフィールドとオブジェクトがあります。この情報は、Amazon EC2 で使用される予約データ型に関する API ドキュメントからの抜粋です。

```
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.
...
```

出力が AWS CLI によって JSON にレンダリングされるときには、次の例と同様の予約オブジェクトの配列になります。

```
{
    "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"
                    },
...
```

各予約オブジェクトには、予約およびインスタンスオブジェクトの配列を説明するフィールドがあり、それぞれにそれを説明する独自のフィールド (例: `PublicDnsName`) とオブジェクト (例: `State`) があります。

Windows ユーザー

ヘルプコマンドの出力を `more` コマンドにパイプ (|) して、ヘルプファイルを 1 ページずつ表示することができます。スペースバーまたは PgDn を押すと、ドキュメントの続きが表示され、**q** を押すと終了します。

```
C:\> aws ec2 describe-instances help | more
```

AWS CLI ドキュメント

AWS CLI Command Reference には、すべての AWS CLI コマンドのヘルプコンテンツも含まれています。説明は、モバイル、タブレット、またはデスクトップ画面で移動や表示がしやすいように表示されます。

Note

ヘルプファイルには、コマンドラインからは表示や移動ができないリンクが含まれています。このようなリンクは、オンライン AWS CLI Command Reference リファレンスを使用して表示し、操作することができます。

API ドキュメント

AWS CLI のすべてのコマンドは、AWS サービスのパブリック API に対して行われるリクエストに対応しています。パブリック API を使用する各サービスには、API リファレンスがあり、AWSドキュメントのウェブサイト のサービスのホームページで検索できます。API リファレンスの内容は、API の構築方法および使用されているプロトコルによって異なります。通常、API リファレンスには、API によってサポートされるアクション、サービスとの間で送受信されるデータ、およびサービスが報告するエラー条件に関する詳細情報が含まれています。

API ドキュメントセクション

- アクション – 各アクションとパラメータに関する詳細情報 (長さまたは内容に関する制約とデフォルト値を含む)。このアクションで発生する可能性のあるエラーが一覧表示されます。各アクションは、AWS CLI のサブコマンドに対応します。
- データタイプ – コマンドがパラメータとして要求するか、リクエストに対する応答で返す構造体に関する詳細情報。
- 一般的なパラメータ – サービスのすべてのアクションに共通するパラメータに関する詳細情報。
- 一般的なエラー – サービスのアクションによって返される可能性のあるエラーに関する詳細情報。

各セクションの名前と有無は、サービスによって異なる場合があります。

サービス固有の CLI

一部のサービスには、すべてのサービスで動作するように単一の AWS CLI が作成される前から存在する個別の CLI があります。これらのサービス固有の CLI には、サービスのドキュメントページからリンクされた個別のドキュメントがあります。サービス固有の CLI のドキュメントは AWS CLI には適用されません。

AWS CLI のコマンド構造

AWS Command Line Interface (AWS CLI) は、コマンドラインでマルチパート構造を使用し、それは次の順序で指定される必要があります。

1. `aws` プログラムのベースコール。
2. 最上位レベルの コマンド。一般に AWS CLI によってサポートされる AWS サービスに対応します。
3. 実行する操作を指定するサブコマンド。
4. 操作に必要な一般的な CLI オプションまたはパラメータ。これらは、最初の 3 つのパートに続く限り、任意の順序で指定することができます。排他的パラメータが複数回指定された場合は、最後の値のみ適用されます。

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

パラメータは数値、文字列、リスト、マップ、JSON 構造体など、様々なタイプの入力値を取得できます。サポートされる内容は、指定したコマンドおよびサブコマンドによって異なります。

AWS CLI のパラメータ値の指定

AWS Command Line Interface (AWS CLI) で使用される多くのパラメータは、以下の例のキーペア名 `my-key-pair` などのように、単純な文字列または数値です。

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

スペース文字のない文字列は引用符で囲んでも囲まなくてもかまいません。ただし、1 つ以上のスペース文字を含む文字列は引用符で囲む必要があります。Linux、macOS、Unix、または PowerShell では、単一引用符 (' ') を使用します。次の例に示すように、Windows のコマンドプロンプトで二重引用符 (" ") を使用します。

PowerShell、Linux, macOS, or Unix

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

Windows コマンドプロンプト

```
C:\> aws ec2 create-key-pair --key-name "my key pair"
```

オプションで、パラメータ名と値をスペースの代わりに等号 (=) で区切ることができます。通常、これはパラメータの値がハイフンで始まる場合にのみ必要です。

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

トピック

一般的なパラメータタイプ

このセクションでは、いくつかの一般的なパラメータタイプと一般的に必要な形式について説明します。特定のコマンドでパラメータの書式化に問題がある場合には、コマンド名の後に **help** と入力することによって、ヘルプを確認してみてください。以下に例を示します。

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

各サブコマンドのヘルプでは、関数、オプション、出力、および例について説明します。オプションのセクションでは、各オプションの名前と説明とともに括弧内にオプションのパラメータタイプが示されています。

文字列 – String パラメータには、ASCII 文字セットの英数字、記号、空白文字を含めることができます。空白文字を含む文字列は引用符で囲まれている必要があります。予期しない結果が生じる可能性があるため、標準の空白文字以外の記号や空白文字は使用しないことをお勧めします。

一部の文字列パラメータはファイルからバイナリデータを受け取ることができます。例については、「バイナリファイル (p. 48)」を参照してください。

タイムスタンプ – タイムスタンプは、ISO 8601 標準に従って書式化されます。これらは、「DateTime」または「Date」パラメータとも呼ばれます。

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

有効な形式は次のとおりです。

- *YYYY-MM-DD*T*hh*:*mm*:*ss.sss*TZD *(UTC)* (例: 2014-10-01T20:30:00.000Z)

- *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss.sssTZD (#######)* (例: 2014-10-01T12:30:00.000-08:00)
- *YYYY*-*MM*-*DD* (例: 2014-10-01)
- Unix 時間 (秒)、例: 1412195400。これは Unix エポック時間と呼ばれることもあり、1970 年 1 月 1 日午前 0 時 (UTC) からの秒数を表します。

リスト – スペースで区切られた 1 つ以上の文字列です。文字列項目にスペースがある場合は、その項目を引用符で囲む必要があります。

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

ブール – オプションをオンまたはオフにするバイナリフラグです。たとえば、`ec2 describe-spot-price-history` にはブール型の `--dry-run` パラメータがあり、指定されたときに、実際にクエリを実行せずにサービスのクエリを検証します。

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

出力にはコマンドが正しい形式だったかどうかが示されます。このコマンドには、`--no-dry-run` バージョンのパラメータも含まれ、これを使用して、コマンドを通常どおりに実行することを明示的に示すことができます。これは、デフォルトの動作であるため、含める必要はありません。

整数 – 符号なしの整数。

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

BLOB – バイナリオブジェクトです。BLOB パラメータは、バイナリデータを含んでいるローカルファイルのパスを取ります。パスにはプロトコル識別子 (`http://` や `file://` など) を含まないようにします。指定されたパスは、現在の作業ディレクトリに対する相対パスとして解釈されます。

たとえば、`aws s3api put-object` の `--body` パラメータは BLOB です。

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

マップ – JSON または CLI の短縮構文 (p. 61)を使用して指定されたキーと値のペアのセット。次の JSON の例では、マップパラメータ `--key` で my-table という名前の Amazon DynamoDB テーブルから項目を読み取ります。パラメータは、ネストされた JSON 構造の数値 1 で id という名前のプライマリキーを指定します。

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

JSON をパラメータに使用する

JSON は複雑なコマンドラインパラメータを指定する場合に便利です。たとえば、次のコマンドでは、`us-west-2c` アベイラビリティーゾーンにもあり `m1.small` または `m1.medium` のインスタンスタイプを持つすべての Amazon EC2 インスタンスをリスト表示します。

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

または、JSON 配列で同等のフィルタのリストを指定することもできます。角括弧は、カンマで区切られた JSON オブジェクトの配列を作成するために使用されます。各オブジェクトはカンマで区切られたキーと値のペアのリストです (この例では、「Name」と「Values」はいずれもキー)。

「Values」キーの右側にある値は、それ自体が配列です。配列に 1 つの値の文字列のみが含まれている場合でも、これは必要です。

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

ただし、最も外側の角括弧は、複数のフィルタが指定された場合にのみ必要です。前のコマンドの 1 つのフィルタのバージョンを JSON 形式にしたら、次のようになります。

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

一部のオペレーションでは、データを JSON 形式にする必要があります。たとえば、`--block-device-mappings` コマンドの `ec2 run-instances` パラメータにパラメータを渡すには、ブロックデバイスの情報を JSON 形式にする必要があります。

この例では、JSON で単一の 20 GiB Amazon Elastic Block Store (Amazon EBS) デバイスを指定し、起動中のインスタンスで `/dev/sdb` にマッピングします。

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

複数のデバイスにアタッチするには、次の例に示されているように、配列内のオブジェクトを一覧表示します。

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

```
      "DeleteOnTermination": true,
      "VolumeType": "standard"
    }
  }
]
```

コマンドラインで直接 JSON を入力するか (文字列に単一引用符を使用する (p. 47) を参照)、ファイルに保存しコマンドラインから参照することができます (ファイルからパラメータをロードする (p. 48) を参照)。

大量のデータを渡すときは、JSON をファイルに保存してから、コマンドラインから参照する方が簡単かもしれません。ファイル内の JSON データは、読み込み、編集、および他のユーザーとの共有がより簡単にできます。この手法については、後のセクションで説明します。

JSON の詳細については、JSON.org、Wikipedia の JSON の項目、および RFC4627 - The application/json Media Type for JSON を参照してください。

文字列に単一引用符を使用する

コマンドラインで JSON 形式のパラメータを入力する方法はオペレーティングシステムによって異なります。

Linux、macOS、または Unix

JSON データ構造を囲むには、次の例のように、単一引用符 (' ') を使用します。

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

PowerShell

次の例に示すように、PowerShell では、単一引用符 (' ') で JSON データ構造を囲み、バックスラッシュ (\) で JSON 構造内の各二重引用符 (") をエスケープする必要があります。

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

Windows コマンドプロンプト

Windows コマンドプロンプトでは、JSON データ構造を二重引用符 (" ") で囲む必要があります。その場合、次の例のように、JSON データ構造内の二重引用符 (") そのものはエスケープする (バックスラッシュ [\ ] 文字を前に付ける) 必要があります。

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

最も外側の二重引用符のみエスケープしません。

パラメータの値自体が JSON ドキュメントである場合は、埋め込み JSON ドキュメントの引用符をエスケープします。たとえば、`attribute` の `aws sqs create-queue` パラメータが `RedrivePolicy` キーであることがあります。`--attributes` パラメータは JSON ドキュメントを取り、JSON ドキュメントは `RedrivePolicy` を含み、これも JSON ドキュメントを値として取ります。外側の JSON に埋め込まれている内側の JSON はエスケープする必要があります。

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

ファイルからパラメータをロードする

コマンドラインの JSON 文字列をエスケープする必要をなくすには、JSON をファイルからロードします。ローカルファイルからパラメータをロードするには、次の例に示すように、`file://` プレフィックスを使用してファイルへのパスを提供します。ファイルパスは、現在の作業ディレクトリに対する相対パスとして解釈されます。

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
```

Windows

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

`file://` プレフィックスオプションは、「`~/`」、「`./`」、および「`../`」など、Unix 形式の拡張子をサポートしています。Windows では、「`~/`」式は、`%USERPROFILE%` 環境変数に格納されているユーザーディレクトリに展開されます。たとえば、Windows 10 では、一般にユーザーディレクトリは `C:\Users\User Name\` にあります。

別の JSON ドキュメントの値として埋め込まれている JSON ドキュメントもエスケープする必要があります。

```
$ 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\"}"
}
```

バイナリファイル

バイナリデータをパラメータとして取るコマンドでは、`fileb://` プレフィックスを使用して、データがバイナリコンテンツであることを指定します。バイナリデータを受け入れるコマンドは次のとおりです。

- **`aws ec2 run-instances`** – `--user-data` パラメータ。
- **`aws s3api put-object`** – `--sse-customer-key` パラメータ。
- **`aws kms decrypt`** – `--ciphertext-blob` パラメータ。

次の例では、Linux コマンドラインツールを使用してバイナリ 256 ビット AES キーを生成し、その後、Amazon S3 に提供してサーバー側のアップロードされたファイルを暗号化します。

```
$ 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\""
}
```

リモートファイル

AWS CLI では、`http://` または `https://` URL を使用して、インターネットでホストされているファイルからパラメータをロードすることもできます。次の例は、Amazon S3 バケットに格納されているファイルを参照しています。これにより、任意のコンピュータからパラメータファイルにアクセスできますが、コンテナが公開されている必要があります。

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

前の例では、ファイル `filename.json` が以下の JSON データを含んでいると仮定しています。

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

より複雑な JSON 形式のパラメータを含むファイルを参照する別の例については、「IAM 管理ポリシーを IAM ユーザーにアタッチする (p. 83)」を参照してください。

CLI Skeleton および CLI Input JSON パラメータの生成

AWS Command Line Interface (AWS CLI) コマンドのほとんどは、`--cli-input-json` パラメータを使用してファイルから入力されるパラメータのすべてを受け入れる機能をサポートしています。

これらのコマンドは、すべてのパラメータを含んだファイルを生成する `--generate-cli-skeleton` を備えています。ファイルは後で編集して値を入力することができます。その後、`--cli-input-json` パラメータを指定してコマンドを実行し、入力済みのファイルを指定することができます。

`--generate-cli-skeleton` パラメータを指定すると、コマンドは実行せず、パラメータテンプレートを生成して表示します。これをカスタマイズして、後でコマンドに対する入力として使用することができます。生成されるテンプレートには、そのコマンドによってサポートされているすべてのパラメータが含まれています。

たとえば、次のコマンドを実行した場合、Amazon Elastic Compute Cloud (Amazon EC2) コマンド run-instances のパラメータテンプレートが生成されます。

```
$ 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
}
```

パラメータスケルトンファイルを生成して使用するには

1. `--generate-cli-skeleton` パラメータを指定してコマンドを実行し、出力を保存用ファイルに送ります。

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

2. テキストエディタでパラメータスケルトンファイルを開き、不要なパラメータを削除します。たとえば、次のように削除できます。

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

   この例では、EC2 の dry run 機能を使用するために `DryRun` パラメータは true のままにしてあります。これにより、リソースを実際に作成したり変更したりせずに、コマンドを安全にテストできます。

3. 残りの値には、シナリオに適した値を入力します。この例では、インスタンスタイプ、キー名、セキュリティグループ、および使用する AMI の識別子を指定しています。この例では、デフォルトのリージョンを前提としています。AMI `ami-dfc39aef` は、64 ビット Amazon Linux イメージ `us-west-2` リージョンでホストされます。別のリージョンを使用する場合は、使用する正しい AMI ID を見つける必要があります。

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

4. `file://` プレフィックスを使用して JSON ファイルを `--cli-input-json` パラメータに渡すことによって、入力済みパラメータでコマンドを実行します。AWS CLI はパスを現在の作業ディレクトリに対する相対パスとして解釈するため、パスを付けずにファイル名だけを表示する次の例は、現在の作業ディレクトリ内で直接検索されます。

```
$ 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.
```

リハーサルのエラーは、JSON の形式が正しく、パラメータ値が有効であることを示します。出力に他の問題が報告された場合は、問題を修正して、"Request would have succeeded" メッセージが表示されるまで、上記のステップを繰り返します。

5. これで、`DryRun` パラメータを `false` に設定して、dry run を無効にできます。

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

6. これでコマンドを実行すると、`run-instances` は EC2 インスタンスを実際に起動し、正常起動によって生成された詳細を表示します。

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

AWS CLI からのコマンド出力の制御

このセクションでは、AWS Command Line Interface (AWS CLI) からの出力を制御するためのさまざまな方法を示します。

トピック

出力形式を選択する方法

AWS CLI は、次の 3 つの出力形式をサポートします。

- JSON (`json`)
- タブ区切りテキスト (`text`)
- ASCII 形式のテーブル (`table`)

「設定 (p. 19)」トピックで説明したように、出力形式は 3 つの異なる方法で指定できます。

- config ファイルの名前付きプロファイルでの output オプションの使用。次の例は、デフォルトの出力形式を text に設定します。

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

- AWS_DEFAULT_OUTPUT 環境変数を使用する。次の出力は、変数が変更されるか、セッションが終了するまで、このコマンドラインセッションでのコマンドの形式を table に設定します。この環境変数を使用すると、config ファイルで設定された値が上書きされます。

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

- コマンドラインの --output オプションを使用する。次の例は、この 1 つのコマンドの出力を json に設定します。このコマンドでこのオプションを使用すると、現在設定されている環境変数または config ファイルの値をオーバーライドします。

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

AWS CLI 優先順位ルール (p. 21)が適用されます。たとえば、AWS_DEFAULT_OUTPUT 環境変数を使用すると、config ファイルで設定された値をオーバーライドし、AWS CLI コマンドに --output で渡された値は、環境変数または config ファイルで設定された値をオーバーライドします。

json オプションは、さまざまな言語または jq (コマンドライン JSON プロセッサ) を介してプログラムで出力を処理するのに最適です。

table は読みやすい形式です。

text 形式は、sed、grep、awk など、従来の Unix テキスト処理ツールでも、Windows PowerShell スクリプトでも機能します。

どの形式の結果も、--query パラメータを使用してカスタマイズおよびフィルタリングすることができます。詳細については、「--query オプションを使用して出力をフィルタリングする方法 (p. 57)」を参照してください。

JSON 出力形式

JSON は AWS CLI のデフォルトの出力形式です。ほとんどの言語は、組み込み関数を使用するか、一般利用可能なライブラリを使用して、簡単に JSON 文字列をデコードできます。前のトピックで出力例と共に示したように、--query オプションでは、AWS CLI の JSON 形式の出力をフィルタ処理および書式設定する強力な機能を使用できます。

--query では可能でないことがある、より高度な機能が必要な場合は、コマンドライン JSON プロセッサの jq をお試しください。これをダウンロードし、公式のチュートリアルを http://stedolan.github.io/jq/ で見ることができます。

テキストの出力形式

テキスト形式では、AWS CLI の出力がタブ区切りの行に整形されます。grep、sed、awk など、従来の Unix テキストツールでも、PowerShell スクリプトによって実行されるテキスト処理でも機能します。

テキスト出力形式は、以下に示す基本的な構造に従います。列は、基になる JSON オブジェクトの対応するキー名によってアルファベット順にソートされます。

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

次はテキスト出力の例です。

```
$ 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
```

Important

text 出力を指定する場合は、*--query* オプションも必ず使用して、一貫した動作を確保することを強くお勧めします。これは、テキスト形式では出力列が基本の JSON オブジェクトのキー名のアルファベット順に並べられるためであり、同様のリソースが同じキー名を持つとは限らないためです。たとえば、Linux ベースの EC2 インスタンスの JSON 表現は、Windows ベースのインスタンスの JSON 表現にはない要素を持つことがあり、逆も同様です。また、リソースのキー値要素が将来の更新で追加または削除されて、列の順序が変わる可能性があります。このような場合、--query はテキスト出力の機能を補強して、出力形式に対する完全な制御を提供します。次の例では、コマンドは表示する要素を指定し、列の順序をリスト表記 `[key1, key2, ...]` で定義します。これにより、正しいキー値が常に予期される列に表示されることを確信できます。最後に、AWS CLI は存在しないキーの値として `None` を出力することに注意してください。

```
$ 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
```

次の例は、grep および awk を `aws ec2 describe-instances` コマンドからの `text` 出力で使用する方法を示しています。最初のコマンドは各インスタンスのアベイラビリティーゾーン、現在の状態、およびインスタンス ID をテキスト出力で表示します。2 番目のコマンドは、その出力を処理して、us-west-2a アベイラビリティーゾーンで実行中のすべてのインスタンスのインスタンス ID のみを表示します。

```
$ 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
```

次の例は、さらに一歩踏み込んで、出力をフィルタリングする方法だけでなく、その出力を使用して、停止した各インスタンスのインスタンスタイプの変更を自動化する方法も示しています。

```
$ 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
```

テキスト出力は、PowerShell でも使用できます。`text` 出力の列はタブ区切りであるため、PowerShell の `` `t `` 区切り記号を使用することによって簡単に配列に分割できます。次のコマンドは、最初の列 (`AvailabilityZone`) が文字列 `us-west-2a` に一致する場合に 3 列目 (`InstanceId`) の値を表示します。

```
PS C:\>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
```

> **Tip**
>
> テキスト出力を行い、--query パラメータを使用して出力を単一のフィールドにフィルタリングすると、出力は 1 行のタブ区切り値になります。次の例に示すように、各値を別々の行に入れるには、出力フィールドを角括弧で囲みます。

タブ区切りの単一の行の出力

```
$ aws iam list-groups-for-user --user-name susan  --output text --query
 "Groups[].GroupName"
HRDepartment    Developers      SpreadsheetUsers  LocalAdmins
```

`[GroupName]` を角括弧で囲むことで、各値を 1 行におさめることができます。

```
$ aws iam list-groups-for-user --user-name susan  --output text --query
 "Groups[].[GroupName]"
HRDepartment
Developers
SpreadsheetUsers
LocalAdmins
```

テーブルの出力形式

`table` 形式は、複雑な AWS CLI 出力を人間が読み取れる表現で、表形式で生成します。

```
$ 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  |||
||+--------------------------+-----------------------+-------------+--------------
+------------+----------------+||
```

--query オプションを表形式と組み合わせて、raw 出力から事前に選択された要素のセットを表示することができます。ディクショナリ表記とリスト表記の出力の違いに注意してください。最初の例では、列名はアルファベット順ですが、2 番目の例では、名前のない列がユーザーによって定義された順序になっています。--query オプションの詳細については、「--query オプションを使用して出力をフィルタリングする方法 (p. 57)」を参照してください。

```
$ 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    |
```

```
+--------------+--------------+--------------+-----+
```

--query オプションを使用して出力をフィルタリングする方法

AWS CLI は、--query オプションによって、組み込みの JSON ベースの出力フィルタリング機能を提供します。--query パラメータは、JMESPath の仕様に準拠している文字列を受け入れます。この機能を示すために、まず以下のデフォルトの JSON 出力で開始します。この出力では、2 つの Amazon Elastic Block Store (Amazon EBS) ボリュームが個別の Amazon EC2 インスタンスにアタッチされています。

```
$ 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
        }
    ]
}
```

まず、Volumes リストから最初のボリュームのみ表示されるように選択するために、配列の最初のボリュームのインデックスを作成する次のコマンドを使用します。

```
$ 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
}
```

次の例では、ワイルドカード表記 `[*]` を使用して、リストのすべてのボリュームを反復処理し、それぞれから `VolumeId`、`AvailabilityZone`、および `Size` の 3 つの要素を抽出します。ディクショナリ表記では、`{Alias1:JSONKey1,Alias2:JSONKey2}` のように、各 JSON キーのエイリアスを指定する必要があります。ディクショナリは本質的に順不同であるため、構造内のキーエイリアスの順序に一貫性がない場合があります。

```
$ 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
    }
]
```

ディクショナリ表記では、`key1.key2[0].key3` のようにキーを連結して、構造内で深く入れ子になった要素をフィルタリングすることもできます。以下の例では、単純に `Attachments[0].InstanceId` に対して `InstanceId` キーのエイリアスを作成して、これを示します。

```
$ 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
    }
]
```

リスト表記 `[key1, key2]` を使用して、複数の要素をフィルタリングすることもできます。これにより、フィルタリングされたすべての属性が、型に関係なく、オブジェクトごとに 1 つの順序付きリスト形式になります。

```
$ 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
    ]
]
```

特定のフィールドの値によって結果をフィルタ処理するには、JMESPath "?" 演算子を使用します。次のクエリの例では、us-west-2a アベイラビリティーゾーンのボリュームのみを出力します。

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

> **Note**
>
> JMESPath クエリ式で上記の "us-west-2" のようなリテラル値を指定するときは、適切に読み込まれるように、値をバックティック (` `) で囲む必要があります。

コマンドの出力から必要な詳細のみを取得する方法を示す他の例を以下に示します。

以下の例では、Amazon EC2 ボリュームを表示します。このサービスでは、us-west-2a アベイラビリティーゾーンの使用中のすべてのボリュームのリストが生成されます。--query パラメータでは、Size 値が 50 を超えるボリュームにのみ出力を制限し、ユーザー定義の名前を持つ指定されたフィールドのみ表示されます。

```
$ aws ec2 describe-volumes \
                                                          ~/workplace/awscli/src/
AWSUnifiedCLIDocs
--filter "Name=availability-zone,Values=us-west-2a" "Name=status,Values=attached" \
--query 'Volumes[?Size > `50`].{Id:VolumeId,Size:Size,Type:VolumeType}'
[
    {
        "Id": "vol-0be9bb0bf12345678",
        "Size": 80,
        "Type": "gp2"
    }
]
```

次の例では、いくつかの基準を満たすイメージのリストを取得します。--query パラメータを使用して、CreationDate で出力を絞り込み、最新のイメージのみ選択します。その結果、1 つのイメージの ImageId が表示されます。

```
$ aws ec2 describe-images \
--owners amazon \
--filters "Name=name,Values=amzn*gp2" "Name=virtualization-type,Values=hvm" "Name=root-
device-type,Values=ebs" \
--query "sort_by(Images, &CreationDate)[-1].ImageId" \
--output text
ami-00ced3122871a4921
```

また、--query パラメータを使用して、出力の項目をカウントすることもできます。次の例では、1000 IOPS を超える利用可能なボリュームの数を表示します。

```
$ aws ec2 describe-volumes \
                                                              ~/workplace/awscli/src/
AWSUnifiedCLIDocs
--filter "Name=status,Values=available" \
--query 'length(Volumes[?Iops > `1000`])'
3
```

次の例は、指定された日付以降に作成されたすべてのスナップショットを一覧表示する方法を示しています (例: 出力の利用可能な一部のフィールド)。

```
$ aws ec2 describe-snapshots --owner self --output json \
                                                          ~/workplace/awscli/src/
AWSUnifiedCLIDocs
--query 'Snapshots[?StartTime>=`2018-02-07`].{Id:SnapshotId,VId:VolumeId,Size:VolumeSize}'
 \
[
    {
        "id": "snap-0effb42b7a1b2c3d4",
        "vid": "vol-0be9bb0bf12345678",
        "Size": 8
    }
]
```

次の例では、作成した最新の 5 つの AMI を最新のものから古いものの順に並べ替えています。

```
$ aws ec2 describe-images --owners self \
                                                              ~/workplace/awscli/src/
AWSUnifiedCLIDocs
--query 'reverse(sort_by(Images,&CreationDate))[:5].{id:ImageId,date:CreationDate}'
[
    {
        "id": "ami-0a1b2c3d4e5f60001",
        "date": "2018-11-28T17:16:38.000Z"
    },
    {
        "id": "ami-0a1b2c3d4e5f60002",
        "date": "2018-09-15T13:51:22.000Z"
    },
    {
        "id": "ami-0a1b2c3d4e5f60003",
        "date": "2018-08-19T10:22:45.000Z"
    },
    {
        "id": "ami-0a1b2c3d4e5f60004",
        "date": "2018-05-03T12:04:02.000Z"
    },
    {
        "id": "ami-0a1b2c3d4e5f60005",
        "date": "2017-12-13T17:16:38.000Z"
    }

]
```

次の例では、指定した AutoScaling グループ内の異常なインスタンスの `InstanceId` のみを表示しています。

```
$ aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name My-AutoScaling-
Group-Name --output text\
```

```
--query 'AutoScalingGroups[*].Instances[?HealthStatus==`Unhealthy`].InstanceId'
```

--query オプションは、以下のセクションで詳細に説明する 3 つの出力形式との組み合わせにより、出力の内容とスタイルをカスタマイズするために使用できる強力なツールです。

JMESPath の詳細な例と完全な仕様、基盤となる JSON 処理ライブラリについては、「http://jmespath.org/specification.html」を参照してください。

AWS Command Line Interface を使用した短縮構文の使用

AWS Command Line Interface (AWS CLI) は、JSON 形式の多くのオプションパラメータを受け入れることができます。ただし、大きな JSON リストや構造体をコマンドラインに入力するのは手間がかかる場合があります。これを簡単にするために、AWS CLI は短縮構文もサポートしているため、完全な JSON 形式を使用するより、オプションパラメータを簡単に表現できます。

構造パラメータ

AWS CLI の短縮構文を利用すると、ユーザーがフラットなパラメータ (ネストされていない構造) に入力するのが容易になります。形式は、キーと値のペアのカンマ区切りリストです。

Linux, macOS, or Unix

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

PowerShell

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

これらはいずれも、JSON で書式化された次の例と同じです。

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

それぞれのカンマ区切りのキーと値のペアの間に空白があってはいけません。これは、省略表現で指定された --provisioned-throughput オプションを使用した Amazon DynamoDB update-table コマンドの例です。

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

これは、JSON 形式の次の例と同じになっています。

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

リストパラメータ

リストフォーム内の入力パラメータは、JSON または省略形の 2 つの方法で指定できます。AWS CLI の短縮構文は、数値、文字列、またはネストされていない構造体が含まれるリストを簡単に渡せるように設計されています。

基本的な形式を次に示します。ここで、リストの値は、1 つのスペースで区切られます。

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

これは、JSON 形式の次の例と同じになっています。

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

前述したように、数字のリスト、文字列のリスト、またはネストされていない構造の省略表現のリストを指定できます。以下に示しているのは、Amazon Elastic Compute Cloud (Amazon EC2) 用の `stop-instances` コマンドの例です。ここで、`--instance-ids` オプションの入力パラメータ (文字列のリスト) は省略表現で指定しています。

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

これは、JSON 形式の次の例と同じになっています。

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

次の例は、Amazon EC2 `create-tags` コマンドを示しています。これは `--tags` オプションとして、入れ子になっていない構造体のリストを取ります。`--resources` オプションは、タグを付けるインスタンスの ID を指定します。

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

これは、JSON 形式の次の例と同じになっています。JSON パラメータは、読みやすくするために複数行で記述されます。

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

AWS CLI のページ分割オプションの使用

項目の大きなリストを返すことができるコマンドの場合、AWS Command Line Interface (AWS CLI) にはさらに 3 つのオプションがあり、それらを使用して、AWS CLI がサービスの API を呼び出してリストを生成するときに出力に含まれる項目の数を制御できます。

デフォルトでは、AWS CLI は 1,000 のページサイズを使用して、利用可能なすべての項目を取得します。たとえば、3,500 のオブジェクトを含む Amazon S3 バケットに対して `aws s3api list-objects` を実行した場合、CLI は Amazon S3 の呼び出しを 4 回行って、サービス固有のページ分割ロジックをバックグラウンドで自動的に処理し、最終的な出力で 3,500 すべてのオブジェクトを返します。

大量のリソースに対してリストコマンドを実行しているときに問題がある場合、1,000 というデフォルトのページサイズが高すぎる可能性があります。これにより、AWS サービスの呼び出しが最大許容時間を超えて、「タイムアウト」エラーを生成することがあります。`--page-size` オプションを使用して、AWS CLI が AWS のサービスの 1 回の呼び出しで要求する項目数を少なくすることができます。その場合でも、CLI は完全なリストを取得しますが、多数のサービス API コールをバックグラウンドで実行し、1 回の呼び出しで取得する項目数が少なくなります。このため、個々の呼び出しがタイムアウトにならずに成

功する可能性が高くなります。ページサイズを変更しても、出力には影響しません。出力を生成するために必要な API 呼び出しの数が変わるだけです。

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

AWS CLI 出力で一度に含める項目を少なくするには、--max-items オプションを使用します。前述したように、AWS CLI はサービスとのページ区切りを処理しますが、指定した時点での項目数のみを出力します。

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

出力される項目数 (--max-items) が基本の API 呼び出しによって返される合計項目数より少ない場合、出力には NextToken が含まれ、これにより、後続のコマンドを渡して、次の項目のセットを取得できます。次の例は、前の例で返された NextToken 値を使用して、2 番目の 100 項目を取得する方法を示しています。

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

指定された AWS サービスは、呼び出しごとに同じ順序で項目を返さないことがあります。--page-size と --max-items に異なる値を指定した場合、項目の不足や重複など、予期しない結果になることがあります。これを防ぐには、--page-size と --max-items に同じ数を使用して、AWS CLI のページ分割と基本のサービスを同期させます。リスト全体を取得し、必要な解析オペレーションをローカルで実行することもできます。

AWS CLI を使用して AWS サービスを使用する

このセクションでは、AWS Command Line Interface (AWS CLI) を使用してさまざまな AWS サービスにアクセスする方法を示す例を紹介します。

各サービスで使用可能なすべてのコマンドの詳細については、「AWS CLI Command Reference」を参照するか、組み込みコマンドラインのヘルプを使用してください。詳細については、「AWS CLI のヘルプ (p. 39)」を参照してください。

トピック

AWS CLI での Amazon DynamoDB の使用

AWS Command Line Interface (AWS CLI) は、Amazon DynamoDB も含め、すべての AWS データベースサービスをサポートしています。テーブルの作成など、その場限りのオペレーションに AWS CLI を使用できます。また、ユーティリティスクリプト内に DynamoDB オペレーションを埋め込むときにも使用できます。

DynamoDB 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws dynamodb help
```

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

コマンドラインの形式は、Amazon DynamoDB API 名の後に、その API のパラメータが続きます。AWS CLI は、完全な JSON のほか、パラメータ値の CLI 短縮構文 (p. 61)をサポートしています。

たとえば、次のコマンドは、`MusicCollection` という名前のテーブルを作成します。

> **Note**
>
> 読みやすくするために、このセクションの長いコマンドは、複数の行に分かれています。バックスラッシュ文字は、Linux コマンドラインの行連結文字であり、Linux プロンプトに対して複数の行をコピーして貼り付け (または入力) できます。文字のエスケープにバックスラッシュを使用しないシェルを使用している場合は、バックスラッシュを別のエスケープ文字に置き換えます。または、バックスラッシュを削除してコマンド全体を一行にしてください。

```
$ 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
```

次の例に示されているようなコマンドと同様のコマンドで、新しい行をテーブルに追加できます。この例では、短縮構文と JSON を組み合わせて使用しています。

```
$ 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"
    }
}
```

有効な JSON を 1 行のコマンドで作成するのは難しい場合があります。これを簡単にするために、AWS CLI は JSON ファイルを読み取ることができます。たとえば、`expression-attributes.json` という名前のファイルに格納されている次の JSON スニペットがあるとします。

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

そのファイルを使用して、AWS CLI を使用する `query` リクエストを発行することができます。次の例では、`expression-attributes.json` ファイルの内容が `--expression-attribute-values` パラメータに使用されます。

```
$ 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
}
```

DynamoDB での AWS CLI の使用の詳細については、AWS CLI Command Reference の DynamoDB を参照してください。

DynamoDB に加えて、DynamoDB Local も AWS CLI で使用できます。DynamoDB Local は、小規模のクライアント側データベースとサーバーで、DynamoDB サービスに似せて作られています。DynamoDB Local では、DynamoDB ウェブサービスでテーブルまたはデータを実際に操作しなくても、DynamoDB API を使用するアプリケーションを作成することができます。すべての API アクションがローカルデータベースに転送されます。これにより、プロビジョニングされたスループット、データストレージ、およびデータ転送料金を節約できます。

DynamoDB Local の詳細および AWS CLI での使用方法については、Amazon DynamoDB 開発者ガイド の以下のセクションを参照してください。

- DynamoDB Local
- DynamoDB Local での AWS CLI の使用

AWS CLI での Amazon EC2 の使用

AWS Command Line Interface (AWS CLI) を使用して Amazon Elastic Compute Cloud (Amazon EC2) の機能にアクセスできます。Amazon EC2 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws ec2 help
```

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

このトピックでは、Amazon EC2 の一般的なタスクを実行する AWS CLI コマンドの例を示します。

トピック

Amazon EC2 キーペアの作成、表示、および削除

AWS Command Line Interface (AWS CLI) を使用して、Amazon EC2 のキーペアを作成、表示、および削除することができます。キーペアは、Amazon EC2 インスタンスに接続するときに使用します。

インスタンスを作成するときに Amazon EC2 にキーペアを指定する必要があり、インスタンスに接続するときには、そのキーペアを使用して認証する必要があります。

Note

以下の例では、デフォルトの認証情報を設定済みであること (p. 66)を想定しています。

トピック

キーペアを作成する

キーペアを作成するには、create-key-pair コマンドを --query オプションおよび --output text オプションとともに使用して、プライベートキーをファイルに直接パイプします。

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

PowerShell では、> file リダイレクトはデフォルトで UTF-8 エンコードされます。これは一部の SSH クライアントでは使用できません。そのため、出力を out-file コマンドにパイプすることによって変換し、エンコードを ascii に明示的に設定する必要があります。

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

出力された MyKeyPair.pem ファイルは以下のようになります。

```
-----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-----
```

プライベートキーは AWS に保存されず、作成時にのみ取得することができます。後で復元することはできません。代わりに、プライベートキーを紛失した場合は、新しいキーペアを作成する必要があります。

Linux コンピュータからインスタンスに接続している場合は、他のユーザーが読み取れないように、次のコマンドを使用してプライベートキーファイルのアクセス許可を設定することをお勧めします。

```
$ chmod 400 MyKeyPair.pem
```

キーペアの表示

「フィンガープリント」はキーペアから生成され、これを使用してローカルマシンのプライベートキーが AWS に保存されたパブリックキーと一致することを確認できます。

フィンガープリントは、DER でエンコードされたプライベートキーのコピーから取得される SHA1 ハッシュです。この値はキーペアの作成時にキャプチャされて、パブリックキーとともに AWS に格納されます。フィンガープリントは、Amazon EC2 コンソールを使用するか、AWS CLI コマンド `aws ec2 describe-key-pairs` を実行して表示することができます。

次の例は、`MyKeyPair` のフィンガープリントを表示します。

```
$ 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"
        }
    ]
}
```

キーおよびフィンガープリントの詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EC2 キーペア」を参照してください。

キーペアの削除

キーペアを削除するには、*MyKeyPair* を削除するペアの名前に置き換えて、次のコマンドを実行します。

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

Amazon EC2 のセキュリティグループの作成、設定、および削除

基本的にファイアウォールとして動作する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスのセキュリティグループを、進入と退出が可能なネットワークトラフィックを決めるルールとともに作成することができます。仮想プライベートクラウド (VPC) または EC2-Classic 共有フラットネットワークで使用するセキュリティグループを作成することができます。EC2-Classic と EC2-VPC の違いの詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイド のサポートされているプラットフォームを参照してください。

AWS Command Line Interface (AWS CLI) を使用して新しいセキュリティグループを作成し、既存のセキュリティグループにルールを追加して、セキュリティグループを削除することができます。

> **Note**
>
> 下の例では、デフォルトの認証情報を設定済みであること (p. 66)を想定しています。

トピック

セキュリティグループを作成する

VPC または EC2-Classic に関連付けられたセキュリティグループを作成することができます。

EC2-VPC

次の例は、指定された VPC のセキュリティグループを作成する方法を示しています。

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

セキュリティグループの初期情報を表示するには、describe-security-groups コマンドを実行します。EC2-VPC セキュリティグループは、名前ではなく `vpc-id` によってのみ参照することができます。

```
$ 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

次の例は、EC2-Classic のセキュリティグループを作成する方法を示しています。

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

`my-sg` の初期情報を表示するには、describe-security-groups コマンドを実行します。EC2-Classic セキュリティグループは名前で参照することができます。

```
$ aws ec2 describe-security-groups --group-names my-sg
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [],
```

```
            "Description": "My security group"
            "IpPermissions": [],
            "GroupName": "my-sg",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

ルールを セキュリティグループに追加する

Amazon EC2 インスタンスを実行するときには、セキュリティグループのルールを有効にして、イメージに接続する手段としての着信ネットワークトラフィックを有効にする必要があります。

たとえば、Windows インスタンスを起動する場合、Remote Desktop Protocol (RDP) をサポートするには、一般に TCP ポート 3389 へのインバウンドトラフィックを許可するルールを追加します。Linux インスタンスを起動する場合、SSH 接続をサポートするには、一般に TCP ポート 22 へのインバウンドトラフィックを許可するルールを追加します。

セキュリティグループにルールを追加するには、`authorize-security-group-ingress` コマンドを使用します。このコマンドの必須パラメータは、コンピュータのパブリック IP アドレス、またはコンピュータが接続しているネットワーク (アドレス範囲の形式で) の CIDR 表記です。

> Note
>
> 当社では、パブリック IP アドレスを調べることができるサービス (https://checkip.amazonaws.com/) を提供しています。IP アドレスの識別に役立つその他のサービスを見つけるには、ブラウザを使用して「what is my IP address」を検索します。ISP 経由、またはファイアウォールの内側から動的な IP アドレスを使用して接続している場合 (プライベートネットワークの NAT ゲートウェイ経由)、アドレスは定期的に変更される場合があります。その場合、クライアントコンピュータによって使用される IP アドレスの範囲を見つける必要があります。

EC2-VPC

次の例は、ID `sg-903004f8` を持つ EC2-VPC セキュリティグループに RDP (TCP ポート 3389) のルールを追加する方法を示しています。この例では、クライアントコンピュータが CIDR 範囲 203.0.113.0/24 のアドレスを持つことを前提としています。

まず、パブリックアドレスが CIDR 範囲 203.0.113.0/24 に含まれていることを確認します。

```
$ curl https://checkip.amazonaws.com
203.0.113.57
```

この情報が確認できたら、`authorize-security-group-ingress` を実行することによって、セキュリティグループに範囲を追加することができます。

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

次のコマンドは、同じセキュリティグループのインスタンスに SSH を有効にする別のルールを追加します。

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

セキュリティグループに加えられた変更を表示するには、describe-security-groups コマンドを実行します。

```
$ 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

次のコマンドは、*my-sg* という名前の EC2-Classic セキュリティグループに RDP のルールを追加します。

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

次のコマンドは、同じセキュリティグループに SSH の別のルールを追加します。

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

セキュリティグループに加えられた変更を表示するには、describe-security-groups コマンドを実行します。

```
$ 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"
        }
    ]
}
```

セキュリティグループの削除

セキュリティグループを削除するには、delete-security-group コマンドを実行します。

Note

環境に現在アタッチされているセキュリティグループは削除できません。

EC2-VPC

次のコマンドは、EC2-VPC セキュリティグループを削除します。

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

EC2-Classic

次のコマンドは、my-sg という名前の EC2-Classic セキュリティグループを削除します。

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

Amazon EC2 インスタンスを起動、リスト、および終了する

AWS Command Line Interface (AWS CLI) を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを起動、一覧表示、終了できます。キーペア (p. 66)とセキュリティグループ (p. 68)が必要です。また、Amazon マシンイメージ (AMI) を選択し、AMI ID を書きとめておく必要があります。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Linux AMI の検索」を参照してください。

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

Note

以下の例では、デフォルトの認証情報を設定済みであること (p. 66)を想定しています。

トピック

インスタンスの起動

選択した AMI を使用して Amazon EC2 インスタンスを起動するには、run-instances コマンドを使用します。インスタンスは、仮想プライベートクラウド (VPC) か、アカウントでサポートされている場合は EC2-Classic に起動することができます。

当初、インスタンスは `pending` 状態で表示されますが、数分後に `running` 状態に変わります。

EC2-VPC

以下の例は、`t2.micro` インスタンスを指定された VPC のサブネットで起動する方法を示しています。*########*パラメータ値をユーザー自身の値に置き換えてください。

```
$ aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-
name MyKeyPair --security-group-ids sg-903004f8 --subnet-id subnet-6e7f829e
{
    "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

アカウントがサポートしている場合は、次のコマンドを使用して `t1.micro` インスタンスを EC2-Classic で起動することができます。*#########*パラメータ値をユーザー自身の値に置き換えてください。

```
$ aws ec2 run-instances --image-id ami-173d747e --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
        }
    ]
}
```

インスタンスへのブロックデバイスの追加

起動する各インスタンスにはルートデバイスボリュームが関連付けられています。ブロックデバイスマッピングを使用すると、インスタンスの起動時にそのインスタンスにアタッチする追加の Amazon Elastic Block Store (Amazon EBS) ボリュームまたはインスタンスストアボリュームを指定できます。

ブロックデバイスをインスタンスに追加するには、`run-instances` を使用するときに `--block-device-mappings` オプションを指定します。

次の例のパラメータは、20 GB のサイズの標準 Amazon EBS ボリュームをプロビジョニングし、識別子 `/dev/sdf` を使用してインスタンスにマッピングします。

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

次の例は、既存のスナップショットに基づいて `/dev/sdf` にマッピングされる Amazon EBS ボリュームを追加します。スナップショットは、ボリュームに自動的にロードされるイメージを表します。スナップショットを指定するとき、ボリュームサイズを指定する必要はありません。イメージを保持できる十分な大きさになります。ただし、サイズを指定する場合は、スナップショットのサイズ以上である必要があります。

```
--block-device-mappings "[{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":\"snap-
a1b2c3d4\"}}]"
```

次の例は、2 つのボリュームをインスタンスに追加します。インスタンスで使用できるボリュームの数は、インスタンスタイプによって異なります。

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

次の例では、マッピング (`/dev/sdj`) を作成しますが、インスタンスのボリュームはプロビジョニングされません。

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

詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「ブロックデバイスのマッピング」を参照してください。

インスタンスにタグを追加する

タグとは、AWS リソースに付けるラベルです。リソースにメタデータを追加して、さまざまな目的に使用できます。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EC2 リソースにタグを付ける」を参照してください。

次の例は、create-tags コマンドを使用して、キー名「Name」と値「MyInstance」を持つタグを指定されたインスタンスに追加する方法を示しています。

```
$ aws ec2 create-tags --resources i-5203422c --tags Key=Name,Value=MyInstance
```

インスタンスへの接続

実行中のインスタンスに接続して、目の前にあるコンピュータと同じように使用することができます。詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイド の「Amazon EC2 インスタンスへの接続」を参照してください。

インスタンスの一覧表示

AWS CLI を使用して、インスタンスを一覧表示し、それらの情報を表示できます。すべてのインスタンスを一覧表示することも、目的のインスタンスに基づいて結果をフィルタリングすることもできます。

次の例では、describe-instances コマンドの使用方法を示しています。

次のコマンドは、リストを t2.micro インスタンスのみにフィルタリングして、一致した InstanceId 値のみを出力します。

```
$ aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro" --query
 "Reservations[].Instances[].InstanceId"
[
    "i-05e998023d9c69f9a"
]
```

次のコマンドは、タグ Name=MyInstance を持つインスタンスをリストします。

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

次のコマンドは、ami-x0123456、ami-y0123456、および ami-z0123456 のいずれかの AMI を使用して起動されたインスタンスをリストします。

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

インスタンスを削除する

インスタンスを終了すると、インスタンスが削除されます。インスタンスの終了後に、インスタンスに再接続することはできません。

インスタンスの状態が shutting-down または terminated に変わったら、そのインスタンスへの課金は停止します。後でインスタンスに再接続する必要がある場合は、terminate-instances の代わりに stop-instances を使用します。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの終了」を参照してください。

インスタンスの仕様が終了したら、コマンド terminate-instances を使用して削除することができます。

```
$ aws ec2 terminate-instances --instance-ids i-5203422c
{
```

```
    "TerminatingInstances": [
        {
            "InstanceId": "i-5203422c",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

AWS CLI での Amazon S3 Glacier の使用

AWS Command Line Interface (AWS CLI) を使用して Amazon S3 Glacier の機能にアクセスできます。S3 Glacier 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws glacier help
```

このトピックでは、S3 Glacier の一般的なタスクを実行する AWS CLI コマンドの例を示します。以下の例は、AWS CLI を使用して、大きなファイルを小さく分割し、コマンドラインからアップロードすることによって Glacier にアップロードする方法を示しています。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

> Note
>
> このチュートリアルでは、通常 Linux や OS X のような Unix に似たオペレーティングシステムにプリインストールされている複数のコマンドラインツールを使用します。Windows ユーザーは、Cygwin をインストールして Cygwin ターミナルからコマンドを実行することで、同じツールを使用できます。同じ機能を実行する Windows のネイティブコマンドとユーティリティはそのように注記されています。

トピック

Amazon S3 Glacier ボールトの作成

`create-vault` コマンドを使用してボールトを作成します。

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

> Note
>
> すべての S3 Glacier コマンドにはアカウント ID パラメータが必要です。現在のアカウントを使用するには、ハイフン文字を使用します (`--account-id -`)。

アップロードするファイルの準備

テストアップロード用のファイルを作成します。以下のコマンドは、ちょうど 3 MiB のランダムデータを含む *largefile* という名前のファイルを作成します。

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
```

dd は、多数のバイトを入力ファイルから出力ファイルにコピーするユーティリティです。前の例では、ランダムデータのソースとしてシステムデバイスファイル /dev/urandom を使用します。fsutil は Windows で同様の関数を実行します。

Windows

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

次に、ファイルを 1 MiB (1,048,576 バイト) のチャンクに分割します。

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

> **Note**
>
> HJ-Split は、Windows および他の多くのプラットフォームで無料で使用できるファイルスプリッタです。

マルチパートアップロードの開始とファイルのアップロード

initiate-multipart-upload コマンドを使用して、Amazon S3 Glacier でマルチパートアップロードを作成します。

```
$ 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"
}
```

S3 Glacier では、マルチパートアップロードを設定するために各パートのサイズ (バイト) (この例では 1 MiB)、ボールト名、アカウント ID が必要です。オペレーションが完了すると、AWS CLI によってアップロード ID が出力されます。後で使用できるように、アップロード ID をシェル変数に保存します。

Linux, macOS, or Unix

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

Windows

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

次に、upload-multipart-part コマンドを使用して、3 つのパートのそれぞれをアップロードします。

```
$ 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**
>
> 前の例では、ドル記号 ($) を使用して Linux の UPLOADID シェル変数の内容を参照しています。Windows コマンドラインでは、変数名の両側にパーセント記号 (%) を使用します (例: %UPLOADID%)。

Glacier によって正しい順序で再構成されるように、アップロード時には各パートのバイト範囲を指定する必要があります。各部分は 1,048,576 バイトであるため、1 番目の部分は 0-1048575、2 番目は 1048576-2097151、3 番目は 2097152-3145727 に配置されます。

アップロードの完了

Amazon S3 Glacier では、アップロードされたすべての部分が損なわれることなく AWS に到達したことを確認するために、元のファイルの木構造ハッシュが必要です。

木構造ハッシュを計算するには、ファイルを 1 MiB のパートに分割し、各部分のバイナリ SHA-256 ハッシュを計算する必要があります。次に、ハッシュのリストをペアに分割し、各ペアの 2 つのバイナリハッシュを結合して、結果のハッシュを取得します。ハッシュが 1 つだけになるまでこのプロセスを繰り返します。レベルのいずれかに奇数のハッシュがある場合は、変更せずに次のレベルに昇格させます。

コマンドラインユーティリティを使用して木構造ハッシュを正しく計算するために重要なことは、各ハッシュをバイナリ形式で保存し、最後のステップでのみ 16 進数に変換することです。木構造で 16 進数バージョンのハッシュを結合またはハッシュすると、正しい結果を得ることができません。

> **Note**
>
> Windows ユーザーは、cat の代わりに type コマンドを使用できます。Windows 用の OpenSSL は OpenSSL.org で入手できます。

木構造ハッシュを計算するには

1. 元のファイルを分割していない場合は、1 MiB のパートに分割します。

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

2. 各チャンクのバイナリ SHA-256 ハッシュを計算して保存します。

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

3. 最初の 2 つのハッシュ結合を実行し、結果のバイナリハッシュを取得します。

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

4. チャンク aa および ab の親ハッシュをチャンク ac のハッシュと結合して結果をハッシュします。今回は 16 進数で出力します。シェル変数に結果を保存します。

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

最後に、complete-multipart-upload コマンドを使用してアップロードを完了します。このコマンドは、元のファイルのサイズ (バイト単位)、最終的な 16 進数の木構造ハッシュ値、およびアカウント ID とボールト名を使用します。

```
$ 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"
}
```

describe-vault コマンドを使用して、ボールトのステータスを確認することもできます。

```
$ aws glacier describe-vault --account-id - --vault-name myvault
{
    "SizeInBytes": 3178496,
    "VaultARN": "arn:aws:glacier:us-west-2:123456789012:vaults/myvault",
    "LastInventoryDate": "2018-12-07T00:26:19.028Z",
    "NumberOfArchives": 1,
    "CreationDate": "2018-12-06T21:23:45.708Z",
    "VaultName": "myvault"
}
```

> **Note**
>
> ボールトのステータスは約 1 日 1 回更新されます。詳細については、「Glacier のボールトに関する各種操作」を参照してください。

これで、作成したチャンクおよびハッシュファイルを安全に削除できます。

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

マルチパートアップロードの詳細については、『Amazon S3 Glacier 開発者ガイド』の「パート単位での大きなアーカイブのアップロード」と「チェックサムの計算」を参照してください。

AWS CLI からの AWS Identity and Access Management の使用

AWS Command Line Interface (AWS CLI) を使用して AWS Identity and Access Management (IAM) の機能にアクセスできます。IAM 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws iam help
```

このトピックでは、IAM の一般的なタスクを実行する AWS CLI コマンドの例を示します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

トピック

IAM ユーザーおよびグループの作成

このトピックでは、AWS Command Line Interface (AWS CLI) コマンドを使用して IAM グループおよび新しい IAM ユーザーを作成し、そのユーザーをグループに追加する方法について説明します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

IAM グループを作成して新しい IAM ユーザーを追加するには

1. グループを作成するには、`create-group` コマンドを使用します。

```
$ aws iam create-group --group-name MyIamGroup
{
    "Group": {
        "GroupName": "MyIamGroup",
        "CreateDate": "2018-12-14T03:03:52.834Z",
        "GroupId": "AGPAJNUJ2W4IJVEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    }
}
```

2. ユーザーを作成するには、`create-user` コマンドを使用します。

```
$ aws iam create-user --user-name MyUser
{
```

```
    "User": {
        "UserName": "MyUser",
        "Path": "/",
        "CreateDate": "2018-12-14T03:13:02.581Z",
        "UserId": "AIDAJY2PE5XUZ4EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/MyUser"
    }
}
```

3. add-user-to-group コマンドを使用して、そのユーザーをグループに追加します。

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

4. MyIamGroup グループが MyUser を含んでいることを確認するには、get-group コマンドを使用します。

```
$ aws iam get-group --group-name MyIamGroup
{
    "Group": {
        "GroupName": "MyIamGroup",
        "CreateDate": "2018-12-14T03:03:52Z",
        "GroupId": "AGPAJNUJ2W4IJVEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    },
    "Users": [
        {
            "UserName": "MyUser",
            "Path": "/",
            "CreateDate": "2018-12-14T03:13:02Z",
            "UserId": "AIDAJY2PE5XUZ4EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/MyUser"
        }
    ],
    "IsTruncated": "false"
}
```

IAM 管理ポリシーを IAM ユーザーにアタッチする

このトピックでは、AWS Command Line Interface (AWS CLI) コマンドを使用して IAM ポリシーを IAM ユーザーにアタッチする方法について説明します。この例のポリシーは、ユーザーに「Power User Access」を付与します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

IAM 管理ポリシーを IAM ユーザーにアタッチするには

1. アタッチするポリシーの ARN を決定します。次のコマンドでは、list-policies を使用して、PowerUserAccess という名前のポリシーの ARN を検索します。次に、その ARN を環境変数に格納します。

```
$ export POLICYARN=$(aws iam list-policies --query 'Policies[?
PolicyName==`PowerUserAccess`].{ARN:Arn}' --output text)       ~
$ echo $POLICYARN
arn:aws:iam::aws:policy/PowerUserAccess
```

2. ポリシーをアタッチするには、attach-user-policy コマンドを使用し、ポリシー ARN を保持している環境変数を参照します。

```
$ aws iam attach-user-policy --user-name MyUser --policy-arn $POLICYARN
```

3. ポリシーがユーザーにアタッチされたことを確認するには、list-attached-user-policies コマンドを実行します。

```
$ aws iam list-attached-user-policies --user-name MyUser
{
    "AttachedPolicies": [
        {
            "PolicyName": "PowerUserAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
        }
    ]
}
```

その他のリソース

詳細については、「アクセス管理リソース」を参照してください。このトピックでは、アクセス許可とポリシーの概要へのリンク、Amazon S3、Amazon EC2、およびその他のサービスにアクセスするポリシーの例へのリンクを示しています。

IAM ユーザーの初期パスワードを設定する

このトピックでは、AWS Command Line Interface (AWS CLI) コマンドを使用して IAM ユーザーの初期パスワードを設定する方法について説明します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

次のコマンドは、create-login-profile を使用して、指定されたユーザーの初期パスワードを設定します。初回サインイン時、ユーザーは自分だけが知っているパスワードに変更するように求められます。

```
$ aws iam create-login-profile --user-name MyUser --password My!User1Login8P@ssword --
password-reset-required
{
    "LoginProfile": {
        "UserName": "MyUser",
        "CreateDate": "2018-12-14T17:27:18Z",
        "PasswordResetRequired": true
    }
}
```

update-login-profile コマンドを使用して、IAM ユーザーのパスワードを変更することができます。

```
$ aws iam update-login-profile --user-name MyUser --password My!User1ADifferentP@ssword
```

IAM ユーザーのアクセスキーを作成する

このトピックでは、AWS Command Line Interface (AWS CLI) コマンドを使用して IAM ユーザーの一連のアクセスキーを作成する方法について説明します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

create-access-key コマンドを使用して、IAM ユーザーのアクセスキーを作成することができます。アクセスキーは、アクセスキー ID とシークレットキーで構成される一連のセキュリティ認証情報です。

IAM ユーザーが作成できるアクセスキーは一度に 2 つのみです。3 番目のセットを作成しようとすると、コマンドは `LimitExceeded` エラーを返します。

```
$ aws iam create-access-key --user-name MyUser
{
    "AccessKey": {
        "UserName": "MyUser",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "Status": "Active",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "CreateDate": "2018-12-14T17:34:16Z"
    }
}
```

delete-access-key コマンドを使用して、IAM ユーザーのアクセスキーを削除します。アクセスキー ID を使用して、削除するアクセスキーを指定します。

```
$ aws iam delete-access-key --user-name MyUser --access-key-id AKIAIOSFODNN7EXAMPLE
```

AWS CLI での Amazon S3 の使用

AWS Command Line Interface (AWS CLI) を使用して Amazon Simple Storage Service (Amazon S3) の機能にアクセスできます。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

AWS CLI には、Amazon S3 へのアクセス用に 2 つの階層のコマンドが用意されています。

- s3 層は、オブジェクトおよびバケットの作成、操作、削除など、一般的なタスクの実行を簡素化する高レベルコマンドで構成されます。
- s3api 層は他の AWS サービスと同様に動作し、すべての Amazon S3 API 操作への直接アクセスを公開します。これにより、次の層の高レベルコマンドだけでは不可能なアドバンスドオペレーションを実行できるようになります。

各層で使用可能なすべてのコマンドのリストを取得するには、`aws s3api` または `aws s3` コマンドで `help` 引数を使用します。

```
$ aws s3 help
```

```
$ aws s3api help
```

Note

AWS CLI は、Amazon S3 によって提供されるサーバーサイドの `COPY` オペレーションを使用して、Amazon S3 から Amazon S3 へのコピー、移動、および同期をサポートします。つまり、ファイルはクラウドに保持され、クライアントマシンにはダウンロードされず、Amazon S3 にバックアップされます。
このようなオペレーションがクラウド内で完全に実行される場合、HTTP リクエストおよびレスポンスに必要な帯域幅のみが使用されます。

Amazon S3 の使用状況の例については、このセクションの以下のトピックを参照してください。

トピック

AWS CLI での高レベル (s3) コマンドの使用

このトピックでは、高レベル `aws s3` コマンドを使用して Amazon S3 バケットおよびオブジェクトを管理する方法について説明します。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

バケットの管理

高レベル `aws s3` コマンドは、バケットの作成、一覧表示、削除など、一般的なバケット操作をサポートしています。

バケットの作成

バケットを作成するには、`s3 mb` コマンドを使用します。バケット名はグローバルに一意でなければならず、DNS に準拠する必要があります。バケット名には、小文字の英文字、数字、ハイフン、およびピリオドを使用することができます。バケット名の先頭と末尾は文字または数値とし、ハイフンまたは別のピリオドの横にピリオドを使用することはできません。

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

バケットの一覧表示

バケットを一覧表示するには、`s3 ls` コマンドを使用します。一般的な使用例をいくつか次に示します。

次のコマンドを実行すると、すべてのバケットが一覧表示されます。

```
$ aws s3 ls
2018-12-11 17:08:50 my-bucket
2018-12-14 14:55:44 my-bucket2
```

次のコマンドは、バケット内のすべてのオブジェクトとフォルダ (S3 で「プレフィックス」として参照) を一覧表示します。

```
$ aws s3 ls s3://bucket-name
                           PRE path/
2018-12-04 19:05:48          3 MyFile1.txt
```

前の出力は、プレフィックス `path/` 下に `MyFile1.txt` という名前の 1 つのファイルがあることを示しています。

コマンドに含めることによって、出力を特定のプレフィックスに絞り込むことができます。次のコマンドを実行すると、*bucket-name*/path 内のオブジェクト (つまり、プレフィックス `path/` でフィルタリングされた、*bucket-name* 内のオブジェクト) が一覧表示されます。

```
$ aws s3 ls s3://bucket-name/path/
2018-12-06 18:59:32          3 MyFile2.txt
```

バケットの削除

バケットを削除するには、`s3 rb` コマンドを使用します。

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

デフォルトでは、オペレーションが成功するにはバケットが空である必要があります。空ではないバケットを削除するには、`--force` オプションを含める必要があります。

次の例は、バケット内のすべてのオブジェクトとサブフォルダを削除してから、バケットを削除します。

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

> **Note**
>
> 以前に削除されたが保持されているオブジェクトを含む、バージョニングされたバケットを使用している場合、このコマンドでバケットを削除することはできません。すべての内容を削除しておく必要があります。

オブジェクトの管理

高レベルの `aws s3` コマンドにより、Amazon S3 オブジェクトの管理が便利になります。オブジェクトコマンドには、`s3 cp`、`s3 ls`、`s3 mv`、`s3 rm`、および `s3 sync` があります。

`cp`、`ls`、`mv`、および `rm` の各コマンドは Unix の対応コマンドと同様に動作するため、ローカルディレクトリと Amazon S3 バケット間でシームレスな作業が可能になります。`sync` コマンドはバケットとディレクトリ、または 2 つのバケットの内容を同期します。

> **Note**
>
> Amazon S3 バケットへのオブジェクトのアップロードに関わるすべての高レベルコマンド (`s3 cp`、`s3 mv`、および `s3 sync`) は、オブジェクトが大きいとき、自動的にマルチパートアップロードを実行します。
> これらのコマンドを使用する場合、失敗したアップロードを再開することはできません。マルチパートアップロードがタイムアウトで失敗するか、Ctrl+C を押して手動でキャンセルされた場合、AWS CLI は作成されたファイルをクリーンアップし、アップロードを中止します。この処理には数分かかることもあります。
> プロセスが kill コマンドまたはシステム障害によって中断された場合、進行中のマルチパートアップロードは Amazon S3 に残り、AWS マネジメントコンソール で、または s3api abort-multipart-upload コマンドを使用して手動でクリーンアップする必要があります。

`cp`、`mv`、および `sync` コマンドには、指定されたユーザーまたはグループにオブジェクトのアクセス許可を付与できる `--grants` オプションが用意されています。次の構文を使用して、`--grants` オプションをアクセス許可のリストに設定します。

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

各値には以下の要素が含まれます。

- *`Permission`* – 付与されたアクセス許可を指定し、`read`、`readacl`、`writeacl`、または `full` に設定できます。
- *`Grantee_Type`* – 被付与者を識別する方法を指定し、`uri`、`emailaddress`、または `id` に設定できます。
- *`Grantee_ID`* – *`Grantee_Type`* に基づいて被付与者を指定します。
  - `uri` – グループの URI。詳細については、「被付与者とは」を参照してください。

- emailaddress – アカウントの E メールアドレス。
- id – アカウントの正規 ID。

Amazon S3 のアクセスコントロールの詳細については、アクセスコントロールに関する説明を参照してください。

次の例では、バケットにオブジェクトをコピーします。これは、オブジェクトの read アクセス権限を全員に付与し、full アクセス権限 (read、readacl、および writeacl) を、user@example.com に関連付けられたアカウントに付与します。

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

Amazon S3 にアップロードするオブジェクトについて、デフォルト以外のストレージクラス (REDUCED_REDUNDANCY または STANDARD_IA) を指定することもできます。そのためには、--storage-class オプションを使用します。

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

s3 sync コマンドでは次の構文を使用します。使用できるソースとターゲットの組み合わせは次のとおりです。

- ローカルファイルシステムから Amazon S3 へ
- Amazon S3 からローカルファイルシステムへ
- Amazon S3 ~ Amazon S3

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

以下の例では、my-bucket の path という Amazon S3 フォルダの内容を、現在の作業ディレクトリと同期させます。s3 sync は、同期先で名前は同じでサイズまたは変更時間が異なるファイルを更新します。出力には、同期中に実行された特定のオペレーションが表示されます。オペレーションでは、サブディレクトリ MySubdirectory とその内容が、s3://my-bucket/path/MySubdirectory と再帰的に同期されます。

```
$ 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
```

通常、s3 sync は見つからないか古いファイルまたはオブジェクトのみを、ソースとターゲット間でコピーします。ただし、--delete オプションを指定して、ソースに存在しないファイルまたはオブジェクトをターゲットから削除することもできます。

前の例を拡張する次の例で、この方法を示します。

```
// 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 および --include オプションを使用して、同期オペレーション中にコピーするファイルまたはオブジェクトをフィルタリングするルールを指定できます。デフォルトでは、指定されたフォルダ内のすべての項目が同期に含まれます。そのため、--include は、--exclude オプションの例外を指定する必要がある場合にのみ必要です (つまり、--include は実質的に「除外しない」を意味します)。これらのオプションは、次の例に示すように、指定された順序で適用されます。

```
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
```

--exclude および --include オプションは、--delete オプションを含む s3 sync オペレーション時に削除されるファイルまたはオブジェクトもフィルタリングします。この場合、パラメータ文字列で、ターゲットディレクトリまたはバケットに関連して、削除から除外するか、削除に含めるファイルを指定する必要があります。例を以下に示します。

```
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
```

s3 sync コマンドは --acl オプションも受け入れ、このオプションによって Amazon S3 にコピーされるファイルのアクセス許可を設定することができます。--acl オプションは、private、public-read、および public-read-write の値を受け入れます。

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

前述したように、s3 コマンドセットには cp、mv、ls、および rm が含まれ、これらは Unix の対応コマンドと同じように動作します。次に例をいくつか示します。

```
// 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 オプションを cp、mv、rm のいずれかとともにディレクトリまたはフォルダで使用する場合、コマンドはすべてのサブディレクトリを含むディレクトリツリーを対象にします。これらのコマンドでは、--exclude コマンドと同じように --include、--acl、および sync オプションも使用できます。

AWS CLI での API レベル (s3api) コマンドの使用

API レベルコマンド (s3api コマンドセットに含まれる) は、Amazon Simple Storage Service (Amazon S3) API への直接アクセスを許可し、高レベルの s3 コマンドで公開されていない一部のオペレーションを有効にします。これらのコマンドは、サービスの機能性への API レベルのアクセスを提供する他の AWS サービスに相当します。

このトピックでは、Amazon S3 API にマッピングする低レベルコマンドの使用方法を示す例について述べます。また、各 S3 API に関する例については、CLI リファレンスガイドの s3api セクションを参照してください。

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

カスタム ACL の適用

高レベルコマンドでは、--acl オプションを使用して、事前定義のアクセスコントロールリスト (ACL) を Amazon S3 オブジェクトに適用できますが、そのコマンドを使用してバケット全体の ACL を設定することはできません。ただし、これは API レベルのコマンド put-bucket-acl で行うことができます。

次の例は、完全なコントロールを 2 人の AWS ユーザー (user1@example.com と user2@example.com) に付与し、読み取りアクセス許可を everyone に付与する方法を示しています。「everyone」の識別子は、パラメータとして渡す特殊な URI から取得されます。

```
$ 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"'
```

ACL の構築方法の詳細については、Amazon Simple Storage Service API Reference の PUT Bucket acl を参照してください。put-bucket-acl など、CLI の s3api ACL コマンドは、同様の引数の略記法を使用します。

ロギングポリシーの設定

API コマンド put-bucket-logging は、バケットロギングポリシーを設定します。

次の例では、AWS ユーザー user@example.com にはログファイルに対する完全なコントロールが付与され、すべてのユーザーがログファイルに対する読み取り許可を持っています。ログの読み取りとバケットへの書き込みに必要なアクセス許可を Amazon S3 のログ配信システム (URI によって指定される) に付与するには、put-bucket-acl コマンドも必要である点に注意してください。

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

前のコマンドのファイル logging.json の内容は、以下のとおりです。

```
{
  "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"
      }
    ]
  }
}
```

AWS CLI での Amazon SNS の使用

AWS Command Line Interface (AWS CLI) を使用して Amazon Simple Notification Service (Amazon SNS) の機能にアクセスできます。Amazon SNS 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws sns help
```

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

このトピックでは、Amazon SNS の一般的なタスクを実行する CLI コマンドの例を示します。

トピック

トピックの作成

トピックを作成するには、`create-topic` コマンドを使用し、トピックに割り当てる名前を指定します。

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

レスポンスの `TopicArn` を書き留めます。これは、後でメッセージを発行するために使用します。

トピックへのサブスクライブ

トピックをサブスクライブするには、`subscribe` コマンドを使用します。

次の例では、`email` プロトコルと `notification-endpoint` の E メールアドレスを指定します。

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

AWS は、`subscribe` コマンドで指定したアドレスに E メールで確認メッセージを送信します。E メールメッセージには、次のようなテキストが含まれています。

```
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
```

受信者が サブスクリプションを確認 リンクをクリックすると、受信者のブラウザに次のような情報を含んだ通知メッセージが表示されます。

```
Subscription confirmed!

You have subscribed saanvi@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-8bb22EXAMPLE

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

トピックへの発行

トピックのすべての受信者にメッセージを送信するには、publish コマンドを使用します。

次の例では、指定されたトピックのすべての受信者に「Hello World!」というメッセージを送信します。

```
$ aws sns publish --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --message "Hello
 World!"
{
    "MessageId": "4e41661d-5eec-5ddf-8dab-2c867EXAMPLE"
}
```

この例では、AWS は、「Hello World!」というテキストを含んだ E メールメッセージを saanvi@example.com に送信します。

トピックからサブスクリプションを解除する

トピックのサブスクライブを解除して、そのトピックに発行されるメッセージの受信を停止するには、unsubscribe コマンドを使用し、サブスクライブを解除するトピックの ARN を指定します。

```
$ aws sns unsubscribe --subscription-arn arn:aws:sns:us-west-2:123456789012:my-
topic:1328f057-de93-4c15-512e-8bb22EXAMPLE
```

サブスクライブが正常に解除されたことを確認するには、list-subscriptions コマンドを使用して、ARN がリストに表示されなくなったことを確認します。

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

トピックの削除

トピックを削除するには、delete-topic コマンドを実行します。

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

AWS がトピックを正常に削除したことを確認するには、list-topics コマンドを使用して、トピックがリストに表示されなくなったことを確認します。

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

AWS CLI での Amazon SWF の使用

AWS Command Line Interface (AWS CLI) を使用して Amazon Simple Workflow Service (Amazon SWF) の機能にアクセスできます。

Amazon SWF 用の AWS CLI コマンドを一覧表示するには、以下のコマンドを使用します。

```
aws swf help
```

コマンドを実行する前に、デフォルトの認証情報を設定します。詳細については、「AWS CLI の設定 (p. 19)」を参照してください。

このトピックでは、Amazon SWF の一般的なタスクを実行する CLI コマンドの例を示します。

トピック

Amazon SWF コマンドのカテゴリ別リスト

AWS Command Line Interface (AWS CLI) を使用して、Amazon Simple Workflow Service (Amazon SWF) でワークフローの作成、表示、および管理を行うことができます。

このセクションでは、AWS CLI の Amazon SWF コマンドのリファレンストピックを機能カテゴリ別に示します。

コマンドのアルファベット順リストについては、AWS CLI Command Reference で Amazon SWF のセクションを参照するか、以下のコマンドを使用してください。

```
$ aws swf help
```

コマンド名の後に `help` ディレクティブを付けることによって、個々のコマンドのヘルプを取得することもできます。例を以下に示します。

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

トピック

アクティビティに関連するコマンド

アクティビティワーカーは `poll-for-activity-task` を使用して新しいアクティビティタスクを取得します。ワーカーは、Amazon SWF からアクティビティタスクを取得した後、それを実行し、成功の場合は `respond-activity-task-completed`、不成功の場合は `respond-activity-task-failed` を使用して応答します。

アクティビティワーカーによって実行されるコマンドを以下に示します。

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

ディサイダーに関連するコマンド

ディサイダーは `poll-for-decision-task` を使用して決定タスクを取得します。ディサイダーは、Amazon SWF から決定タスクを受信した後、そのワークフロー実行履歴を調べ、次の動作を決定します。決定タスクを完了するために `respond-decision-task-completed` が呼び出され、ゼロ以上の次の判断が提供されます。

ディサイダーによって実行されるコマンドを以下に示します。

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

ワークフロー実行に関連するコマンド

次のコマンドはワークフロー実行で動作します。

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

管理に関するコマンド

Amazon SWF コンソールから管理タスクを実行することもできますが、このセクションのコマンドを使用して、関数を自動化したり独自の管理ツールを構築したりできます。

アクティビティ管理

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

ワークフロー管理

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

ドメイン管理

- register-domain
- deprecate-domain

これらのドメイン管理コマンドの詳細と例については、「AWS CLI を使用して Amazon SWF ドメインを操作する (p. 96)」を参照してください。

ワークフロー実行管理

- request-cancel-workflow-execution
- terminate-workflow-execution

可視化コマンド

Amazon SWF コンソールから可視化アクションを実行することもできますが、このセクションのコマンドを使用して、独自のコンソールまたは管理ツールを構築できます。

アクティビティの可視化

- list-activity-types

- describe-activity-type

ワークフローの可視化

- list-workflow-types
- describe-workflow-type

ワークフロー実行の可視化

- describe-workflow-execution
- list-open-workflow-executions
- list-closed-workflow-executions
- count-open-workflow-executions
- count-closed-workflow-executions
- get-workflow-execution-history

ドメインの可視化

- list-domains
- describe-domain

これらのドメイン可視化コマンドの詳細と例については、「AWS CLI を使用して Amazon SWF ドメインを操作する (p. 96)」を参照してください。

タスクリストの可視化

- count-pending-activity-tasks
- count-pending-decision-tasks

AWS CLI を使用して Amazon SWF ドメインを操作する

AWS Command Line Interface (AWS CLI) を使用して、Amazon Simple Workflow Service (Amazon SWF) ドメインを管理することができます。

トピック

ドメインを一覧表示する

AWS アカウントを登録した Amazon SWF ドメインを一覧表示するには、`swf list-domains` を使用することができます。`--registration-status` を含めて、`REGISTERED` または `DEPRECATED` を指定する必要があります。

以下に最小限の例を示します。

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

Note

DEPRECATED の使用例については、「ドメインの廃止 (p. 98)」を参照してください。

ドメインに関する情報を取得する

特定のドメインに関する詳細情報を表示するには、swf describe-domain を使用します。--name という必須パラメータが 1 つあり、これを使用して、情報が必要なドメインの名前を取得します。以下に例を示します。

```
$ aws swf describe-domain --name ExampleDomain
{
    "domainInfo": {
        "status": "REGISTERED",
        "name": "ExampleDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "1"
    }
}
```

ドメインを登録する

新しいドメインを登録するには、swf register-domain を使用します。

2 つの必須パラメータがあります。--name は登録するドメイン名を取り、--workflow-execution-retention-period-in-days はこのドメインにワークフロー実行データを保持する日数を指定する整数を取り、最大 90 日間です (詳細については、Amazon SWF FAQ を参照してください)。

この値にゼロ (0) を指定した場合、保持期間は自動的に最大継続時間に設定されます。それ以外の場合、ワークフロー実行データは指定された日数の経過後は保持されません。次の例は、新しいドメインを登録する方法を示しています。

```
$ aws swf register-domain --name MyNeatNewDomain --workflow-execution-retention-period-in-
days 0
```

このコマンドは出力を返しませんが、swf list-domains または swf describe-domain を使用して新しいドメインを表示することができます。以下に例を示します。

```
$ aws swf describe-domain --name MyNeatNewDomain
{
    "domainInfo": {
```

```
        "status": "REGISTERED",
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
    }
}
```

ドメインの廃止

ドメインを廃止するには (まだ表示できますが、新しいワークフロー実行または登録タイプを作成することはできません)、`swf deprecate-domain` を使用します。これには、唯一の必須パラメータ `--name` があり、廃止するドメイン名を取得します。

```
$ aws swf deprecate-domain --name MyNeatNewDomain
```

`register-domain` と同様に、出力は返されません。ただし、登録したドメインを表示するために `list-domains` を使用する場合、ドメイン間でそのドメインは表示されなくなります。`--registration-status DEPRECATED` を使用することもできます。

```
$ aws swf list-domains --registration-status DEPRECATED
{
    "domainInfos": [
        {
            "status": "DEPRECATED",
            "name": "MyNeatNewDomain"
        }
    ]
}
```

以下の資料も参照してください。

- 『AWS CLI Command Reference』の「deprecate-domain」
- 『AWS CLI Command Reference』の「describe-domain」
- 『AWS CLI Command Reference』の「list-domains」
- 『AWS CLI Command Reference』の「register-domain」

AWS CLI エラーのトラブルシューティング

インストール問題

pip を使用して AWS Command Line Interface (AWS CLI) をインストールする場合、aws プログラムを含んでいるフォルダをオペレーティングシステムの PATH 環境変数に追加するか、モード変更して、実行可能にする必要がある場合があります。

エラー: aws: コマンドが見つかりません

オペレーティングシステムの PATH 環境変数への aws 実行ファイルの追加が必要になる場合があります。

- Windows – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 12)
- macOS – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 15)
- Linux – AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 7)

aws が PATH にあるが、このエラーが表示される場合は、ファイルモードが適切でない可能性があります。直接実行してみてください。

```
$ ~/.local/bin/aws --version
```

アクセス許可問題

メイン CLI プログラムには「run」アクセス許可が必要です。

エラー: アクセス許可が拒否されました

aws プログラムに呼び出し元のユーザーに対する実行アクセス許可があることを確認してください。一般に、755 を使用します。

chmod +x を実行して、ファイルに実行アクセス許可を追加します。

```
$ chmod +x ~/.local/bin/aws
```

有効な認証情報を使用する必要があります。

エラー :AWS は、提供された認証情報を検証できませんでした

AWS CLI は、予期しているのとは異なる場所から認証情報を読み取っている可能性があります。aws configure list を実行して、使用される認証情報を確認することができます。

次の例は、デフォルトのプロファイルに使用される認証情報をチェックする方法を示しています。

```
$ 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
```

次の例は、名前付きプロファイルの認証情報をチェックする方法を示しています。

```
$ aws configure list --profile saanvi
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                    saanvi           manual    --profile
access_key         ********** shared-credentials-file
secret_key         ********** shared-credentials-file
    region                us-west-2      config-file    ~/.aws/config
```

有効な認証情報を使用している場合は、クロックが同期していない可能性があります。Linux, macOS, or Unix で `date` を実行して、時間を確認します。

```
$ date
```

システムクロックのずれが数分以内の場合は、`ntpd` を使用して同期します。

```
$ sudo service ntpd stop
$ sudo ntpdate time.nist.gov
$ sudo service ntpd start
$ ntpstat
```

Windows では、コントロールパネルで日付と時刻オプションを使用してシステムクロックを設定します。

IAM ユーザーは、コマンドを実行できる必要があります。

エラー: *`CreateKeyPair`* オペレーションを呼び出すときにエラーが発生しました (UnauthorizedOperation)。このアクションを実行する権限がありません。

IAM ユーザーまたはロールには、AWS CLI を使用して実行するコマンドに対応する API アクションを呼び出すアクセス許可がありません。

ほとんどのコマンドは、コマンド名と一致する名前を指定して 1 つのアクションを呼び出します。ただし、`aws s3 sync` などのカスタムコマンドは複数の API を呼び出します。`--debug` オプションを使用して、コマンドが呼び出す API を確認できます。

IAM ユーザーおよびロールへのアクセス許可の割り当ての詳細については、IAM ユーザーガイド の 「アクセス管理の概要：アクセス許可とポリシー」 を参照してください。