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 CLI は、AWS のサービスとやり取りするためのコマンドを提供する、AWS SDK for Python (Boto) 上に構築された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から、AWS マネジメントコンソールで提供されるすべて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 – Microsoft Windows で、PowerShell または Windows コマンドプロセッサのどちらかでコマンドを実行します。
- リモートに – PuTTY や SSH などのリモートターミナルを通じて、または Amazon EC2 システムマネージャーを使用して、Amazon EC2 インスタンスでコマンドを実行します。

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します。または、他の言語でのプログラム開発で得た経験を AWSSDK で活用します。

低レベルの同等の API コマンドに加えて、AWS CLI も複数のサービスのカスタマイズを提供します。カスタマイズは、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です。たとえば、`aws s3` の一連のコマンドは、Amazon S3 のファイルを管理するための使い慣れた構文を提供します。

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、ツールキット、または AWS Tools for Windows PowerShell を使用することをお勧めします。

- AWS Tools for Windows 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 Toolkit for Eclipse
- AWS Toolkit for Visual Studio
- 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 (「$」) として表示されます。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**######**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 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、***###########***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、特別な形式なし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。

以下の例に示しているのは、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です。

```
$ 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、より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 アクセスキー ID など) と置き換える必要のある引数 (グループ名など) の両方が***###########***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、例を説明する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 Command Line Interface のインストール

Linux、Windows、macOS での AWS CLI の主なディストリビューション方法は `pip` です。これは、Python パッケージとその依存関係のインストール、アップグレード、削除を容易にする Python の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。

要件

- 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

古いバージョンの Python は、すべての AWS 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、最新バージョンに更新します。

`pip` およびサポートされるバージョンの Python をすでにインストールしてある場合は、次のコマンドで 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。

`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するか、通常使用しているものと異なるバージョンの Python を使用できます。

スタンドアロンインストーラ

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。代わりに、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。

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. 9)

- 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. 14)

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

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。

```
$ 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、使用しているオペレーティングシステムに応じた手順に従ってください。

セクション

- Linux に AWS Command Line Interface をインストールします (p. 5)
- Microsoft Windows で AWS Command Line Interface をインストールする (p. 9)
- macOS で AWS Command Line Interface をインストールする (p. 12)
- 仮想環境で AWS Command Line Interface をインストールする (p. 15)
- バンドルされたインストーラ (Linux, macOS, or Unix) を使用して AWS CLI をインストールする (p. 16)

Linux に AWS Command Line Interface をインストールします

AWS Command Line Interface 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
```

Python 2 バージョン 2.6.5+ または Python 3 バージョン 3.3+ がない場合は、Python をインストール (p. 6)します。それ以外の場合は、pip および AWS CLI をインストールします。

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、付属しているバージョンが古い場合は、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 がインストール済み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を使用して、開発者パッケージ (通常は python-dev または python-devel という名前) 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。

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

Amazon Linux 2017 で AWS Command Line Interface をインストールする

AWS CLI は Amazon Linux AMI にプリインストールされています。`aws --version` で、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/1.11.83 Python/2.7.12 Linux/4.9.20-11.31.amzn1.x86_64 botocore/1.5.46
```

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。最新バージョンを取得するには、pip を使用します。

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

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/1.11.85 Python/2.7.12 Linux/4.9.20-11.31.amzn1.x86_64 botocore/1.5.48
   ```

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(user) で AWS CLI をアップグレードするには

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/1.11.85 Python/2.7.12 Linux/4.9.20-11.31.amzn1.x86_64 botocore/1.5.48
   ```

セクション

pip のインストール

pip がない場合は、Python Packaging Authority が提供するスクリプトを使用して pip をインストールします。

pip をインストールするには

1. pypa.io から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
   ```

このスクリプトは、最新バージョンの pip と、もう 1 つの必要なパッケージ (setuptools) をダウンロードしてインストールします。

2. Python を使用してスクリプトを実行します。

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

3. PATH 変数に実行可能パスを追加します。~/.local/bin

   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わからない場合は、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
      ```

4. pip が正しくインストールされたことを確認します。

```
$ pip --version
pip 8.1.2 from ~/.local/lib/python3.4/site-packages (python 3.4)
```

pip を使用した AWS CLI のインストール

pip を使用して AWS CLI をインストールします。

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

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

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. 93)」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 のインストール後は、OS 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。

Example AWS CLI のインストール場所 - Linux と pip (ユーザーモード)

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

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 インストール環境の bin フォルダにある可能性があります。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
~/.local/Python/3.6/bin/python3.6
```

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わからない場合は、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
   ```

Microsoft Windows で AWS Command Line Interface をインストールする

Windows で 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

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

   > **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 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 インストーラを実行します。
3. 表示される手順に従います。

CLI はデフォルトでは `C:\Program Files\Amazon\AWSCLI`(64 ビット) または `C:\Program Files (x86)\Amazon\AWSCLI`(32 ビット) へ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がインストールされているかどうか不明な場合は、スタートメニューを開き「cmd」を検索します)。

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

コマンドを入力するときは、プロンプト記号 (上記の「>」) 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 が実行ファイルを見つけることができない場合、コマンドプロンプトを再度開くか、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)します。

アップグレードされた MSI インストーラが利用可能

アップグレードされた MSI インストーラを実行できます。これは Python 2 ではなく Python 3 を使用します。

- 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 ビットおよび 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)

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 のリリースページを確認して、最新バージョンがいつリリースされたかを確認します。最新バージョンに更新するには、上述のようにもう一度 MSI インストーラをダウンロードして実行します。

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 コマンドラインインターフェイス] という名前のエントリを選択し、[アンインストール] をクリック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。

次のコマンドでコマンドラインから [プログラムと機能] メニューを起動することもできます。

```
> appwiz.cpl
```

Windows で Python、pip、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 3.6 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 3.6 Windows x86-64 実行可能ファイルのインストーラー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 3.6 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、実行ディレクトリをユーザーパスに追加します。

pip で 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 Windows コマンドプロセッサを開きます。
2. 次のコマンドを使用して、Python と pip の両方が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 3.6.2
C:\Windows\System32> pip --version
pip 9.0.1 from c:\users\myname\appdata\local\programs\python\python36\lib\site-packages
 (python 3.6)
```

3. `pip` を使用して AWS CLI をインストールします。

```
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/1.11.84 Python/3.6.2 Windows/7 botocore/1.5.47
```

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 のインストール後は、OS の aws 環境変数に PATH 実行ファイルを追加します。MSI のインストールでは、これは自動的に行われますが、aws コマンドが動作しない場合は手動設定が必要な場合があります。

- Python 3.6 と pip – `%USERPROFILE%\AppData\Local\Programs\Python\Python36\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`

PATH 変数を変更するには (Windows)

1. Windows キーを押し、「**####**」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、[Edit] を選択します。
4. セミコロンで区切って、[Variable value] フィールドにパスを追加します。以下に例を示します。*C:\existing\path*;***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、もう一度開きます。

macOS で AWS Command Line Interface をインストールする

macOS で AWS CLI をインストールするための推奨の方法は、バンドルインストーラの使用です。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

- AWS CLI 実行ファイルをコマンドラインパスに追加する (p. 14)

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 Command Line Interface をインストールします (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 Bundled Installer 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。以下に例を示します。

   ```
   $ sudo /usr/local/bin/python2.7 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。こうすると、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 を呼び出すことができます。

`-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 が提供するスクリプトを使用して、`pip` をインストールします。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   $ python3 get-pip.py --user
   ```
3. `pip` を使用して AWS CLI をインストールします。

   ```
   $ pip3 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 1.11.84 (Python 3.6.1)
   ```

   実行可能ファイルが見つからない場合は、そのファイルをコマンドラインパスに追加 (p. 14)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。

```
$ pip3 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` のインストール後は、OS 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。実行可能ファイルの場所は、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 3.6 および pip のある macOS (ユーザーモード)

```
~/Library/Python/3.6/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
~/Library/Python/3.6/bin/python3.6
```

`pip` は、Python 実行可能ファイルが含まれているのと同じフォルダに、実行可能ファイル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わからない場合は、`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
   ```

仮想環境で AWS Command Line Interface をインストールする

仮想環境に 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 で `virtualenv` をインストールします。

   ```
   $ pip install --user virtualenv
   ```
2. 仮想環境を作成します。

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

   デフォルト以外の Python 実行可能ファイルを使用するには、`-p` オプションを指定します。

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

3. 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. 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/1.11.84 Python/3.6.2 Linux/4.4.0-59-generic botocore/1.5.47
   ```

`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、アクティベーションコマンドを再度実行します。

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。

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

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

Linux, macOS, or Unix では、バンドルされているインストーラを使用して AWS CLI をインストールすることもできます。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 Command Line Interface をインストールします (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 Bundled Installer 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。以下に例を示します。

   ```
   $ sudo /usr/local/bin/python2.7 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。こうすると、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 を呼び出すことができます。

`-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 権限がない場合、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、上記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ために、`~/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。

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やデフォルトリージョンなど、AWS Command Line Interface が AWS とやり取りする際に使用する設定を指定する方法を説明します。

> Note
>
> AWS CLI がユーザーに代わってリクエストに署名し、署名に日付を含めます。コンピュータの日時が正しく設定されていることを確認します。正しくない場合は、署名の日付がリクエストの日付と一致しないことがあり、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

クイック設定

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
```

AWS CLI によって、4 種類の情報の入力が求められます。AWS アクセスキー ID と AWS シークレットアクセスキーはアカウントの認証情報です。

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)

デフォルトのリージョンは、デフォルトで呼び出しを実行する対象のリージョンの名前です。これは、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、米国西部 (オレゴン) を使用するには、「`us-west-2`」と入力します。

> Note
>
> AWS CLI を使用するときは AWS リージョンを指定する必要があります。サービスと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。

デフォルトの出力形式は `json`、`text`、`table` のいずれかです。出力形式を指定しない場合、`json` が使用されます。

複数のプロファイルがある場合は、`--profile` オプションを使用して追加の名前付きプロファイルを設定できます。

```
$ 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
```

いずれかの設定を更新するには、`aws configure` を再び実行し、必要に応じて新しい値を入力します。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、プロバイダーチェーンを使用して、システムまたはユーザー環境変数、ローカル AWS 設定ファイルなど、さまざまな場所で AWS の認証情報が検索されます。

AWS CLI では、以下の順序で認証情報と構成設定が検索されます。

1. コマンドラインオプション – デフォルト設定より優先されるようにコマンドオプションでリージョン、出力形式、プロファイルを指定できます。
2. 環境変数 (p. 23) – `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、および `AWS_SESSION_TOKEN`。
3. AWS 認証情報ファイル – Linux, macOS, or Unix の `~/.aws/credentials`、または Windows の `C:\Users\USERNAME \.aws\credentials` にあります。このファイルには、デフォルトのプロファイルに加えて、複数の名前付きプロファイルを含めることができます。
4. CLI 設定ファイル – 一般的に、Linux, macOS, or Unix では `~/.aws/config` に、Windows では `C:\Users\USERNAME \.aws\config` にあります。このファイルには、デフォルトのプロファイル、名前付きプロファイル、 CLI 固有の設定パラメータを含めることができます。
5. 認証情報 – タスクにロールを割り当てるときに、コンテナインスタンスで Amazon Elastic Container Service によって提供されます。
6. インスタンスプロファイルの認証情報 - これらの認証情報は、インスタンスロールが割り当てられた EC2 インスタンスで使用でき、Amazon EC2 メタデータサービスを介して提供されます。

設定ファイルと認証情報ファイル

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。ホームディレクトリの場所は異なりますが、Windows では `%UserProfile%`、また Unix 系のシステムでは `$HOME` または `~` (チルダ) といった環境変数を使用して参照できます。

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

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

認証情報を機密性の低いオプションから分離するために、リージョンおよび出力形式は同じフォルダの `config` という名前の別個のファイルに保存されます。

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 ファイルのデフォルトの設定ファイルの場所より優先させることができます。詳細については、「Environment Variables (p. 23)」を参照してください。

認証情報を Config 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ります。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
```

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 – AWS リージョン。

output – 出力形式 (json、text、table)

名前付きプロファイル

AWS CLI は config と認証情報ファイルに保存された 名前付きプロファイルをサポートしています。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、または、config と認証情報ファイルへエントリを追加します。

2 つのプロファイルのある認証情報ファイルの例を以下に示します。

~/.aws/credentials

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

[user2]
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 (2 人の異なる IAM ユーザーのもの) を使用します。別の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 user2]
region=us-east-1
output=text
```

> **Important**
>
> AWS 認証情報ファイルは、CLI 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。AWS 認証情報ファイルで名前付きプロファイルを設定するときは、「profile」プレフィックスは含めないでください。

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` オプションを追加します。次の例では、前のセクションの `user2` プロファイルを使用して実行中の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場合、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

```
> 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。変数の詳細は次のセクションを参照してください。

Environment Variables

環境変数は設定と認証情報ファイルより優先されます。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。

AWS CLI は次の環境変数をサポートしています。

- `AWS_ACCESS_KEY_ID` – AWS アクセスキー。
- `AWS_SECRET_ACCESS_KEY` – AWS シークレットキー. アクセスキーおよびシークレットキーの変数は認証情報ファイルおよび config ファイルに保存されている認証情報よりも優先されます。
- `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、セッショントークンを指定します。
- `AWS_DEFAULT_REGION` – AWS リージョン. この変数が設定してあると、使用中のプロファイルのデフォルトリージョンよりも優先されます。
- `AWS_DEFAULT_OUTPUT` – AWS CLI の出力形式を、`json`、`text`、または `table` に変更します。
- `AWS_PROFILE` – 使用する CLI プロファイルの名前。これは、認証情報ファイルまたは 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 `default` となります。
- `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 が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。
- `AWS_CONFIG_FILE` – AWS CLI が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。

次の例では、このガイドで前述した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。

Linux, macOS, or Unix

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

```
$ export AWS_DEFAULT_REGION=us-west-2
```

Windows

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

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。コマンドラインオプションを使用して認証情報を指定することはできません。

--profile

使用する名前付きプロファイル (p. 22)の名前。

--region

呼び出す AWS リージョン。

--出力

出力形式。

--endpoint-url

呼び出しの対象となる URL。ほとんどのコマンドでは、AWS CLI により、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。

これらのいずれかのオプションをコマンドラインで指定すると、デフォルト設定および 1 つのコマンドに対応する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。引数文字列にスペースが含まれている場合は、引数を引用符で囲みます。

Tip

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 を使用します。

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

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わからない場合は、次に示すように、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. 40)」で詳細に説明されています。

インスタンスメタデータ

EC2 インスタンスから CLI を使用するには、必要なリソースへのアクセス権限を持つロールを作成し、そのロールを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、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します。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、認証情報を入力せずに `aws configure` を使用してリージョンを設定できます。

```
$ 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
```

AWS CLI はインスタンスのメタデータから認証情報を読み取ります。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 にアクセスする必要がある場合は、`HTTP_PROXY` および `HTTPS_PROXY` 環境変数をプロキシサーバーの IP アドレスを使用して設定する必要があります。

Linux, macOS, or Unix

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

Windows

```
> set HTTP_PROXY=http://a.b.c.d:n
> 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。プロキシ URL にユーザー名とパスワードを次のように指定します。

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

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

> Note
>
> AWS CLI は NTLM プロキシをサポートしていません。NTLM または Kerberos 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。

EC2 インスタンスでのプロキシの使用

IAM ロールで起動した EC2 インスタンスにプロキシを設定する場合は、IP アドレス 169.254.169.254 で `NO_PROXY` 環境変数を設定し、AWS CLI がインスタンスメタデータにアクセスできるようにする必要があります。

Linux, macOS, or Unix

```
$ export NO_PROXY=169.254.169.254
```

Windows

```
> set NO_PROXY=169.254.169.254
```

ロールを割り当てる

IAM ロール は、追加のアクセス権限や、別のアカウントでアクションを実行するためのアクセス権限をユーザーが取得できるようにする認証ツールです。

`~/.aws/config` ファイルで、ロールのためのプロファイルを作成して、ロールを使用するために AWS Command Line Interface を設定できます。次の例では、デフォルトプロファイルで引き受ける `marketingadmin` という名前のロールプロファイルを示します。

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

この場合、デフォルトのプロファイルは、marketingadmin という名前のロールを引き受ける認証情報とアクセス権限を持つ IAM ユーザーです。ロールにアクセスするには、名前付きプロファイルを作成します。このプロファイルを認証情報を使用して設定する代わりに、ロールの ARN と、アクセス権限を持つプロファイルの名前を指定します。

セクション

ロールの設定と使用

ロール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 を呼び出し、指定したロールを引き受けます。ソースプロファイルはロールに対して `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、ロールには引き受けさせるソースプロファイルとの信頼関係がある必要があります。

『AWS Identity and Access Management ユーザーガイド』の「IAM ユーザーにアクセス権限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します。ロールとターゲット IAM ユーザーが同じアカウント内に存在する場合、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。以下の例は、`jonsmith` という IAM ユーザーによって引き受けられることをロールに許可する信頼関係を示します。

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

次に、IAM ユーザーにロールを引き受けるアクセス許可を付与します。以下の例は、IAM ユーザーが `marketingadmin` ロールを引き受けることを許可する AWS Identity and Access Management ポリシーを示します。

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

ユーザーには、ロールプロファイルを使用してコマンドを実行するのに、追加のアクセス権限は必要ありません。ユーザーが、ロールを使用しないで AWS リソースにアクセスできるようにするには、それらのリソースに対する追加のインラインポリシーまたは管理ポリシーを適用します。

たとえば、ロールプロファイル、ロールのアクセス権限、信頼関係およびユーザーアクセス許可が適用されると、コマンドラインで `profile` オプションを使用して、ロールを引き受けることができます。

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

複数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。

Linux, macOS, or Unix

```
$ export AWS_PROFILE=marketingadmin
```

Windows

```
> 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、『AWS Identity and Access Management User Guide』の「ユーザーとグループ」および「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しようとする場合に、多要素認証デバイスから生成された一回限りのキー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、ロールで信頼関係を変更して多要素認証を要求することができます。

```
{
  "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/jonsmith
```

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。

クロスアカウントのロール

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。ロールの作成中に、ロールタイプを [Role for Cross-Account Access] のいずれかのオプションに設定し、オプションで [Require MFA] を選択します。[Require MFA] オプションは、「多要素認証を使用する (p. 28)」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 パラメータを追加します。

```
[profile crossaccountrole]
role_arn = arn:aws:iam::234567890123:role/xaccount
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/jonsmith
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

```
> 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。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。

コマンド補完の設定には 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 での補完

コマンド補完は、Amazon Linux を実行しているインスタンスにデフォルトで設定されます。

セクション

シェルを識別する

使用しているシェルがわからない場合は、次のいずれかのコマンドを使用して識別します。

echo $SHELL – シェルのインストールディレクトリが表示されます。これは通常、ログイン後に別のシェルを起動しない限り、使用中のシェルと一致します。

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

ps – 現在のユーザーの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

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 などのプログラムは、通常、標準のパスの場所に AWS コンプリータ (またはシンボリックリンク) をインストールします。この場合、`which` がコンプリータを見つけます。

```
$ which aws_completer
/usr/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。

- bash – 組み込みのコマンド `complete` を使用します。

  ```
  $ complete -C '/usr/local/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` を使用します。

```
% source /usr/local/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` の一番上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、タブ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
```

AWS Command Line Interface を使用して Amazon EC2 で開発環境をデプロイする

このチュートリアルでは、AWS CLI を使用して Amazon EC2 で開発環境を設定する方法を詳しく説明します。これには、インストールおよび設定の手順のショートバージョンが含まれており、Windows、Linux, macOS, or Unix で起動から終了まで実行できます。

ステップ

AWS CLI のインストール

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 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実行します。

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

2. `pip` を使用して AWS CLI をインストールします

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

AWS CLI を設定する

コマンドラインで `aws configure` を実行して、認証情報と設定をセットアップします。

```
$ 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 が以下の情報のプロンプトを表示します。

- AWS アクセスキー ID と AWS シークレットアクセスキー – アカウントの認証情報です。キーがない場合は、「セキュリティ認証情報の入手方法」(『アマゾン ウェブ サービス全般のリファレンス』) を参照してください。
- デフォルトのリージョン名 – デフォルトで呼び出しを実行する対象のリージョンの名前です。
- デフォルトの出力形式 – 形式は json、text、table のいずれかです。出力形式を指定しない場合、json が使用されます。

コマンドを実行して、認証情報が正しく設定されていること、および 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        ||
|+-----------------------------------+-------------------+|
```

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

次のステップは、SSH を使用してアクセスできる 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することです。Amazon EC2 の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。

セキュリティグループ、キーペア、ロールを作成するには

1. 最初に、新しいセキュリティグループを作成し、SSH でポート 22 を通過する着信トラフィックを許可するルールを追加します。リージョンのデフォルトの 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。セキュリティを強化するため、`0.0.0.0/0`CIDR 範囲を、インスタンスに接続するネットワークの範囲に置き換えてください。

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

   セキュリティグループ ID を書き留めます。これは後でインスタンスを起動するときに使用します。
2. 次に、キーペアを作成し、インスタンスに接続できるようにします。このコマンドでは、キーの内容が `devenv-key.pem` という名前のファイルに保存されます。

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

   Windows

   Windows コマンドプロンプトでは、一重引用符の代わりに二重引用符を使用します。
3. Linux では、ファイルモードを変更して、自分以外がキーファイルにアクセスできないようにする必要があります。

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

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そこに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 Machine Image (AMI) を指定します。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。

```
$ 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. インスタンスの起動にはしばらくかかることがあります。インスタンスの稼働後、接続にはインスタンスのパブリック IP アドレスが必要になります。パブリック IP アドレスを取得するには、次のコマンドを使用します。

```
$ 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。
- [Instance] – 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] パネルで、[Connection]、[SSH]、[Auth] の順に移動し、[Browse] をクリックして `.ppk` ファイルを選択してから、[Open] をクリックして接続します。

4. ターミナルにサーバーのパブリックキーを受け入れるプロンプトが表示されます。「`yes`」と入力して [Enter] をクリックし、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 Command Line Interface の使用

このセクションでは、AWS Command Line Interface 全体で使用される一般的な機能と呼び出しパターンについて説明します。

> **Note**
>
> AWS CLI では、HTTPS 経由でサービスに API コールを実行します。コールを実行するために、TCP ポート 443 でのアウトバウンド接続が有効になっている必要があります。

トピック

AWS Command Line Interface のヘルプ

AWS CLI を使用する際にヘルプを表示するには、コマンドの末尾に `help` を追加するだけです。たとえば、次のコマンドでは一般的な AWS CLI オプションのヘルプおよび使用可能な最上位レベルのコマンドが一覧表示されます。

```
$ aws help
```

次のコマンドを実行すると、Amazon EC2 で使用できるサブコマンドが一覧表示されます。

```
$ aws ec2 help
```

次の例では、入力パラメータ、フィルタ、および出力の説明を含む、EC2 `DescribeInstances` オペレーションの詳細なヘルプが一覧表示されます。コマンドの記述方法がわからない場合は、ヘルプの例のセクションを確認してください。

```
$ 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。

Name – コマンドの名前です。

```
NAME
       describe-instances -
```

Description – コマンドが呼び出す API オペレーションの説明です。コマンドのサービスの API ドキュメントから取得されます。

```
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.
...
```

Synopsis – コマンドとそのオプションのリストが表示されます。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]
```

`describe-instances` のデフォルトの動作では、現在のアカウントおよびリージョンのすべてのインスタンスを記述します。オプションで `instance-ids` のリストを指定して、1 つまたは複数のインスタンスを記述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には、`OPTIONS` セクションで説明されるか、例で表示されます。

Options – Synopsis で表示される各オプションの説明です。

```
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です。必要なコマンドまたはユースケースの例がない場合は、このページまたはそのコマンドのヘルプページの AWS CLI コマンドリファレンスのフィードバックリンクを使用して例を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 a Owner tag

    Command:

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

Output – AWS からのレスポンスで返される各フィールドとデータタイプの説明です。

`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、関連する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 で使用されている API documentation for the reservation datatype から取得されています。

```
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.
```

```
...
```

AWS CLI によって出力が JSON にレンダリングされると、次のように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にパイプで `more` とつなげると、ヘルプファイルを 1 度に 1 ページずつ表示します。ドキュメントをさらに表示するにはスペースバーまたは Page Down を押します。q で終了します。

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

AWS CLI ドキュメント

AWS CLI Command Reference はすべての AWS CLI コマンドのヘルプファイルの内容を表示します。モバイル、タブレット、デスクトップ画面で移動や表示がしやすいように、コンパイルされオンラインで表示されます。

ヘルプファイルには、コマンドラインビューから表示またはジャンプできないリンクが含まれている場合があります。これらはオンライン AWS CLI リファレンスにあります。

API ドキュメント

AWS CLI のすべてのサブコマンドは、サービスのパブリック API に対する呼び出しに対応しています。パブリック API を使用する各サービスには、逆に、AWS ドキュメントウェブサイトのそのサービスのホームページで参照できる API リファレンスドキュメント一式があります。

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 でサポートされているアクション、サービスとやり取りするデータ、および考えられるエラー条件の詳細情報が含まれています。

API ドキュメントセクション

- Actions – パラメータ (長さや内容の制約含む) およびアクション固有のエラーの詳細情報です。Actions は AWS CLI のサブコマンドに対応しています。

- Data Types – サブコマンドによって返されるオブジェクトデータに関する追加情報が含まれていることがあります。
- Common Parameters – サービスのアクションすべてで使用されるパラメータに関する詳細情報です。
- Common Errors – サービスのアクションすべてで返されるエラーに関する詳細情報です。

各セクションの名前や使用可能かについては、サービスによって異なる可能性があります。

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単一の AWS CLI が作成される前からある個別の 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 Command Line Interface のコマンド構造

AWS CLI は、コマンドラインでマルチパート構造を使用します。`aws` へのベースコールから開始されます。次のパートは、最上位コマンドを指定しますが、これは多くの場合 AWS CLI でサポートされている AWS サービスを表しています。各 AWS サービスには実行するオペレーションを指定する追加のサブコマンドがあります。一般的な CLI オプション、またはオペレーション固有のパラメータは、コマンドラインで、任意の順序で指定できます。排他的パラメータが複数回指定された場合、最後の値のみが適用されます。

```
$ 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。

AWS Command Line Interface のパラメータ値の指定

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。次の例に示すとおり、Linux, macOS, or Unix および Windows PowerShell では一重引用符 (') を使用し、Windows コマンドプロンプトでは二重引用符 (") を使用します。

Windows PowerShell、Linux, macOS, or Unix

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

Windows コマンドプロセッサ

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

スペースの代わりに等号も使用できます。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、サービスが準拠することを想定しているいくつかの一般的なパラメータタイプとその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フォーマットに問題がある場合、コマンド名の後に **help** と入力してマニュアルを確認します。次に例を示します。

```
$ 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 – 文字列パラメータには、ASCII 文字セットで英数字、記号、および空白文字が含まれます。空白文字を含む文字列は引用符で囲んである必要があります。標準の空白文字以外の記号と空白文字の使用は推奨されておらず、AWS CLI の使用中に問題が発生する可能性があります。

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. 45)」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 標準形式に基づいています。これらは、「DateTime」または「Date」タイプの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-DDThh:mm:ss.sssTZD (UTC)、例:2014-10-01T20:30:00.000Z
- YYYY-MM-DDThh:mm:ss.sssTZD (オフセットあり)、例:2014-10-01T12:30:00.000-08:00
- YYYY-MM-DD、例:2014-10-01
- Unix 時間 (秒)、例:1412195400

リスト – スペースで区切られた 1 つ以上の文字列です。

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

ブール型 – 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に対してコマンドを検証します。

```
$ 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、dry-run ではないバージョンのパラメータも含まれ、コマンドを通常通り実行するように明示的に指定します。ただし、これはデフォルトの動作であるため、必ずしも含める必要はありません。

整数 – 符号なしの整数です。

```
$ 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、バイナリデータが格納されているローカルファイルへのパスを取ります。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。

`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 または短縮構文 (p. 55)で指定された一連のキーと値のペアです。次の例では、マップパラメータ `--key` で my-table という名前の 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"
        }
    }
}
```

次のセクションでは、JSON 引数について詳しく説明します。

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 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"
```

次の例では、JSON 配列で同等のフィルタのリストを指定します。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 (「Name」と「Values」は両方ともこのインスタンスのキー)。

「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"]
  }
]
```

一方、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。上記のコマンドの 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、データが JSON 形式である必要があります。たとえば、`ec2 run-instances` コマンドの `--block-device-mappings` 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 Elastic Block Store 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、次の例のように配列内のオブジェクトをリストにします。

```
[
  {
    "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 を入力することもできますし (文字列の引用 (p. 44) を参照)、または、ファイルに保存しコマンドラインから参照することもできます (ファイルからパラメータをロードする (p. 44) 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、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。この手法は次のセクションで説明されます。

JSON の詳細については、「Wikipedia-JSON」および「RFC4627 - application/json JSON のメディアタイプ」を参照してください。

文字列の引用

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, or Unix と Windows PowerShell では、以下の例のように一重引用符 (') を使用して JSON データ構造を囲みます。

```
$ aws ec2 run-instances --image-id ami-05355a6c --block-device-mappings '[{"DeviceName":"/
dev/sdb","Ebs":{"VolumeSize":20,"DeleteOnTermination":false,"VolumeType":"standard"}}]'
```

一方、Windows コマンドプロンプトでは、二重引用符 (") を使用して JSON データ構造を囲みます。さらに、次の例のように、JSON データ構造自体の中にある各二重引用符 (") にはバックスラッシュ (\) のエスケープ文字が必要です。

```
> aws ec2 run-instances --image-id ami-05355a6c --block-device-mappings "[{\"DeviceName
\":\"/dev/sdb\",\"Ebs\":{\"VolumeSize\":20,\"DeleteOnTermination\":false,\"VolumeType\":
\"standard\"}}]"
```

Windows PowerShell では、次の例に示すように、JSON データ構造を囲む一重引用符 (')、および JSON 構造内で各二重引用符をエスケープするバックスラッシュ (\) が必要です。

```
> aws ec2 run-instances --image-id ami-05355a6c --block-device-mappings '[{\"DeviceName
\":\"/dev/sdb\",\"Ebs\":{\"VolumeSize\":20,\"DeleteOnTermination\":false,\"VolumeType\":
\"standard\"}}]'
```

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、`aws sqs create-queue` の `attribute` パラメータが `RedrivePolicy` キーであることがあります。`RedrivePolicy` の値は JSON ドキュメントで、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ーをロードするには、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。

Linux, macOS, or Unix

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

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

Windows

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

`file://` プレフィックスのオプションは、'~/', ' を含め Unix 形式の拡張をサポートしています。/', および '../'. Windows では、'~/' 式で %USERPROFILE% 環境変数に保存されているユーザーディレクトリへ拡

張します。たとえば、Windows 7 では、一般的に、ユーザーディレクトリは `C:\Users\User Name\` です。

パラメータキーの値として提供されている JSON ドキュメントを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ー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 `http://` または `https://` の URL でインターネット上にホストされているファイルからのパラメーターのロードもサポートしています。以下の例では、Amazon S3 バケットにあ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、ファイルはパブリックアクセス可能な場所に保存されている必要があります。

```
$ aws ec2 run-instances --image-id ami-a13d6891 --block-device-mappings http://my-
bucket.s3.amazonaws.com/filename.json
```

上記の例で、`filename.json` ファイルには以下の JSON データが含まれます。

```
[
  {
    "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 ユーザー用の IAM ポリシーを設定する (p. 76)」を参照してください。

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

ほとんどの AWS CLI コマンドは、JSON でパラメーターを保存し、コマンドラインで入力する代わりにファイルから読み取ることができる、`--generate-cli-skeleton` と `--cli-input-json` パラメーターをサポートしています。

CLI Skeleton を生成すると、オペレーションに指定できるすべてのパラメーターの概要を示す JSON が出力されます。

--generate-cli-skeleton を aws ec2 run-instances で使用するには

1. `--generate-cli-skeleton` オプションを指定して run-instances コマンドを実行し、JSON スケルトンを表示します。

```
$ 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
}
```

2. 出力先としてファイルを指定し、スケルトンをローカルに保存します。

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

3. スケルトンをテキストエディタで開き、使用しないパラメーターは削除します。

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

`DryRun` パラメーターを true に設定したままにして、EC2 のリハーサル機能を使用します。これにより、リソースを作成することなく設定をテストできます。

4. デフォルトリージョンの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 AMI の値を入力します。この例では、`ami-dfc39aef` は `us-west-2` リージョンの 64 ビット Amazon Linux イメージです。

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

5. `file://` プレフィックスを使用して JSON 設定を `--cli-input-json` パラメーターに渡します。

```
$ 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、パラメーター値が有効であることを示します。出力でその他の問題が報告された場合は、それを修正し、リハーサルのエラーが表示されるまで上記のステップを繰り返します。

6. DryRun パラメーターを false に設定して、リハーサル機能を無効にします。

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

7. 再度 `run-instances` コマンドを実行してインスタンスを起動します。

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

AWS Command Line Interface からのコマンド出力の制御

このセクションでは、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 (テキスト)
- ASCII 形式のテーブル (テーブル)

「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。

- 設定ファイル内の `output` オプションを使用する。次の例では、出力を `text` に設定します。

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

- `AWS_DEFAULT_OUTPUT` 環境変数を使用する。以下に例を示します。

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

- コマンドラインの `--output` オプションを使用する。以下に例を示します。

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

> **Note**
>
> 出力形式が複数の方法で指定されている場合、通常の AWS CLI の優先順位ルール (p. 20)が適用されます。たとえば、`AWS_DEFAULT_OUTPUT` 環境変数を使用すると、`output` を指定して設定ファイルに設定された値がオーバーライドされ、`--output` を指定して AWS CLI コマンドに渡された値は、環境で設定された値または設定ファイルの値をオーバーライドします。

JSON 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) を介して出力をプログラムで処理するのに最適です。テーブル形式は人間にとって読みやすく、テキスト形式は従来の Unix テキスト処理ツール (`sed`、`grep`、`awk` など) や、Windows PowerShell スクリプトとの連携に優れます。

--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 では、`--query` オプションにより組み込みの出力フィルタリング機能を使用できます。この機能を示すため、最初に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 EBS (Elastic Block Storage) ボリュームが個別の 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
        }
    ]
}
```

最初に、次のコマンドで、Volumes リストの最初のボリュームのみを表示できます。

```
$ 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
}
```

ここで、ワイルドカード表記 [*] を使用してリスト全体を反復処理し、3 つの要素 VolumeId、AvailabilityZone、および Size をフィルタリングします。ディクショナリ表記では、{Alias1:Key1,Alias2:Key2} のように、各キーの 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序が設定されていないため、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 など連鎖キーを使用して、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 InstanceId に対して Attachments[0].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] で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、すべてのフィルタリングされた属性は、型にかかわらず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リングするには、JMESPath "?" operator. 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。

`--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。

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。出力例とともに前のトピックに示したように、`--query` オプションは AWS CLI の JSON 形式の出力をフィルタリングおよびフォーマットするための強力な方法を備えています。`--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 の出力をタブ区切りの行に整理します。これは、従来の Unix テキストツール (`grep`、`sed`、`awk` など) や、Windows PowerShell との連携に優れています。

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
```

一貫した動作を確保するため、テキスト出力は *`--query`* オプションとともに使用することを強くお勧めします。これは、テキスト形式により出力列がアルファベット順になり、類似したリソースにキーの同じコレクションがあるとは限らないためです。たとえば、Linux EC2 インスタンスの JSON 表現には、Windows インスタンスの JSON 表現にはない要素があり、その逆も同様です。また、リソースでは今後の更新でキーと値の要素が追加または削除され、列の順序が変更される可能性があります。このような場合に、`--query` はテキスト出力の機能を補強し、出力形式を完全に制御できるようにします。次の例では、コマンドで、表示する要素をあらかじめ選択し、リスト表記 `[key1, key2, ...]` で列の順序を定義します。これにより、ユーザーは予期される列で常に正しいキー値が表示されることを完全に信頼できます。最後に、AWS CLI が、存在しないキーの値として 'None' をどのように出力するかを確認してください。

```
$ 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
```

以下の例は、grep および awk の使用方法を、aws ec2 describe-instances コマンドのテキスト出力とともに示しています。最初のコマンドでは、各インスタンスのアベイラビリティーゾーン、状態、およびインスタンス ID がテキスト出力で表示されます。2 番目のコマンドでは、us-west-2a アベイラビリティーゾーンで実行されているすべてのインスタンスのインスタンス ID のみが出力されます。

```
$ 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
```

次のコマンドでは、停止しているすべてのインスタンスの同様の例を示します。さらに一歩進んで、停止している各インスタンスのインスタンスタイプの変更を自動化します。

```
$ 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 Windows PowerShell でも役立ちます。AWS CLI のテキスト出力はタブ区切りであるため、`t 区切り記号を使用して、PowerShell で簡単に配列に分割されます。次のコマンドでは、最初の列 (AvailabilityZone) が us-west-2a と一致する場合に、3 列目の値 (InstanceId) が表示されます。

```
> 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
```

テーブルの出力形式

table 形式では、人間が読み取れる表現の AWS CLI 出力が作成されます。例を示します。

```
$ 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 オプションはテーブル形式で使用して、raw 出力から事前に選択された要素のセットを表示できます。ディクショナリとリスト表記の出力の違いに注意してください。最初の例では列名の順序はアルファベット順になり、2 番目の例では名前のない列はユーザーの定義どおり順序が設定されます。

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

$ aws ec2 describe-volumes --query 'Volumes[*].
[VolumeId,Attachments[0].InstanceId,AvailabilityZone,Size]' --output table
-----------------------------------------------------
|                 DescribeVolumes                   |
+--------------+--------------+--------------+-----+
```

```
|  vol-e11a5288|  i-a071c394  |  us-west-2a  |  30 |
|  vol-2e410a47|  i-4b41a37c  |  us-west-2a  |  8  |
+--------------+--------------+--------------+-----+
```

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

AWS Command Line Interface は、JSON 形式の非スカラーオプションパラメータを取得できますが、コマンドラインで大規模な JSON リストまたは JSON 構造を入力するのに手間がかかる場合があります。この問題に対処するため、AWS CLI では、完全な JSON 形式の使用ではなく、オプションパラメータをより簡単に表現できる短縮構文をサポートしています。

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
```

Windows PowerShell

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

これは、JSON 形式の次の例と同じになっています。

```
--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 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リストは JSON および省略形の 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。Amazon EC2 用の `stop-instances` コマンドの例を次に示します。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) は省略表現で指定されます。

```
$ 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` コマンドで、これは `--tags` オプションのネストされていない構造のリストを取得します。`--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 Command Line Interface 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では、AWS CLI に追加された 3 つのオプションを使用して、リストを生成するためにサービスの API を呼び出す際、CLI のページ分割の動作を変更することができます。

デフォルトでは、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 で `aws s3api list-objects` を実行すると、CLI は、Amazon S3 に 4 つの呼び出しを行い、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処理します。

大量のリソースでリストコマンドを実行するときに問題が起きるのは、デフォルトのページサイズが高すぎて、AWS サービスへの呼び出しがタイムアウトする場合です。`--page-size` オプションを使用してより小さなページサイズを指定することで、この問題を解決できます。CLI は引き続きリスト全体を取得しますが、より多くの呼び出しをバックグラウンドで実行し、それぞれの呼び出しでより少ない項目を取得します。

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

より少ない項目を取得するには、--max-items オプションを使用します。CLI は、同じ方法でページ分割しますが、指定した数の項目のみをプリントアウトします。

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

出力される項目の数 (--max-items) が項目の合計数よりも少ない場合、項目の次のセットを取得するために以降のコマンドに渡す NextToken が出力に含まれます。

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

サービスは、呼び出すたびに同じ順序で項目を返すとは限りません。次のトークンをページの中で指定した場合は、予想とは異なる結果が表示されることがあります。これを防止するには、--page-size と --max-items に同じ番号を使用し、CLI のページ分割をサービスのページ分割と同期します。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。

アマゾン ウェブ サービスの使用

このセクションでは、AWS Command Line Interface を使用して AWS のサービスにアクセスする例を示します。以下の例は、AWS CLI を使用して管理タスクを実行する方法を示すことを目的としています。

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 Command Line Interface のヘルプ (p. 36)」を参照してください。

トピック

AWS Command Line Interface を使用した Amazon DynamoDB の使用

AWS Command Line Interface (AWS CLI) では、Amazon DynamoDB がサポートされます。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。

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 では、パラメータ値の短縮構文および JSON をサポートしています。

たとえば、次のコマンドでは、`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を使用して、複数の行を Linux ターミナルに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るか、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
```

次のコマンドでは、新しい項目をテーブルに追加します。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"
    }
}
```

コマンドラインで、有効な JSON を作成するのは難しい場合があります。ただし、AWS CLI は、JSON ファイルを読み込むことができます。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。

Example expression-attributes.json

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

次のように、AWS CLI を使用して、`Query` リクエストを発行できます。この例では、`expression-attributes.json` ファイルの内容は、`--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 AWS CLI を使用する方法に関するドキュメントについては、http://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html を参照してください。

DynamoDB に加えて、AWS CLI を使用して DynamoDB Local を使用できます。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 のテーブルまたはデータを実際に操作することなく、DynamoDB API を使用するアプリケーションを記述できます。すべての API アクションは、DynamoDB Local に転送されます。アプリケーションによってテーブルが作成されるか、データが変更されると、その変更はローカルデータベースに書き込まれます。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 の次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 Command Line Interface を介した Amazon EC2 の使用

AWS CLI を使用して 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、次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。

Amazon EC2 の一般的なタスクの例については、以下のトピックを参照してください。

トピック

キーペアの使用

AWS CLI を使用して、キーペアを作成、表示、削除できます。Amazon EC2 インスタンスを起動して接続する際にキーペアを指定する必要があります。

> Note
>
> コマンド例を試す前に、デフォルトの認証情報を設定してください。

トピック

- キーペアの削除 (p. 62)

キーペアを作成する

MyKeyPair という名前のキーペアを作成するには、create-key-pair コマンドを使用し、--query オプションと --output text オプションを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
```

Windows PowerShell の場合は、> 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、out-file コマンドで ASCII エンコードを明示的に指定する必要があります。

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

結果として得られる MyKeyPair.pem ファイルは次のようになります。

```
-----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、作成時にしか取得できません。

Linux コンピュータで SSH クライアントを使用してインスタンスに接続する場合は、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権限を設定することで、自分以外のユーザーがそれを読み取ることができないようにします。

```
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 AWS に保存され、EC2

管理コンソールで、または aws ec2 describe-key-pairs を呼び出すことで表示できます。たとえば、次のコマンドを使用して、MyKeyPair のフィンガープリントを表示できます。

```
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"
        }
    ]
}
```

キーやフィンガープリントの詳細については、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

MyKeyPair を削除するには、delete-key-pair コマンドを次のように使用します。

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

セキュリティグループの使用

EC2-Classic または EC2-VPC で使用するセキュリティグループを作成します。EC2-Classic と EC2-VPC 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 CLI を使用して、セキュリティグループを作成したり、そこにルールを追加したり、セキュリティグループを削除したりできます。

> **Note**
>
> コマンド例を試す前に、デフォルトの認証情報を設定してください。

トピック

セキュリティグループを作成する

my-sg という名前のセキュリティグループを作成するには、create-security-group コマンドを使用します。

EC2-VPC

次のコマンドでは、指定の VPC に my-sg という名前のセキュリティグループが作成されます。

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

```
}
```

my-sg の初期情報を表示するには、次のように describe-security-groups コマンドを使用します。EC2-VPC 用セキュリティグループは名前では参照できないことに注意してください。

```
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

次のコマンドを実行では、EC2-Classic 用のセキュリティグループが作成されます。

```
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、次のように describe-security-groups コマンドを使用します。

```
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

Windows インスタンスを起動する場合は、TCP ポート 3389 (RDP) のインバウンドトラフィックを許可するルールを追加する必要があります。Linux インスタンスを起動する場合は、TCP ポート 22 (SSH) のイン

バウンドトラフィックを許可するルールを追加する必要があります。authorize-security-group-ingress コマンドを使用して、セキュリティグループにルールを追加します。このコマンドの必須パラメータの 1 つは、コンピュータのパブリック IP アドレス (CIDR 表記) です。

> Note
>
> サービスを使用して、ローカルコンピュータのパブリック IP アドレスを取得できます。たとえば、次のサービスが提供されています。https://checkip.amazonaws.com/. IP アドレスを提供する別のサービスを検索するには、検索フレーズ「what is my IP address」を使用します。ISP 経由で、またはファイアウォールの内側から静的な IP アドレスなしで接続している場合は、クライアントコンピュータで使用されている IP アドレスの範囲を見つける必要があります。

EC2-VPC

次のコマンドでは、ID sg-903004f8 のセキュリティグループに、RDP のルールが追加されます。

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

次のコマンドでは、ID sg-903004f8 のセキュリティグループに、SSH のルールが追加されます。

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

my-sg の変更を表示するには、次のように describe-security-groups コマンドを使用します。

```
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

次のコマンドでは、セキュリティグループ my-sg に、RDP のルールが追加されます。

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

次のコマンドでは、セキュリティグループ my-sg に、SSH のルールが追加されます。

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

my-sg の変更を表示するには、次のように describe-security-groups コマンドを使用します。

```
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 コマンドを使用します。環境にアタッチされているセキュリティグループは削除できないことに注意してください。

EC2-VPC

次のコマンドでは、ID sg-903004f8 のセキュリティグループが削除されます。

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

EC2-Classic

次のコマンドでは、my-sg という名前のセキュリティグループが削除されます。

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

Amazon EC2 インスタンスの使用

AWS CLI を使用して、インスタンスを起動、一覧表示、終了できます。キーペアとセキュリティグループが必要になります。AWS CLI を使用したこれらの作成ついては、「キーペアの使用 (p. 60)」および「セキュリティグループの使用 (p. 62)」を参照してください。また、Amazon Machine Image (AMI) を選択し、AMI ID を書きとめておく必要があります。詳しくは、Linux インスタンス用 Amazon EC2 ユーザーガイドの適切な 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
>
> コマンド例を試す前に、デフォルトの認証情報を設定してください。

トピック

インスタンスの作成

選択した AMI を使用して単一の 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。アカウントがサポートするプラットフォームに応じて、EC2-Classic または EC2-VPC にインスタンスを起動できます。

最初、インスタンスは `pending` 状態ですが、数分後には `running` 状態になります。

EC2-VPC

次のコマンドでは、指定したサブネットで `t2.micro` インスタンスが起動されます。

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-
name MyKeyPair --security-group-ids sg-xxxxxxxx --subnet-id subnet-xxxxxxxx
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
```

```
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "PrivateIpAddress": "10.0.1.114",
            "ProductCodes": [],
            "VpcId": "vpc-1a2b3c4d",
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": ip-10-0-1-114.ec2.internal,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "SubnetId": "subnet-6e7f829e",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
                    "Status": "in-use",
                    "SourceDestCheck": true,
                    "VpcId": "vpc-1a2b3c4d",
                    "Description": "Primary network interface",
                    "NetworkInterfaceId": "eni-a7edb1c9",
                    "PrivateIpAddresses": [
                        {
                            "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                            "Primary": true,
                            "PrivateIpAddress": "10.0.1.114"
                        }
                    ],
                    "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                    "Attachment": {
                        "Status": "attached",
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "AttachmentId": "eni-attach-52193138",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    },
                    "Groups": [
                        {
                            "GroupName": "my-sg",
                            "GroupId": "sg-903004f8"
                        }
                    ],
                    "SubnetId": "subnet-6e7f829e",
                    "OwnerId": "123456789012",
                    "PrivateIpAddress": "10.0.1.114"
                }
            ],
            "SourceDestCheck": true,
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
```

```
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "RootDeviceName": "/dev/sda1",
            "VirtualizationType": "hvm",
            "RootDeviceType": "ebs",
            "Tags": [
                {
                    "Value": "MyInstance",
                    "Key": "Name"
                }
            ],
            "AmiLaunchIndex": 0
        }
    ]
}
```

EC2-Classic

次のコマンドでは、EC2-Classic に `t1.micro` インスタンスが起動されます。

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t1.micro --key-
name MyKeyPair --security-groups my-sg
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "ProductCodes": [],
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": null,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
```

```
            ],
            "ClientToken": null,
            "InstanceType": "t1.micro",
            "NetworkInterfaces": [],
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "RootDeviceName": "/dev/sda1",
            "VirtualizationType": "hvm",
            "RootDeviceType": "ebs",
            "Tags": [
                {
                    "Value": "MyInstance",
                    "Key": "Name"
                }
            ],
            "AmiLaunchIndex": 0
        }
    ]
}
```

インスタンスへのブロックデバイスマッピングの追加

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 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。

ブロックデバイスマッピングをインスタンスに追加するには、`run-instances` を使用する際に `--block-device-mappings` オプションを指定します。

次の例では、標準の Amazon EBS ボリュームが追加され、サイズが 20 GB の `/dev/sdf` にマッピングされます。

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

次の例では、Amazon EBS ボリュームが追加され、スナップショットに基づいて `/dev/sdf` にマッピングされます。スナップショットを指定する場合は、ボリュームサイズを指定する必要はありませんが、その場合、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-xxxxxxxx\"}}]"
```

次の例では、2 つのインスタンスストアボリュームが追加されます。インスタンスで使用できるインスタンスストアボリュームの数は、インスタンスタイプによって異なることに注意してください。

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

次の例では、インスタンスの起動に使用される AMI (`/dev/sdj`) によって指定されたデバイスのマッピングが省略されます。

```
--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「ブロックデバイスマッピング」を参照してください。

名前タグのインスタンスへの追加

タグ `Name=MyInstance` をインスタンスに追加するには、次のように create-tags コマンドを使用します。

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

詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドのリソースにタグを付けるを参照してください。

インスタンスへの接続

インスタンスを実行しているときは、そこに接続して目の前のコンピュータを使用するのと同じように使用できます。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 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。

> Note
>
> コマンド例を試す前に、デフォルトの認証情報を設定してください。

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

Example 1: 指定のインスタンスタイプのインスタンスを一覧表示する

次のコマンドでは、`t2.micro` インスタンスが一覧表示されます。

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

Example 2: 指定のタグでインスタンスを一覧表示する

次のコマンドを実行すると、Name=MyInstance タグを含むインスタンスが一覧表示されます。

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

Example 3: 指定されたイメージを使用して起動されたインスタンスを一覧表示する

次のコマンドでは、AMI `ami-x0123456`、`ami-y0123456`、および `ami-z0123456` から起動されたインスタンスが一覧表示されます。

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

インスタンスの削除

インスタンスを削除するということは、実質的には、そのインスタンスを削除するということです。いったん終了した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。

インスタンスを終了したら、次のように、terminate-instances コマンドを使用します。

```
aws ec2 terminate-instances --instance-ids i-5203422c
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-5203422c",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

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「インスタンスの削除」を参照してください。

AWS Command Line Interface を使用した Amazon Glacier の使用

大きなファイルは、小さく分割してコマンドラインからアップロードすることで、Amazon Glacier にアップロードできます。このトピックでは、AWS CLI を使用したボールトの作成、ファイルの分割、Amazon Glacier へのマルチパートアップロードの設定および実行のプロセスを説明します。

> 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

- アップロードの完了 (p. 73)

Amazon Glacier ボールトを作成する

`aws glacier create-vault` コマンドを使用してボールトを作成します。次のコマンドでは、`myvault` というボールトが作成されます。

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

Note

すべての glacier コマンドではアカウント ID をパラメータが必須です。現在のアカウントを指定するには、ハイフンを使用します。

ファイルのアップロードの準備

テストアップロード用のファイルを作成します。次のコマンドでは、正確に 3 MiB (3 x 1024 x 1024 バイト) のランダムデータを含むファイルが作成されます。

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。上記の例では、ランダムデータのソースとして `/dev/urandom` デバイスファイルを使用します。`fsutil` は Windows で同様の関数を実行します。

Windows

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

次に、ファイルを 1 MiB (1048576 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

`aws glacier initiate-multipart-upload` コマンドを使用して、Amazon 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"
}
```

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

次に、`aws glacier upload-multipart-part` コマンドを使用して各パート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**
>
> 上記の例では、`UPLOADID` シェル変数を逆参照するためにドル記号 (「$」) を使用しています。Windows コマンドラインでは、2 つのパーセント記号 (`%UPLOADID%`) を使用します。

アップロード時に各パートのバイト範囲を指定する必要があります。これにより、Amazon Glacier によって正しい順序で再構成されます。各部分は 1048576 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 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 ハッシュを計算します。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、`aws glacier 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"
}
```

また、`aws glacier describe-vault` を使用してボールトのステータスを確認できます。

```
$ aws glacier describe-vault --account-id - --vault-name myvault
```

```
{
    "SizeInBytes": 3178496,
    "VaultARN": "arn:aws:glacier:us-west-2:123456789012:vaults/myvault",
    "LastInventoryDate": "2015-04-07T00:26:19.028Z",
    "NumberOfArchives": 1,
    "CreationDate": "2015-04-06T21:23:45.708Z",
    "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、「ボールトに関する各種操作」を参照してください。

これで、作成したパートとハッシュ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 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およびチェックサムの計算を参照してください。

AWS Command Line Interface からの AWS Identity and Access Management

このセクションでは、AWS Identity and Access Management (IAM) に関連するいくつかの一般的なタスクおよび AWS Command Line Interface を使用した実行方法を説明します。

ここで示すコマンドは、デフォルトの認証情報とデフォルトのリージョンが設定されていることを前提としています。

トピック

新しい IAM ユーザーとグループを作成する

このセクションでは、新しい IAM グループと新しい IAM ユーザーを作成して、ユーザーをグループに追加する方法を説明します。

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. まず、`create-group` コマンドを使用して、グループを作成します。

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

```
    }
}
```

2. 次に、create-user コマンドを使用して、ユーザーを作成します。

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

3. 最後に、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 グループに MyUser が含まれて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": "2012-12-20T03:03:52Z",
        "GroupId": "AKIAI44QH8DHBEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    },
    "Users": [
        {
            "UserName": "MyUser",
            "Path": "/",
            "CreateDate": "2012-12-20T03:13:02Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/MyUser"
        }
    ],
    "IsTruncated": "false"
}
```

また、AWS マネジメントコンソール を使用して IAM のユーザーとグループを表示することもできます。

IAM ユーザー用の IAM ポリシーを設定する

以下のコマンドは、IAM ポリシーを IAM ユーザーに割り当てる方法を示しています。ここで指定されたポリシーにより、ユーザーに "Power User Access" が提供されます。このポリシーは、IAM コンソールで提供されている Power User Access ポリシーテンプレートと同じです。この例では、ポリシーは MyPolicyFile.json ファイルに保存されます。

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

```
    }
  ]
}
```

ポリシーを指定するには、put-user-policy コマンドを使用します。

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

list-user-policies コマンドで、ポリシーがユーザーに割り当てられていることを確認します。

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

その他のリソース

詳細については、「アクセス権限とポリシーに関するリソース」を参照してください。このトピックでは、アクセス権限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提供しています。

IAM ユーザーの初期パスワードを設定する

次の例では、create-login-profile 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を示します。

```
$ aws iam create-login-profile --user-name MyUser --password My!User1Login8P@ssword
{
    "LoginProfile": {
        "UserName": "MyUser",
        "CreateDate": "2013-01-02T21:10:54.339Z",
        "MustChangePassword": "false"
    }
}
```

update-login-profile コマンドを使用して IAM ユーザーのパスワードを更新します。

IAM ユーザーのセキュリティ認証情報を作成する

次の例では、create-access-key コマンドを使用して IAM ユーザーのセキュリティ認証情報を作成します。一連のセキュリティ認証情報は、アクセスキー ID とシークレットキーで構成されます。IAM ユーザーは、任意の時点で最大で 2 セットの認証情報を持つことができます。3 セット目の認証情報を作成しようとすると、create-access-key コマンドは、"LimitExceeded" エラーを返します。

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

```
    }
}
```

delete-access-key コマンドを使用して IAM ユーザーの認証情報のセットを削除します。アクセスキー ID を使用して、削除する認証情報を指定します。

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

AWS Command Line Interface を使用した Amazon S3 の使用

AWS CLI は Amazon S3 にアクセスするために 2 つの階層の コマンドを提供します。

- 最初の階層は、s3 という名前で、オブジェクトおよびバケットの作成、操作、削除などの頻繁に使用されるオペレーションに対する高レベルコマンドで構成されます。
- 2 番目の階層は、s3api という名前で、バケットアクセスコントロールリスト (ACL) の変更、Cross-Origin Resource Sharing (CORS) の使用、またはポリシーのロギングなどのすべての Amazon S3 オペレーションを公開します。これにより、高レベルコマンドだけでは難しいアドバンスドオペレーションを実行することができます。

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

```
$ aws s3 help
```

または

```
$ aws s3api help
```

> **Note**
>
> AWS CLI は、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。これらのオペレーションは、Amazon S3 によって提供される service-side COPY 操作を使用します。お客様のファイルはクラウドに保存され、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 Command Line Interface での高レベルの S3 コマンドの使用

このセクションでは、高レベルの aws s3 コマンドを使用して、Amazon S3 のバケットとオブジェクトを管理する方法について説明します。

バケットの管理

高レベルの `aws s3` コマンドは、バケットの作成、削除、一覧表示など、よく使用されるバケットオペレーションをサポートします。

バケットの作成

新しいバケットを作成するには、`aws s3 mb` コマンドを使用します。バケット名は一意で、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、およびピリオドを含めることができます。バケット名の最初と最後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含めることはできません。

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

バケットの削除

バケットを削除するには、`aws 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。

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

これにより、バケット内のすべてのオブジェクトとサブフォルダが削除されてから、バケットが削除されます。

> 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。

バケットの一覧表示

すべてのバケットまたはそのコンテンツを一覧表示するには、`aws 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
2013-07-11 17:08:50 my-bucket
2013-07-24 14:55:44 my-bucket2
```

次のコマンドを実行すると、バケット内のすべてのオブジェクトとフォルダ (プレフィックス) が一覧表示されます。

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

次のコマンドを実行すると、*bucket-name*/path 内のオブジェクト (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/
```

```
2013-09-06 18:59:32          3 MyFile2.txt
```

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も便利になります。オブジェクトのコマンドには、aws s3 cp、aws s3 ls、aws s3 mv、aws s3 rm、および 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し、ローカルディレクトリと Amazon S3 バケット間でシームレスに作業を行うことができます。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 バケットにオブジェクトをアップロードするすべての高レベルのコマンド (aws s3 cp、aws s3 mv、および aws s3 sync) は、オブジェクトが大きい場合に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権限を付与するために使用できる --grants オプションが用意されています。次の構文を使用して、アクセス権限のリストに --grants オプションを設定します。

```
--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権限を指定し、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」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、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
```

sync コマンドの形式は次のとおりです。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 から Amazon S3 へ

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

次の例では、my-bucket の path という名前の Amazon S3 フォルダの内容を、現在の作業ディレクトリと同期します。s3 sync は、同期先で同じ名前を持つファイルとは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、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ができます。

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 オプションでは、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、指定したディレクトリ内のすべての項目が同期に含まれます。したがって、--exclude オプションの例外を指定するときは、--include のみが必要です (たとえば、--include は実質的に「除外しない」を意味します)。このオプションは、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 オプションでは、--delete オプションで同期オペレーション中に削除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こともできます。この場合、パラメーター文字列で、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
```

sync コマンドでは、--acl オプションも使用できます。これにより、Amazon S3 にコピーされたファイルにアクセス権限を設定できます。このオプションでは、private、public-read、および public-read-write の値を使用できます。

```
$ 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`、または `rm` とともにディレクトリ/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、`sync` コマンドと同じように `--exclude`、`--include`、および `--acl` オプションも使用できます。

AWS Command Line Interface で API レベル (s3api) を使用する

API レベルコマンド (`s3api` コマンドセットに含まれる) は、Amazon S3 API への直接アクセスを提供し、高レベル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。このセクションでは、API レベルコマンドを説明し、いくつかの例を示します。Amazon S3 の例の詳細については、「s3api コマンドラインリファレンス」を参照し、使用可能なコマンドをリストから選択します。

カスタム ACL

高レベルコマンドでは、`--acl` オプションを使用でき、事前定義された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、バケット全体の ACL を設定することはできません。これを行うには、API レベルコマンド `put-bucket-acl` を使用します。次の例では、2 人の AWS ユーザー (user1@example.com および user2@example.com) に完全なコントロールが付与され、全員に読み取り許可が付与されます。

```
$ 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"'
```

カスタム ACL の詳細については、PUT Bucket acl を参照してください。`put-bucket-acl` などの `s3api` ACL コマンドでは、同様の引数の略記法を使用します。

ロギングポリシー

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、MyBucket のロギングポリシーを設定します。AWS ユーザー user@example.com がログファイルを全面的に管理し、すべてのユーザーはログファイルにアクセスできます。`put-bucket-acl` コマンドでは、Amazon S3 のログ配信システムに必要なアクセス権限 (WRITE および READ_ACP) を付与する必要があります。

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

logging.json

```
{
  "LoggingEnabled": {
    "TargetBucket": "MyBucket",
    "TargetPrefix": "MyBucketLogs/",
```

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

Amazon SNS での AWS Command Line Interface の使用

このセクションでは、Amazon Simple Notification Service (Amazon SNS) に関連するいくつかの一般的なタスクおよび AWS Command Line Interface を使用した実行方法を説明します。

トピック

トピックの作成

次のコマンドでは、**my-topic** という名前のトピックが作成されます。

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

TopicArn を書き留めます。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

次のコマンドでは、E メールプロトコルおよび通知エンドポイント用の E メールアドレスを使用してトピックにサブスクライブされます。

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

```
}
```

サブスクライブコマンドにリストされた E メールアドレスに E メールメッセージが送信されます。E メールメッセージには次の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
```

[Confirm subscription] をクリックすると、「Subscription confirmed!」という通知メッセージが、次のような情報とともにブラウザに表示されます。

```
Subscription confirmed!

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

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

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

トピックへの発行

次のコマンドでは、トピックにメッセージが発行されます。

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

「Hello World!」というテキストを含む E メールメッセージがemailusername@example.com に送信されます

トピックからサブスクリプションを解除する

次のコマンドでは、トピックからサブスクリプションが解除されます。

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

トピックからサブスクリプションが解除されたことを確認するには、次のように入力します。

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

トピックの削除

次のコマンドでは、トピックが削除されます。

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

トピックの削除を確認するには、次のように入力します。

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

AWS Command Line Interface を使用した Amazon Simple Workflow Service の使用

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。

コマンドのリストおよび Amazon SWF のドメインでの動作については、以下のトピックを参照してください。

トピック

Amazon SWF コマンドのカテゴリ別リスト

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一覧表示します。ここに挙げられるコマンドは機能カテゴリ別に一覧表示されます。

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の Amazon SWF セクション を参照するか、次のコマンドを使用してください。

```
$ aws swf help
```

特定のコマンドのヘルプを取得するには、コマンド名の後に `help` ディレクティブを使用します。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 Command Line Interface を使用して Amazon SWF ドメインを操作する (p. 88)」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 Command Line Interface を使用して Amazon SWF ドメインを操作する (p. 88)」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 Command Line Interface を使用して Amazon SWF ドメインを操作する

このセクションでは、AWS CLI を使用して Amazon SWF ドメインの一般的なタスクを実行する方法を示します。

トピック

ドメインのリスト化

アカウントに登録されている Amazon SWF ドメインを一覧表示するには、`swf list-domains` を使用できます。必須パラメータは 1 つしかありません。`--registration-status` は、`REGISTERED` または `DEPRECATED` に設定できます。

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. 91)」を参照してください。推測される通り、廃止されたドメインは返されます。

ページサイズを結果を制限するように設定する

多数のドメインがある場合は、`--maximum-page-size` パラメータを、返された結果の数を制限するように設定します。指定した最大数よりも多くの結果を取得した場合は、`nextPageToken` を受け取り、`list-domains` への次の呼び出しに送って、追加のエントリを取得します。

以下に `--maximum-page-size` の使用例を示します。

```
$ aws swf list-domains --registration-status REGISTERED --maximum-page-size 1
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        }
    ],
    "nextPageToken": "ANeXAMPLEtOKENiSpRETTYlONG=="
}
```

Note

返される `nextPageToken` は、実際にははるかに長くなります。この値は、あくまで例です。

`--next-page-token` 引数で `nextPageToken` の値を指定して、もう一度呼び出しを行うと、別のページの結果を取得できます。

```
$ aws swf list-domains --registration-status REGISTERED --maximum-page-size 1 --next-page-
token "ANeXAMPLEtOKENiSpRETTYlONG=="
```

```
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

取得する結果のページがそれ以上存在しない場合、nextPageToken が、結果に返されることはありません。

ドメインに関する情報の取得

特定のドメインに関する詳細情報を取得するには、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、これは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"
    }
}
```

ドメインの登録

新しいドメインを登録するには、swf register-domain を使用します。必須パラメータは 2 つあります。ドメイン名を取得する --name と、このドメインのワークフロー実行データを保持する日数 (最大 90 日間) を指定する整数を取得する --workflow-execution-retention-period-in-days です (詳細については、「Amazon SWFFAQ」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。それ以外の場合は、ワークフロー実行データは、指定した日数が経過した後は保持されません。

新しいドメインの登録の例を示します。

```
$ aws swf register-domain --name MyNeatNewDomain --workflow-execution-retention-period-in-
days 0
```

ドメインを登録すると何も返されませんが (""), swf list-domains または swf describe-domain を使用して、新しいドメインを表示できます。以下に例を示します。

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "MyNeatNewDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
```

```
        }
    ]
}
```

以下に swf describe-domain の使用例を示します。

```
$ 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 (表示はできますが、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。

```
$ aws swf list-domains --registration-status REGISTERED
{
    "domainInfos": [
        {
            "status": "REGISTERED",
            "name": "ExampleDomain"
        },
        {
            "status": "REGISTERED",
            "name": "mytest"
        }
    ]
}
```

--registration-status DEPRECATED を list-domains と共に使用して、廃止されたドメインを確認できます。

```
$ aws swf list-domains --registration-status DEPRECATED
{
    "domainInfos": [
        {
            "status": "DEPRECATED",
            "name": "MyNeatNewDomain"
        }
    ]
}
```

また、describe-domain を使用して、廃止されたドメインに関する情報を取得できます。

```
$ aws swf describe-domain --name MyNeatNewDomain
```

```
{
    "domainInfo": {
        "status": "DEPRECATED",
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
    }
}
```

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

pip のインストール後は、OS の aws 環境変数に PATH 実行可能ファイルを追加したり、モードを変更して実行可能にしたりする必要が生じる場合があります。

エラー: aws: コマンドが見つかりません

OS 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. 14)
- 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. 9)

aws が PATH にあるが、それでもこのエラーが表示される場合は、適切なファイルモード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
```

エラー: アクセス許可が拒否されました

aws スクリプトに、実行可能なファイルモードがあることを確認します。たとえば、755 と指定します。

chmod +x を実行して、ファイルを実行可能にします。

```
$ chmod +x ~/.local/bin/aws
```

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、正しい認証情報が使用されていることを確認します。

```
$ 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
```

正しい認証情報が使用されている場合、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、data を実行して時間を確認します。

```
date
```

システムクロックがオフの場合は、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。

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。