AWS Command Line Interface

用户指南

AWS Command Line Interface: 用户指南

Table of Contents

AWS Command Line Interface 是什么？

AWS Command Line Interface (AWS CLI) 是一种开源工具，让您能够在命令行 Shell 中使用命令与 AWS 服务进行交互。仅需最少的配置，您就可以从常用终端程序中的命令提示符开始使用基于浏览器的 AWS 管理控制台提供的相同功能：

- Linux Shell – 使用常见 Shell 程序（例如 `bash`、`zsh` 和 `tsch`）在 Linux, OS X, or Unix 中运行命令。
- Windows 命令行 – 在 Windows 上，在 PowerShell 或 Windows 命令提示符处运行命令。
- 远程 – 通过远程终端（如 PuTTY 或 SSH）或者使用 AWS Systems Manager 在 Amazon Elastic Compute Cloud (Amazon EC2) 实例上运行命令。

所有 IaaS（基础设施即服务）AWS 管理和访问 AWS API 和 CLI 中提供的 AWS 管理控制台 函数。新的 AWS IaaS 功能和服务在启动时或在 180 天启动期内通过 API 和 CLI 提供全部 AWS 管理控制台 功能。

AWS CLI 提供对 AWS 服务的公共 API 的直接访问。您可以使用 AWS CLI 探索服务的功能，可以开发 Shell 脚本来管理资源。或者，也可以通过 AWS 开发工具包利用所学知识开发其他语言的程序。

除了低级别的 API 等效命令，多项 AWS 服务还为 AWS CLI 提供了自定义项。自定义项可能包括更高级别的命令，可简化具有复杂 API 的服务的使用。例如，`aws s3` 命令集提供熟悉的语法，用于管理 Amazon Simple Storage Service (Amazon S3) 中的文件。

Example 将文件上传到 Amazon S3

`aws s3 cp` 提供了一个类似于 shell 的复制命令，并自动执行分段上传，以快速、弹性地传输大型文件。

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

使用低级别命令 (在 `aws s3api` 下提供) 执行同一任务需要更多的工作。

根据您的用例，您可能希望使用 AWS 开发工具包或 适用于 PowerShell 的 AWS 工具 之一：

- 适用于 PowerShell 的 AWS 工具
- AWS SDK for Java
- 适用于 .NET 的 AWS 开发工具包
- AWS SDK for JavaScript
- 适用于 Ruby 的 AWS 开发工具包
- AWS SDK for Python (Boto)
- 适用于 PHP 的 AWS 开发工具包
- 适用于 Go 的 AWS 开发工具包
- AWS Mobile SDK for iOS
- 适用于 Android 的 AWS 移动软件开发工具包

您可以在 aws-cli 存储库 中的 GitHub 上查看和复制 AWS CLI 的源代码。加入 GitHub 上的用户社区，提供反馈、请求功能和提交自己的文章！

使用本指南中的示例

本指南中示例的格式是使用下列约定进行设置的：

- 提示符 – 命令提示符通常显示为美元符号后跟一个空格 (`$ `)。对于特定于 Windows 的命令，使用 `C:\>` 作为提示符。键入命令时，请勿包含提示符。
- 目录 – 当必须从特定目录执行命令时，目录名称将显示在提示符符号之前。
- 用户输入 – 您应在命令行处输入的命令文本采用 **`user input`** 格式。
- 可替换文本 – 变量文本（包括您选择的资源的名称，或您必须包含在命令中的由 AWS 服务生成的 ID）采用的格式为*`#####`*。在多行命令中或需要特定键盘输入的命令中，键盘命令也可显示为可替换文本。
- 输出 – AWS 服务返回的输出显示在用户输入下方，采用 `computer output` 格式。

例如，以下命令包含用户输入、可替换文本和输出。

```
$ 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
```

要使用该示例，请在命令行处输入 **`aws configure`** 并按 Enter。**`aws configure`** 是命令。此命令是交互式的，因此 AWS CLI 将输出文本行，用来提示您输入其他信息。依次输入每个访问密钥，然后按 Enter。然后，以显示的格式输入 AWS 区域名称，按 Enter，然后最后一次按 Enter 以跳过输出格式设置。最终 Enter 命令将显示为可替换文本，因为这一行没有用户输入。否则，此命令将是隐含的。

以下示例介绍一个简单的非交互式命令（来自服务的输出采用 JSON 格式）。

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

要使用此示例，请输入命令的完整文本（提示符后突出显示的文本），然后按 Enter。安全组的名称 *`my-sg`* 是可替换的。在这种情况下，您可以使用显示的组名称，但您可能希望使用更具描述性的名称。

> **Note**
>
> 必须替换的参数（如 AWS Access Key ID）和应替换的参数（如 group name）均显示为*`#######`*。如果某个参数是必须替换的，则描述示例的文本中会为其添加注释。

JSON 文档（包括大括号）是输出。如果您将 CLI 配置为以文本或表格式进行输出，输出的格式将有差异。JSON 是默认输出格式。

关于 Amazon Web Services

Amazon Web Services (AWS) 是数字基础设施服务的集合，开发人员可在开发应用程序时对其进行利用。这些服务包括计算、存储、数据库和应用程序同步（消息发送和队列）。AWS 采用即付即用的服务模式。您只需为您或您的应用程序使用的服务付费。此外，AWS 还提供免费使用套餐，以便让其作为原型制作和实验平台更易实现。在此套餐中，低于某种使用水平的服务是免费的。有关 AWS 成本和免费套餐的更多信息，请参阅试用免费使用套餐中的 AWS。要获取 AWS 账户，请打开 AWS 主页，然后单击注册。

安装 AWS CLI

安装 AWS Command Line Interface (AWS CLI) 的方式

先决条件

- Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+
- Windows、Linux, OS X, or Unix

> **Note**
>
> 较早版本的 Python 可能无法兼容所有 AWS 服务。如果在安装或使用 AWS CLI 时看到 `InsecurePlatformWarning` 或弃用通知，请更新到更高的版本。

您可以查找最新 CLI 的版本号，网址为：https://github.com/aws/aws-cli/blob/master/CHANGELOG.rst。

在本指南中，所示的命令假设您安装了 Python v3，并且所示的 `pip` 命令使用 `pip3` 版本。

使用 pip 安装 AWS CLI

AWS CLI 在 Linux、Windows 和 macOS 上的主要分发方式为 `pip`。这是一个用于 Python 的程序包管理器，提供了简单的方式来安装、升级和删除 Python 程序包及其相关组件。

安装当前 AWS CLI 版本

经常更新 AWS CLI 以支持新服务和命令。要确定您是否拥有最新版本，请查看 GitHub 上的版本页面。

如果您已经有 `pip` 和支持的 Python 版本，则可以使用以下命令安装 AWS CLI：如果您安装了 Python 3+ 版本，我们建议您使用 **`pip3`** 命令。

```
$ pip3 install awscli --upgrade --user
```

`--upgrade` 选项通知 `pip3` 升级已安装的任何必要组件。`--user` 选项通知 `pip3` 将程序安装到用户目录的子目录中，以避免修改您的操作系统所使用的库。

升级到最新版本的 AWS CLI

我们建议您定期检查以查看是否有新的 AWS CLI 版本，并尽可能升级到该版本。

使用 `pip3 list -o` 命令来检查哪些程序包已“过时”：

```
$ aws --version
aws-cli/1.16.170 Python/3.7.3 Linux/4.14.123-111.109.amzn2.x86_64 botocore/1.12.160

$ pip3 list -o
```

```
Package    Version  Latest   Type
---------- -------- -------- -----
awscli     1.16.170 1.16.198 wheel
botocore   1.12.160 1.12.188 wheel
```

由于前一个命令显示有较新版本的 AWS CLI 可用，您可以运行 `pip3 install --upgrade` 以获取最新版本：

```
$ pip3 install --upgrade --user awscli
Collecting awscli
  Downloading https://files.pythonhosted.org/packages/dc/70/
b32e9534c32fe9331801449e1f7eacba6a1992c2e4af9c82ac9116661d3b/awscli-1.16.198-py2.py3-none-
any.whl (1.7MB)
     |################################| 1.7MB 1.6MB/s
Collecting botocore==1.12.188 (from awscli)
  Using cached https://files.pythonhosted.org/packages/10/
cb/8dcfb3e035a419f228df7d3a0eea5d52b528bde7ca162f62f3096a930472/botocore-1.12.188-py2.py3-
none-any.whl
Requirement already satisfied, skipping upgrade: docutils>=0.10 in ./venv/lib/python3.7/
site-packages (from awscli) (0.14)
Requirement already satisfied, skipping upgrade: rsa<=3.5.0,>=3.1.2 in ./venv/lib/
python3.7/site-packages (from awscli) (3.4.2)
Requirement already satisfied, skipping upgrade: colorama<=0.3.9,>=0.2.5 in ./venv/lib/
python3.7/site-packages (from awscli) (0.3.9)
Requirement already satisfied, skipping upgrade: PyYAML<=5.1,>=3.10; python_version !=
 "2.6" in ./venv/lib/python3.7/site-packages (from awscli) (3.13)
Requirement already satisfied, skipping upgrade: s3transfer<0.3.0,>=0.2.0 in ./venv/lib/
python3.7/site-packages (from awscli) (0.2.0)
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.7.1 in ./venv/lib/
python3.7/site-packages (from botocore==1.12.188->awscli) (0.9.4)
Requirement already satisfied, skipping upgrade: urllib3<1.26,>=1.20; python_version >=
 "3.4" in ./venv/lib/python3.7/site-packages (from botocore==1.12.188->awscli) (1.24.3)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1;
 python_version >= "2.7" in ./venv/lib/python3.7/site-packages (from botocore==1.12.188-
>awscli) (2.8.0)
Requirement already satisfied, skipping upgrade: pyasn1>=0.1.3 in ./venv/lib/python3.7/
site-packages (from rsa<=3.5.0,>=3.1.2->awscli) (0.4.5)
Requirement already satisfied, skipping upgrade: six>=1.5 in ./venv/lib/python3.7/site-
packages (from python-dateutil<3.0.0,>=2.1; python_version >= "2.7"->botocore==1.12.188-
>awscli) (1.12.0)
Installing collected packages: botocore, awscli
  Found existing installation: botocore 1.12.160
    Uninstalling botocore-1.12.160:
      Successfully uninstalled botocore-1.12.160
  Found existing installation: awscli 1.16.170
    Uninstalling awscli-1.16.170:
      Successfully uninstalled awscli-1.16.170
Successfully installed awscli-1.16.198 botocore-1.12.188
```

在虚拟环境中安装 AWS CLI

如果您在尝试随 `pip3` 一起安装 AWS CLI 时遇到问题，可以在虚拟环境中安装 AWS CLI (p. 16) 来隔离工具及其依赖项。或者，您可以使用与通常不同的 Python 版本。

使用安装程序安装 AWS CLI

若要在 Linux, OS X, or Unix 上进行离线或自动安装，请尝试捆绑安装程序 (p. 17)。捆绑安装程序包括 AWS CLI 和其依赖项，以及为您执行安装的 Shell 脚本。

在 Windows 上，您也可以使用 MSI 安装程序 (p. 11)。这两种方法都简化了初始安装。但缺点是，当新版本的 AWS CLI 发布时，升级更加困难。

安装后需要执行的步骤

设置路径以包含 AWS CLI

在安装 AWS CLI 后，您可能需要将可执行文件路径添加到您的 `PATH` 变量中。有关特定于平台的说明，请参阅以下主题：

- Linux – 将 AWS CLI 可执行文件添加到命令行路径 (p. 9)
- Windows – 将 AWS CLI 可执行文件添加到命令行路径 (p. 13)
- macOS – 将 AWS CLI 可执行文件添加到 macOS 命令行路径 (p. 16)

通过运行 `aws --version` 来验证 AWS CLI 是否已正确安装。

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

使用您的凭证配置 AWS CLI

在运行 CLI 命令之前，您必须先使用您的凭证配置 AWS CLI。

通过在 AWS CLI 配置文件 (p. 22)（默认存储在用户的主目录中）中定义配置文件 (p. 29)，您可以在本地存储凭证信息。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

> Note
>
> 如果您在 Amazon EC2 实例上运行，可以从实例元数据中自动检索凭证。有关更多信息，请参阅 实例元数据 (p. 34)。

升级到最新版本的 AWS CLI

定期更新 AWS CLI，以便添加对新服务和命令的支持。要更新到最新版本的 AWS CLI，请再次运行安装命令。有关 AWS CLI 最新版本的详细信息，请参阅 AWS CLI 发行说明。

```
$ pip3 install awscli --upgrade --user
```

卸载 AWS CLI

如果需要卸载 AWS CLI，请使用 `pip uninstall`。

```
$ pip3 uninstall awscli
```

如果您没有 Python 和 pip，则使用适合您的环境的过程。

针对每个环境的详细说明

在 Linux 上安装 AWS CLI

您可以使用 pip（一种适用于 Python 的程序包管理器）在大多数 Linux 发行版上安装 AWS Command Line Interface (AWS CLI) 及其依赖项。

> **Important**
>
> awscli 程序包在存储库中可用于其他程序包管理器（如 apt 和 yum），但除非您通过 pip 或使用捆绑安装程序 (p. 17)获得该程序包，否则不保证您获得最新版本。

如果您已有 pip，请按照主要安装主题 (p. 3)中的说明执行操作。运行 pip --version 可查看您的 Linux 版本是否已包含 Python 和 pip。如果您安装了 Python 3+ 版本，我们建议您使用 pip3 命令。

```
$ pip3 --version
```

如果您还没有安装 pip，请检查以查看安装的是哪个版本的 Python。

```
$ python --version
```

或

```
$ python3 --version
```

如果还没有 Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+，则首先必须安装 Python (p. 9)。如果已安装 Python，可继续安装 pip 和 AWS CLI。

小节目录

安装 pip

如果尚未安装 pip，可以使用 Python 打包权威机构 提供的脚本进行安装。

安装 pip

1. 使用 curl 命令下载安装脚本。

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

2. 使用 Python 运行脚本以下载并安装最新版本的 pip 和其他必需的支持包。

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

或

```
$ python3 get-pip.py --user
```

当您包含 --user 开关时，脚本将 pip 安装到路径 ~/.local/bin。

3. 确保包含 pip 的文件夹是您的 PATH 变量的一部分。

a. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 echo $SHELL。

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

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

b. 在配置文件脚本末尾添加与以下示例类似的导出命令。

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

此命令将路径（在本示例中为 ~/.local/bin）插入到现有 PATH 变量的前面。

c. 将配置文件重新加载到当前会话中，以使更改生效。

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

4. 接下来，可以进行测试，以验证是否正确安装了 pip。

```
$ pip3 --version
pip 19.0.3 from ~/.local/lib/python3.7/site-packages (python 3.7)
```

通过 pip 安装 AWS CLI

使用 pip 安装 AWS CLI。

```
$ pip3 install awscli --upgrade --user
```

当您使用 --user 开关时，pip 将 AWS CLI 安装到 ~/.local/bin。

验证 AWS CLI 是否已正确安装。

```
$ aws --version
```

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

如果出现错误，请参阅排查 AWS CLI 错误 (p. 101)。

升级到最新版本的 AWS CLI

我们建议您定期检查以查看是否有新的 AWS CLI 版本，并尽可能升级到该版本。

使用 `pip list -o` 命令来检查哪些程序包已“过时”：

```
$ aws --version
aws-cli/1.16.170 Python/3.7.3 Linux/4.14.123-111.109.amzn2.x86_64 botocore/1.12.160

$ pip3 list -o
Package    Version  Latest   Type
---------- -------- -------- -----
awscli     1.16.170 1.16.198 wheel
botocore   1.12.160 1.12.188 wheel
```

由于前一个命令显示有较新版本的 AWS CLI 可用，您可以运行 `pip install --upgrade` 以获取最新版本：

```
$ pip3 install --upgrade --user awscli
Collecting awscli
  Downloading https://files.pythonhosted.org/packages/dc/70/
b32e9534c32fe9331801449e1f7eacba6a1992c2e4af9c82ac9116661d3b/awscli-1.16.198-py2.py3-none-
any.whl (1.7MB)
     |################################| 1.7MB 1.6MB/s
Collecting botocore==1.12.188 (from awscli)
  Using cached https://files.pythonhosted.org/packages/10/
cb/8dcfb3e035a419f228df7d3a0eea5d52b528bde7ca162f62f3096a930472/botocore-1.12.188-py2.py3-
none-any.whl
Requirement already satisfied, skipping upgrade: docutils>=0.10 in ./venv/lib/python3.7/
site-packages (from awscli) (0.14)
Requirement already satisfied, skipping upgrade: rsa<=3.5.0,>=3.1.2 in ./venv/lib/
python3.7/site-packages (from awscli) (3.4.2)
Requirement already satisfied, skipping upgrade: colorama<=0.3.9,>=0.2.5 in ./venv/lib/
python3.7/site-packages (from awscli) (0.3.9)
Requirement already satisfied, skipping upgrade: PyYAML<=5.1,>=3.10; python_version !=
 "2.6" in ./venv/lib/python3.7/site-packages (from awscli) (3.13)
Requirement already satisfied, skipping upgrade: s3transfer<0.3.0,>=0.2.0 in ./venv/lib/
python3.7/site-packages (from awscli) (0.2.0)
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.7.1 in ./venv/lib/
python3.7/site-packages (from botocore==1.12.188->awscli) (0.9.4)
Requirement already satisfied, skipping upgrade: urllib3<1.26,>=1.20; python_version >=
 "3.4" in ./venv/lib/python3.7/site-packages (from botocore==1.12.188->awscli) (1.24.3)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1;
 python_version >= "2.7" in ./venv/lib/python3.7/site-packages (from botocore==1.12.188-
>awscli) (2.8.0)
Requirement already satisfied, skipping upgrade: pyasn1>=0.1.3 in ./venv/lib/python3.7/
site-packages (from rsa<=3.5.0,>=3.1.2->awscli) (0.4.5)
Requirement already satisfied, skipping upgrade: six>=1.5 in ./venv/lib/python3.7/site-
packages (from python-dateutil<3.0.0,>=2.1; python_version >= "2.7"->botocore==1.12.188-
>awscli) (1.12.0)
Installing collected packages: botocore, awscli
  Found existing installation: botocore 1.12.160
    Uninstalling botocore-1.12.160:
      Successfully uninstalled botocore-1.12.160
  Found existing installation: awscli 1.16.170
    Uninstalling awscli-1.16.170:
      Successfully uninstalled awscli-1.16.170
Successfully installed awscli-1.16.198 botocore-1.12.188
```

将 AWS CLI 可执行文件添加到命令行路径

在使用 pip 进行安装后，可能需要将 aws 可执行文件添加到操作系统的 PATH 环境变量中。

您可以运行以下命令验证 pip 已将 AWS CLI 安装到哪个文件夹中。

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

您可以将此路径 ~/.local/bin/ 作为参考，因为在 Linux 中 /home/username 对应于 ~。

如果您忽略了 --user 开关且未在用户模式下安装，可执行文件可能位于 Python 安装的 bin 文件夹中。如果您不知道 Python 的安装位置，请运行此命令。

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

输出可能是符号链接的路径，而不是实际的可执行文件。运行 ls -al 以查看所指向的路径。

```
$ ls -al /usr/local/bin/python
/usr/local/bin/python -> ~/.local/Python/3.6/bin/python3.6
```

如果这是在安装 pip (p. 6)的步骤 3 中添加到路径的相同文件夹，则不必再执行任何操作。否则，请再次执行步骤 3a 到 3c 将该附加文件夹添加到路径中。

在 Linux 上安装 Python

如果您的分发没有随 Python 提供，或者提供的是较早的版本，请在安装 pip 和 AWS CLI 之前安装 Python。

在 Linux 上安装 Python 3

1. 检查是否已安装 Python。

   ```
   $ python --version
   ```

   或

   ```
   $ python3 --version
   ```

   Note

   如果您的 Linux 分发版本附带了 Python，则可能需要安装 Python 开发人员程序包以获取编译扩展和安装 AWS CLI 时需要的标头和库。使用程序包管理器安装开发人员程序包（名称通常为 python-dev 或 python-devel）。

2. 如果尚未安装 Python 2.7 或更高版本，请使用分发版本的程序包管理器来安装 Python命令和程序包名称会有所不同：

   - 在 Debian 衍生系统（如 Ubuntu）上，请使用 apt。检查适用于您的 Python 版本的 apt 存储库。然后，运行如下命令（替换为正确的程序包名称）：

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

- 在 Red Hat 及其衍生系统上，请使用 yum。检查适用于您的 Python 版本的 yum 存储库。然后，运行如下命令（替换为正确的程序包名称）：

```
$ sudo yum install python36
```

- 在 SUSE 及其衍生系统上，请使用 zypper。检查适用于您的 Python 版本的存储库。然后，运行如下命令（替换为正确的程序包名称）：

```
$ sudo zypper install python3
```

有关程序包的安装位置以及使用方法的详细信息，请参阅适用于您系统的程序包管理器和 Python 的文档。

3. 打开命令提示符或 shell，并运行以下命令验证 Python 是否已正确安装。

```
$ python3 --version
Python 3.6.8
```

在 Amazon Linux 上安装 AWS CLI

该 AWS Command Line Interface (AWS CLI) 预装在 Amazon Linux 和 Amazon Linux 2 上。使用以下命令检查当前安装的版本。

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

> **Important**
>
> 使用 sudo 完成一个命令，向该命令授予对您的系统的完全访问权限。我们建议仅在不再存在安全选项时使用该命令。对于诸如 pip 之类的命令，我们建议您避免使用 sudo，方式是使用 Python 虚拟环境 (venv)；或者通过指定 --user 选项，在用户文件夹而不是系统文件夹中安装。

如果您使用 yum 程序包管理器，您可以使用以下命令安装 AWS CLI：yum install aws-cli。您可以使用命令 yum update 获取 yum 存储库中可用的最新版本。

> **Note**
>
> yum 存储库不归 Amazon 所有或维护，因此可能不包含最新版本。我们建议您使用 pip 来获取最新版本。

先决条件

验证 Python 和 pip 是否均已安装。有关更多信息，请参阅 在 Linux 上安装 AWS CLI (p. 6)。

在 Amazon Linux 上安装或升级 AWS CLI（用户）

1. 使用 pip3 install 安装最新版本的 AWS CLI。如果您安装了 Python 3+ 版本，我们建议您使用 pip3。如果您在 Python 虚拟环境 (venv) 中运行命令，则不需要使用 --user 选项。

```
$ pip3 install --upgrade --user awscli
```

2. 将安装位置添加到 PATH 变量的开头。

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

将此命令添加到配置文件的启动脚本（例如，`~/.bashrc`）的末尾，以在命令行会话之间保留更改。

3. 使用 `aws --version` 验证您正在运行新版本。

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

在 Windows 上安装 AWS CLI

可以在 Windows 上使用独立安装程序或 `pip`（一种适用于 Python 的程序包管理器）来安装 AWS Command Line Interface (AWS CLI)。如果您已有 `pip`，请按照主要安装主题 (p. 3)中的说明执行操作。

小节目录

使用 MSI 安装程序安装 AWS CLI

Microsoft Windows XP 或更高版本支持 AWS CLI。对于 Windows 用户，MSI 安装程序包提供了一种熟悉而方便的方式来安装 AWS CLI，且无需安装其他任何必备软件。

更新发布后，您必须重复安装过程以获取最新版本的 AWS CLI。要经常更新，请考虑使用 pip (p. 12)，让您的更新操作更轻松。

使用 MSI 安装程序安装 AWS CLI

1. 下载相应的 MSI 安装程序。

   - 下载适用于 Windows（64 位）的 AWS CLI MSI 安装程序
   - 下载适用于 Windows（32 位）的 AWS CLI MSI 安装程序
   - 下载 AWS CLI 安装文件（包括 32 位和 64 位 MSI 安装程序，并将自动安装正确的版本）

   Note

   AWS CLI 的 MSI 安装程序不适用于 Windows Server 2008（版本 6.0.6002）。在此版本的 Windows Server 中使用 pip (p. 12) 进行安装。

2. 运行下载的 MSI 安装程序或设置文件。
3. 按照屏幕上的说明进行操作。

默认情况下，CLI 安装到 `C:\Program Files\Amazon\AWSCLI`（64 位版本）或 `C:\Program Files (x86)\Amazon\AWSCLI`（32 位版本）。要确认安装，请在命令提示符下使用 `aws --version` 命令（打开开始菜单并搜索 `cmd` 以启动命令提示符）。

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

键入命令时，请勿包含提示符符号（上面显示的 `C:\>`）。程序列表中包含这些符号是为了区分您键入的命令与 CLI 返回的输出。除非是特定于 Windows 的命令，否则本指南其余部分使用通用提示符符号 `$`。

如果 Windows 无法找到该程序，您需要关闭并重新打开命令提示符以刷新该路径，或手动将安装目录添加到您的 PATH (p. 13) 环境变量。

更新 MSI 安装

AWS CLI 会定期更新。查看 GitHub 上的版本页面，了解何时发布了最新版本。要更新到最新版本，请按照前面的说明，再次下载和运行 MSI 安装程序。

卸载 AWS CLI

要卸载 AWS CLI，请打开 Control Panel (控制面板)，然后选择 Programs and Features (程序和功能)。选择名为 AWS Command Line Interface 的条目，然后选择 Uninstall (卸载) 启动卸载程序。收到提示时，请确认您要卸载 AWS CLI。

您还可以使用以下命令，从命令行启动 Programs and Features (程序和功能) 程序。

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

在 Windows 上使用 Python 和 pip 安装 AWS CLI

Python Software Foundation 为包含 pip 的 Windows 提供了安装程序。

安装 Python 3.6 和 **pip** (Windows)

1. 从 Python.org 的下载页面下载 Python Windows x86-64 安装程序。
2. 运行安装程序。
3. 选择 Add Python 3 to PATH (将 Python 3 添加到 PATH)。
4. 选择 Install Now。

安装程序在您的用户文件夹中安装 Python 并将其程序文件夹添加到您的用户路径。

使用 **pip3** 安装 AWS CLI (Windows)

如果您使用的是 Python 3+ 版本，我们建议您使用 pip3 命令。

1. 从开始菜单中打开命令提示符。
2. 使用以下命令验证 Python 和 pip 是否已正确安装。

```
C:\> python --version
Python 3.7.1
C:\> pip3 --version
pip 18.1 from c:\program files\python37\lib\site-packages\pip (python 3.7)
```

3. 使用 pip 安装 AWS CLI。

```
C:\> pip3 install awscli
```

4. 验证 AWS CLI 是否已正确安装。

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

要升级到最新版本，请重新运行安装命令。

```
C:\> pip3 install --user --upgrade awscli
```

将 AWS CLI 可执行文件添加到命令行路径

在使用 pip 安装 AWS CLI 后，可能需要将 aws 程序添加到操作系统的 PATH 环境变量中。对于 MSI 安装，此操作将自动执行，但如果 aws 命令在您安装它后不运行，则您可能需要手动设置它。

如果此命令返回一个响应，则应准备好运行该工具。默认情况下，where 命令将显示它所找到的指定程序所在的系统路径：

```
C:\> where aws
C:\Program Files\Amazon\AWSCLI\bin\aws.exe
```

您可以通过运行以下命令找到安装 aws 程序的位置。

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

否则，where 命令会返回以下错误，这表示程序并不在系统路径下，因此您无法只通过键入其名称来运行。

```
C:\> where c:\ aws
INFO: Could not find files for the given pattern(s).
```

在这种情况下，请在运行 where 命令时结合使用 /R *path* 参数，以要求命令搜索所有文件夹并进行查看，然后您必须手动添加路径。使用命令行或 Windows 资源管理器发现它在计算机上的安装位置。

```
C:\> where /R c:\ aws
c:\Program Files\Amazon\AWSCLI\bin\aws.exe
c:\Program Files\Amazon\AWSCLI\bincompat\aws.cmd
c:\Program Files\Amazon\AWSCLI\runtime\Scripts\aws
c:\Program Files\Amazon\AWSCLI\runtime\Scripts\aws.cmd
...
```

所显示的路径取决于安装 AWS CLI 所采用的方法。

典型路径包括：

- Python 3 和 **pip3** – C:\Program Files\Python37\Scripts\
- Windows 较早版本上的 Python 3 和 **pip3** --user 选项 – %USERPROFILE%\AppData\Local\Programs\Python\Python37\Scripts
- Windows 10 上的 Python 3 和 **pip3** --user 选项 – %USERPROFILE%\AppData\Roaming\Python\Python37\Scripts
- MSI 安装程序（64 位）– C:\Program Files\Amazon\AWSCLI\bin
- MSI 安装程序（32 位）– C:\Program Files (x86)\Amazon\AWSCLI\bin

> **Note**
>
> 包含版本号的文件夹名称可能有所不同。上述示例反映所使用的是 Python 版本 3.7。根据需要替换为您使用的版本号。

修改您的 PATH 变量 (Windows)

1. 按 Windows 键并输入 **environment variables**。
2. 选择 Edit environment variables for your account。

3. 选择 PATH，然后选择编辑。
4. 将路径添加到 Variable value (变量值) 字段。例如：`C:\new\path`
5. 选择 OK 两次以应用新设置。
6. 关闭任何运行的命令提示符并重新打开命令提示符窗口。

在 macOS 上安装 AWS CLI

在 macOS 上安装 AWS Command Line Interface (AWS CLI) 的推荐方法是使用捆绑安装程序。捆绑安装程序包含所有依赖项，并可以离线使用。

> **Important**
>
> 捆绑安装程序不支持安装到包含空格的路径。

小节目录

先决条件

- Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+

检查您的 Python 安装。

```
$ python --version
```

如果您的计算机上还没有安装 Python，或者您希望安装 Python 的其他版本，请按照在 Linux 上安装 AWS CLI (p. 6)中的过程执行操作。

使用捆绑安装程序安装 AWS CLI

使用捆绑安装程序，在命令行中执行以下步骤来安装 AWS CLI。

使用捆绑安装程序安装 AWS CLI

以下是可方便地复制和粘贴以作为一组命令运行的步骤，各个步骤的详述见下文。有关每一行的具体作用，请参阅下文中各个步骤中的相关描述。

```
curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
unzip awscli-bundle.zip
sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

1. 将 AWS CLI 捆绑安装程序文件下载到您当前的工作文件夹中。

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

2. 将程序包解压缩到当前工作文件夹中的同名文件夹中。

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

Note

如果没有 unzip，请使用您喜欢的程序包管理器或等效工具进行安装。

3. 运行安装程序。该命令将 AWS CLI 安装到 /usr/local/aws，并在 /usr/local/bin 目录中创建符号链接 aws。使用 -b 选项创建符号链接将免除在用户的 $PATH 变量中指定安装目录的需要。这应该能让所有用户在任何目录下通过键入 aws 来调用 AWS CLI。

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

Note

默认情况下，安装脚本在系统默认版本的 Python 下运行。如果已安装其他 Python 版本并且需要使用该版本安装 AWS CLI，请指定该版本（通过包括 Python 应用程序的绝对路径）来运行安装脚本。例如：

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

要查看 -i 和 -b 选项的说明，请使用 -h 选项。

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

使用 pip 在 macOS 上安装 AWS CLI

您还可以直接使用 pip 安装 AWS CLI。如果您没有 pip，请按照主要安装主题 (p. 3)中的说明执行操作。运行 pip3 --version 可查看您的 macOS 版本是否已包含 Python 和 pip3。

```
$ pip3 --version
```

在 macOS 上安装 AWS CLI

1. 从 Python.org 的下载页面下载并安装最新版本的 Python。
2. 下载并运行 Python 打包权威机构提供的 pip3 安装脚本。

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

3. 使用新安装的 pip3 安装 AWS CLI。如果您使用的是 Python 3+ 版本，我们建议您使用 pip3 命令。

```
$ pip3 install awscli --upgrade --user
```

4. 验证 AWS CLI 是否已正确安装。

```
$ aws --version
AWS CLI 1.16.116 (Python 3.6.8)
```

如果未找到该程序，请将它添加到命令行路径 (p. 16)。

要升级到最新版本，请重新运行安装命令。

```
$ pip3 install awscli --upgrade --user
```

将 AWS CLI 可执行文件添加到 macOS 命令行路径

在使用 pip 进行安装后，可能需要将 aws 程序添加到操作系统的 PATH 环境变量中。程序的位置取决于 Python 的安装位置。

Example AWS CLI 安装位置 - 带 Python 3.6 和 **pip**（用户模式）的 macOS

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

将上面示例中的版本替换为您的 Python 版本。

如果您不知道 Python 的安装位置，请运行 `which python`。

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

输出可能是符号链接的路径，而不是实际的程序。运行 `ls -al` 以查看所指向的路径。

```
$ ls -al /usr/local/bin/python
~/Library/Python/3.7/bin/python3.6
```

pip 将程序安装到 Python 应用程序所在的文件夹中。将此文件夹添加到 PATH 变量。

修改您的 **PATH** 变量 (Linux, OS X, or Unix)

1. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 `echo $SHELL`。

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

   - Bash – `.bash_profile`、`.profile` 或 `.bash_login`。
   - Zsh – `.zshrc`
   - Tcsh – `.tcshrc`、`.cshrc` 或 `.login`。
2. 向配置文件脚本中添加导出命令。

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

   在本示例中，此命令将路径 `~/.local/bin` 添加到当前 PATH 变量中。
3. 将更新的配置文件加载到当前会话中。

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

在虚拟环境中安装 AWS CLI

您可以通过在虚拟环境中安装 AWS Command Line Interface (AWS CLI)，避免所需版本与其他 pip 程序包冲突。

在虚拟环境中安装 AWS CLI

1. 使用 pip 安装 virtualenv。

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

2. 创建虚拟环境并命名它。

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

或者，您也可以使用 -p 选项指定默认版本以外的 Python 版本。

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

3. 激活新虚拟环境。

Linux, OS X, or Unix

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

Windows

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

4. 将 AWS CLI 安装到虚拟环境中。

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

5. 验证 AWS CLI 是否已正确安装。

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

您可以使用 deactivate 命令退出虚拟环境。不管何时启动新会话，都必须重新激活环境。

要升级到最新版本，请重新运行安装命令。

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

使用捆绑安装程序安装 AWS CLI (Linux, OS X, or Unix)

在 Linux, OS X, or Unix 上，可以使用捆绑安装程序来安装 AWS Command Line Interface (AWS CLI)。捆绑安装程序包含所有依赖项，并可以离线使用。

> **Important**
>
> 捆绑安装程序不支持安装到包含空格的路径。

小节目录

先决条件

- Linux, OS X, or Unix
- Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+

检查您的 Python 安装。

```
$ python --version
```

如果您的计算机上还没有安装 Python，或者您希望安装 Python 的其他版本，请按照在 Linux 上安装 AWS CLI (p. 6)中的过程执行操作。

使用捆绑安装程序安装 AWS CLI

以下步骤使您能够从任何版本的 Linux 或 macOS 上的命令行安装 AWS CLI。

要下载直接（而不使用 curl），请使用此链接：

•

以下是可剪切和粘贴以作为一组命令运行的安装命令的摘要，各个命令的具体解释见下文。

```
curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
unzip awscli-bundle.zip
sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

使用捆绑安装程序，在命令行中执行以下步骤来安装 AWS CLI。

使用捆绑安装程序安装 AWS CLI

1. 使用以下命令下载 AWS CLI 捆绑安装程序：

   如果您在北京区域：

   ```
   $ curl "s3.cn-north-1.amazonaws.com.cn/cli/downloads/awscli-bundle.zip" -o "awscli-bundle.zip"
   ```

   如果您在宁夏区域：

   ```
   $ curl "s3.cn-northwest-1.amazonaws.com.cn/cli/downloads/awscli-bundle.zip" -o "awscli-bundle.zip"
   ```

2. 解压缩程序包。

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

   Note

   如果没有 `unzip`，请使用 Linux 发行版的内置程序包管理器进行安装。

3. 运行安装可执行文件。

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

Note

默认情况下，安装脚本在系统默认版本的 Python 下运行。如果您已安装 Python 的可选版本并希望使用该版本安装 AWS CLI，请使用该版本按 Python 可执行文件的绝对路径运行安装脚本。例如：

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

安装程序在 `/usr/local/aws` 中安装 AWS CLI，并在 `/usr/local/bin` 目录中创建符号链接 `aws`。使用 `-b` 选项创建符号链接将免除在用户的 `$PATH` 变量中指定安装目录的需要。这应该能让所有用户通过在任何目录下键入 `aws` 来调用 AWS CLI。

要查看 `-i` 和 `-b` 选项的说明，请使用 `-h` 选项。

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

在不使用 Sudo 的情况下安装 AWS CLI (Linux, OS X, or Unix)

如果您没有 `sudo` 权限，或打算仅为当前用户安装 AWS CLI，则可使用先前命令的修改版本。

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

这会将 AWS CLI 安装到默认位置 (`~/.local/lib/aws`) 并在 `~/bin/aws` 中创建符号链接 (symlink)。确保您的 `~/bin` 环境变量中包含 `PATH`，以使该符号链接生效。

```
$ 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

为确保您的 `$PATH` 设置在多次会话之间保留，请将 `export` 行添加到 shell 配置文件 (`~/.profile`、`~/.bash_profile` 等) 。

卸载 AWS CLI

除了可选的符号链接之外，捆绑安装程序不会将任何内容放在安装目录之外，所以卸载十分简单，就是直接删除这两个项目。

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

配置 AWS CLI

本节介绍如何配置 AWS Command Line Interface (AWS CLI) 在与 AWS 交互时使用的设置，包括您的安全凭证、默认输出格式和默认 AWS 区域。

> Note
>
> AWS 要求所有传入的请求都进行加密签名。AWS CLI 为您执行该操作。“签名”包括日期/时间戳。因此，您必须确保正确设置计算机的日期和时间。否则，如果签名中的日期/时间与 AWS 服务认定的日期/时间相差太远，AWS 会拒绝请求。

小节目录

快速配置 AWS CLI

对于一般用途，`aws configure` 命令是设置 AWS CLI 安装的最快方法。

```
$ 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
```

键入该命令时，AWS CLI 会提示您输入四条信息（访问密钥、私有访问密钥、AWS 区域和输出格式）。以下各节介绍了这些信息。AWS CLI 将这些信息存储在名为 `default` 的配置文件（一个设置集合）中。每当您运行的 AWS CLI 命令未明确指定要使用的配置文件时，就会使用 `default` 配置文件中的信息。

访问密钥和私有访问密钥

`AWS Access Key ID` 和 `AWS Secret Access Key` 是您的 AWS 凭证。它们与 AWS Identity and Access Management (IAM) 用户或角色相关联，用于确定您拥有的权限。有关如何使用 IAM 服务创建用户的教程，请参阅 IAM 用户指南 中的创建您的第一个 IAM 管理员用户和组。

要获取 IAM 用户的访问密钥 ID 和秘密访问密钥。

访问密钥包含访问密钥 ID 和秘密访问密钥，用于签署对 AWS 发出的编程请求。如果没有访问密钥，您可以使用AWS 管理控制台进行创建。建议您使用 AWS 账户根用户 访问秘钥而不是使用 IAM 账户根用户访问秘钥。IAM 让您可以安全地控制对您的 AWS 账户中 AWS 服务和资源的访问。

仅当创建访问密钥时，您才能查看或下载秘密访问密钥。以后您无法恢复它们。不过，您随时可以创建新的访问密钥。您还必须拥有执行所需 IAM 操作的权限。有关更多信息，请参阅 IAM 用户指南 中的访问 IAM 资源所需的权限。

1. 打开 IAM 控制台。
2. 在控制台的导航窗格中，选择 Users。
3. 选择您的 IAM 用户名称（而不是复选框）。
4. 选择安全证书选项卡，然后选择创建访问秘钥。
5. 要查看新访问秘钥，请选择显示。您的凭证与下面类似：
   - 访问密钥 ID：AKIAIOSFODNN7EXAMPLE
   - 私有访问密钥：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. 要下载密钥对文件，请选择下载 .csv 文件。将密钥存储在安全位置。

   请对密钥保密以保护您的 AWS 账户，切勿通过电子邮件发送密钥。请勿对组织外部共享密钥，即使有来自 AWS 或 Amazon.com 的询问。合法代表 Amazon 的任何人永远都不会要求您提供密钥。

相关主题

- 什么是 IAM？（在 IAM 用户指南 中）
- AWS General Reference 中的 AWS 安全证书

区域

`Default region name` 标识默认情况下您要将请求发送到的服务器所在的 AWS 区域。通常是离您最近的区域，但可以是任意区域。例如，您可以键入 `us-west-2` 以使用美国西部（俄勒冈）。除非在命令中另行指定，否则这是所有后续请求将发送到的区域。

> **Note**
>
> 使用 AWS CLI 时，必须明确指定或通过设置默认区域来指定 AWS 区域。有关可用区域的列表，请参阅区域和终端节点。AWS CLI 使用的区域指示符与您在 AWS 管理控制台 URL 和服务终端节点中看到的名称相同。

输出格式

`Default output format` 指定结果的格式。可以是以下列表中的任何值。如果未指定输出格式，则默认使用 `json`。

- **`json`**：输出采用 JSON 字符串的格式。
- **`text`**：输出采用多行制表符分隔的字符串值的格式，如果要将输出传递给文本处理器（如 `grep`、`sed` 或 `awk`），则该格式非常有用。
- **`table`**：输出采用表格形式，使用字符 +|- 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

创建多个配置文件

如果您使用上一节中显示的命令，则结果是名为 `default` 的单个配置文件。通过指定 `--profile` 选项并分配名称，您可以创建可按名称引用的其他配置。以下示例创建一个名为 `produser` 的配置文件。您指定的凭证所来自的账户和区域可与其他配置文件的完全不同。

```
$ aws configure --profile produser
```

```
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
```

然后，运行命令时，可以省略 `--profile` 选项，并使用 `default` 配置文件中存储的凭证和设置。

```
$ aws s3 ls
```

或者，您也可以指定 `--profile` *profilename* 并使用按该名称存储的凭证和设置。

```
$ aws s3 ls --profile produser
```

要更新任何设置，只需再次运行 `aws configure`（根据要更新的配置文件，使用或不使用 `--profile` 参数），并根据需要输入新值。下面几节包含有关 `aws configure` 创建的文件、其他设置和命名配置文件的更多信息。

配置设置和优先顺序

AWS CLI 使用一组凭证提供程序 查找 AWS 凭证。每个凭证提供程序在不同的位置查找凭证，例如系统或用户环境变量、本地 AWS 配置文件，或在命令行上显式声明为参数。AWS CLI 通过按以下顺序调用提供程序来查找凭证和配置设置，在找到要使用的一组凭证时停止：

1. 命令行选项 (p. 32) – 您可以在命令行上指定 `--region`、`--output` 和 `--profile` 作为参数。
2. 环境变量 (p. 31) – 您可以在环境变量中存储值：`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 `AWS_SESSION_TOKEN`。如果存在环境变量，则会使用这些变量。
3. CLI 凭证文件 (p. 22) – 这是运行命令 `aws configure` 时更新的文件之一。该文件位于 `~/.aws/credentials`（在 Linux, OS X, or Unix 上）或 `C:\Users\`*USERNAME*`\.aws\credentials`（在 Windows 上）。该文件可以包含 `default` 配置文件和任何命名配置文件的凭证详细信息。
4. CLI 配置文件 (p. 22) – 这是运行命令 `aws configure` 时更新的另一个文件。该文件位于 `~/.aws/config`（在 Linux, OS X, or Unix 上）或 `C:\Users\`*USERNAME*`\.aws\config`（在 Windows 上）。该文件包含默认配置文件和任何命名配置文件的配置设置。
5. 容器凭证 – 您可以将 IAM 角色与每个 Amazon Elastic Container Service (Amazon ECS) 任务定义相关联。之后，该任务的容器就可以使用该角色的临时凭证。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的适用于任务的 IAM 角色。
6. 实例配置文件凭证 – 您可以将 IAM 角色与每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例相关联。之后，在该实例上运行的代码就可以使用该角色的临时凭证。凭证通过 Amazon EC2 元数据服务提供。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的适用于 Amazon EC2 的 IAM 角色和 IAM 用户指南 中的使用实例配置文件。

配置和证书文件

您可以将常用的配置设置和凭证保存在由 AWS CLI 维护的文件中。这些文件分为可按名称引用的多个部分。这称为“配置文件”。除非您另行指定，否则 CLI 将使用在名为 `default` 的配置文件中找到的设置。要使用备用设置，您可以创建和引用其他配置文件。您也可以通过设置某个支持的环境变量或使用命令行参数来覆盖个别设置。

- 配置设置存储在何处？ (p. 23)
- 全局设置 (p. 24)
- S3 自定义命令设置 (p. 27)

配置设置存储在何处？

AWS CLI 将使用 `aws configure` 指定的凭证存储在主目录中名为 `.aws` 的文件夹中名为 `credentials` 的本地文件中。使用 `aws configure` 指定的其他配置选项存储在名为 `config` 的本地文件中，该文件也存储在主目录的 `.aws` 文件夹中。主目录位置因操作系统而异，但在 Windows 中使用环境变量 `%UserProfile%` 引用，在基于 Unix 的系统中使用 `$HOME` 或 `~`（波形符）引用。

例如，下面的命令列出 `.aws` 文件夹的内容。

Linux, OS X, or Unix

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

Windows

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

AWS CLI 使用两个文件将敏感的凭证信息（位于 `~/.aws/credentials` 中）与不太敏感的配置选项（位于 `~/.aws/config` 中）分开。

通过将 `AWS_CONFIG_FILE` 环境变量设置为另一个本地路径，可以为 `config` 文件指定非默认位置。有关更多信息，请参阅 环境变量 (p. 31)。

在 Config 文件中存储证书

AWS CLI 也可以从 `config` 文件读取凭证。您可以将所有配置文件设置保存在一个文件中。如果在一个配置文件的两个位置都有证书（假设您使用 `aws configure` 更新了配置文件密钥），则证书文件中的密钥有优先顺序。
这些文件也被各种语言软件开发工具包 (SDK) 使用。如果除 AWS CLI 外您还使用一个软件开发工具包，当证书不是存储在它自己的文件中时，您会收到其他警告。

CLI 为上一部分中配置的配置文件生成的文件如下所示。

`~/.aws/credentials`

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

`~/.aws/config`

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

Note

上面的示例介绍具有单个默认配置文件的文件。有关具有多个命名配置文件的文件的示例，请参阅命名配置文件 (p. 29)。

当您使用共享配置文件指定 IAM 角色时，AWS CLI 将调用 AWS STS AssumeRole 操作来检索临时凭证。随后，这些凭证将存储起来（存储在 `~/.aws/cli/cache` 中）。后续 AWS CLI 命令将使用缓存的临时凭证，直到它们过期，这时 AWS CLI 将自动刷新这些凭证。

支持的 config 文件设置

主题

config 文件支持以下设置。将使用指定（或默认）配置文件中列出的值，除非它们被具有相同名称的环境变量或具有相同名称的命令行选项覆盖。

您可以通过直接使用文本编辑器编辑配置文件或使用 aws configure set 命令来配置这些设置。使用 --profile 设置指定要修改的配置文件。例如，以下命令设置名为 integ 的配置文件中的 region 设置。

```
aws configure set region us-west-2 --profile integ
```

您还可以使用 get 子命令检索任何设置的值。

```
$ aws configure get region --profile default
us-west-2
```

如果输出为空，则不会显式设置该设置，并将使用默认值。

全局设置

aws_access_key_id (p. 20)

指定用作凭证一部分的对命令请求进行身份验证的 AWS 访问密钥。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS_ACCESS_KEY_ID 环境变量覆盖。请注意，您不能将访问密钥 ID 指定为命令行选项。

```
aws_access_key_id = 123456789012
```

aws_secret_access_key (p. 20)

指定用作凭证一部分的对命令请求进行身份验证的 AWS 私有密钥。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS_SECRET_ACCESS_KEY 环境变量覆盖。请注意，您不能将私有访问密钥指定为命令行选项。

```
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

role_arn (p. 36)

指定要用于运行 AWS CLI 命令的 IAM 角色的 Amazon 资源名称 (ARN)。此外，还必须指定以下参数之一以标识有权代入此角色的凭证：

- source_profile
- credential_source

```
role_arn = arn:aws:iam::123456789012:role/role-name
```

source_profile (p. 36)

指定包含长期凭证的命名配置文件，AWS CLI 可使用这些凭证代入通过 role_arn 参数指定的角色。不能在同一配置文件中同时指定 source_profile 和 credential_source。

```
source_profile = production-profile
```

credential_source (p. 36)

在 EC2 实例或 EC2 容器中使用，指定 AWS CLI 在何处可以找到要用于代入通过 role_arn 参数指定的角色的凭证。不能在同一配置文件中同时指定 source_profile 和 credential_source。

此参数具有三个值：

- Environment：从环境变量检索源凭证。
- Ec2InstanceMetadata：将附加到 EC2 实例配置文件的 IAM 角色用作源凭证。
- EcsContainer：将附加到 ECS 容器的 IAM 角色用作源凭证。

```
credential_source = Ec2InstanceMetadata
```

role_session_name (p. 40)

指定要附加到角色会话的名称。此值在 AWS CLI 调用 AssumeRole 操作时将提供给 RoleSessionName 参数，并成为代入角色用户 ARN 的一部分：arn:aws:sts::*123456789012*:assumed-role/*role_name*/*role_session_name*。此参数为可选参数。如果未提供此值，则将自动生成会话名称。此名称显示在与此会话关联的条目的 AWS CloudTrail 日志中。

```
role_session_name = maria_garcia_role
```

mfa_serial (p. 38)

代入角色时要使用的 MFA 设备的标识号。仅当代入角色的信任策略包含需要 MFA 身份验证的条件，此值才是必需的。该值可以是硬件设备（例如 GAHT12345678）的序列号，也可以是虚拟 MFA 设备（例如 arn:aws:iam::123456789012:mfa/*user*）的 Amazon 资源名称 (ARN)。

duration_seconds

指定角色会话的最大持续时间（以秒为单位）。该值的范围在 900 秒（15 分钟）到角色的最大会话持续时间设置之间。此参数为可选参数，默认情况下，该值设置为 3600 秒。

aws_session_token

指定 AWS 会话令牌。只有在手动指定临时安全凭证时才需要会话令牌。虽然它可以存储在 config 文件中，但我们建议您将其存储在 credentials 文件中。

可以被 AWS_SESSION_TOKEN 环境变量覆盖。您不能将会话令牌指定为命令行选项。

```
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

external_id (p. 39)

第三方用于在其客户账户中代入角色的唯一标识符。这将映射到 AssumeRole 操作中的 ExternalId 参数。此参数是可选的，除非角色的信任策略指定 ExternalId 必须为特定值。

ca_bundle

指定用于验证 SSL 证书的 CA 证书捆绑包（具有 .pem 扩展名的文件）。

可以被 AWS_CA_BUNDLE 环境变量或 --ca-bundle 命令行选项覆盖。

```
ca_bundle = dev/apps/ca-certs/cabundle-2019mar05.pem
```

cli_follow_urlparam

指定 CLI 是否尝试跟踪命令行参数中以 http:// 或 https:// 开头的 URL 链接。启用后，将检索到的内容而不是 URL 用作参数值。

- true：这是默认值。配置后，将抓取任何以 http:// 或 https:// 开头的字符串参数，并将任何下载的内容用作该命令的参数值。
- false：CLI 不将以 http:// 或 https:// 开头的参数字符串值与其他字符串区别对待。

该条目没有等效的环境变量或命令行选项。

```
cli_follow_urlparam = false
```

cli_timestamp_format

指定输出中包含的时间戳值的格式。可以指定以下任一值：

- none：这是默认值。按原样显示 HTTP 查询响应中收到的时间戳值。
- iso8601：按 ISO 8601 指定的格式重新格式化时间戳。

该条目没有等效的环境变量或命令行选项。

```
cli_timestamp_format = iso8601
```

credential_process (p. 34)

指定 CLI 运行的外部命令，以生成或检索用于该命令的身份验证凭证。命令必须以特定格式返回凭证。有关如何使用该设置的更多信息，请参阅使用外部进程获取凭证 (p. 34)。

该条目没有等效的环境变量或命令行选项。

```
credential_process = /opt/bin/awscreds-retriever --username susan
```

web_identity_token_file (p. 40)

指定一个文件的路径，该文件包含由身份提供商提供的 OAuth 2.0 访问令牌或 OpenID Connect ID 令牌。AWS CLI 加载此文件的内容，并将其作为 WebIdentityToken 参数传递给 AssumeRoleWithWebIdentity 操作。

output (p. 21)

指定使用该配置文件请求的命令的默认输出格式。您可以指定以下任意值：

- **json**：这是默认值。输出采用 JSON 字符串的格式。
- **text**：输出采用多行制表符分隔的字符串值的格式，如果要将输出传递给文本处理器（如 grep、sed 或 awk），则该格式非常有用。
- **table**：输出采用表格形式，使用字符 +|- 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

可以被 AWS_DEFAULT_OUTPUT 环境变量或 --output 命令行选项覆盖。

```
output = table
```

parameter_validation

指定 CLI 客户端在将参数发送到 AWS 服务终端节点之前是否尝试验证参数。

- true：这是默认值。配置后，CLI 将执行命令行参数的本地验证。
- false：配置后，CLI 在将命令行参数发送到 AWS 服务终端节点前不对其进行验证。

该条目没有等效的环境变量或命令行选项。

```
parameter_validation = false
```

区域 (p. 21)

对于使用该配置文件请求的命令，指定要将请求发送到的默认 AWS 区域。您可以指定可用于所选服务的任何区域代码（有关服务和区域代码的列表，请参阅 Amazon Web Services 一般参考 中的 AWS 区域和终端节点）。

可以被 AWS_DEFAULT_REGION 环境变量或 --region 命令行选项覆盖。

```
region = us-west-2
```

tcp_keepalive

指定 CLI 客户端是否使用 TCP keep-alive 数据包。

该条目没有等效的环境变量或命令行选项。

```
tcp_keepalive = false
```

api_versions

某些 AWS 服务维护多个 API 版本以支持向后兼容性。默认情况下，CLI 命令使用最新的可用 API 版本。您可以通过在 config 文件中包含 api_versions 设置来指定要用于配置文件的 API 版本。

这是一个“嵌套”设置，后跟一个或多个缩进行，每行标识一个 AWS 服务和要使用的 API 版本。请参阅每项服务的文档以了解可用的 API 版本。

以下示例显示如何为两种 AWS 服务指定 API 版本。这些 API 版本仅用于在包含这些设置的配置文件下运行的命令。

```
api_versions =
    ec2 = 2015-03-01
    cloudfront = 2015-09-017
```

S3 自定义命令设置

Amazon S3 支持多项配置 CLI 如何执行 S3 操作的设置。一些设置适用于 s3api 和 s3 命名空间中的所有 S3 命令。其他的则专门用于抽象常见操作的 S3“自定义”命令，而不仅仅是对 API 操作的一对一映射。aws s3 传输命令 cp、sync、mv 和 rm 具有可用于控制 S3 传输的其他设置。

可以通过在 config 文件中指定 s3 嵌套设置来配置所有这些选项。每个设置在其自己的行上缩进。

Note

这些设置完全是可选的。即使不配置这些设置中的任何一个，您也应该能够成功使用 aws s3 传输命令。提供这些设置是为了让您能够调整性能或匹配运行这些 aws s3 命令的特定环境。

以下设置适用于 s3 或 s3api 命名空间中的任何 S3 命令。

use_accelerate_endpoint

为所有 s3 和 s3api 命令使用 Amazon S3 加速终端节点。默认值为 False。该设置与 use_dualstack_endpoint 设置互斥。

如果设置为 true，CLI 会将所有 Amazon S3 请求定向到 s3-accelerate.amazonaws.com 的 S3 加速终端节点。要使用该终端节点，您必须让您的存储桶使用 S3 加速。使用存储桶寻址的虚拟样式发送所有请求：*my-bucket*.s3-accelerate.amazonaws.com。不会将任何

ListBuckets、CreateBucket 和 DeleteBucket 请求发送到加速终端节点，因为该终端节点不支持这些操作。如果将任何 s3 或 s3api 命令的 --endpoint-url 参数设置为 https://s3-accelerate.amazonaws.com 或 http://s3-accelerate.amazonaws.com，也可以设置该行为。

use_dualstack_endpoint

为所有 s3 和 s3api 命令使用 Amazon S3 双 IPv4 / IPv6 终端节点。默认值为 False。该设置与 use_accelerate_endpoint 设置互斥。

如果设置为 true，CLI 会将所有 Amazon S3 请求定向到配置的区域的双 IPv4/IPv6 终端节点。

addressing_style

指定要使用的寻址样式。这将控制存储桶名称位于主机名还是 URL 中。有效值为：path、virtual 和 auto。默认值为 auto。

构造 S3 终端节点的样式有两种。第一种称为 virtual，它将存储桶名称包含为主机名的一部分。例如：https://*bucketname*.s3.amazonaws.com。另一种为 path 样式 - 将存储桶名称视为 URI 中的路径。例如：https://s3.amazonaws.com/*bucketname*。CLI 中的默认值是使用 auto，它尝试尽可能使用 virtual 样式，但在需要时回退到 path 样式。例如，如果您的存储桶名称与 DNS 不兼容，则存储桶名称不能是主机名的一部分，而必须位于路径中。使用 auto 时，CLI 将检测这种情况并自动切换到 path 样式。如果将寻址方式设置为 path，您必须确保在 AWS CLI 中配置的 AWS 区域与存储桶的区域匹配。

payload_sigining_enabled

指定是否对 sigv4 负载进行 SHA256 签名。默认情况下，使用 https 时，将对流式上传（UploadPart 和 PutObject）禁用该设置。默认情况下，对于流式上传（UploadPart 和 PutObject），此设置为 false，但仅限存在 ContentMD5（默认生成）并且终端节点使用 HTTPS 时。

如果设置为 true，则 S3 请求接收 SHA256 校验和形式的额外内容验证（替您计算并包含在请求签名中）。如果设置为 false，则不计算校验和。禁用该设置可减少校验和计算产生的性能开销。

以下设置仅适用于 s3 命名空间命令集中的命令：

max_concurrent_requests

指定最大并发请求数。默认值是 10。

aws s3 传输命令是多线程的。在任意给定时间，都可以运行多个 Amazon S3 请求。例如，当您使用命令 aws s3 cp localdir s3://bucket/ --recursive 将文件上传到 S3 存储桶时，AWS CLI 可以并行上传文件 localdir/file1、localdir/file2 和 localdir/file3。设置 max_concurrent_requests 指定可同时运行的最大传输操作数。

您可能由于以下原因而需要更改该值：

- 减小该值 – 在某些环境中，默认的 10 个并发请求可能会占用过多的系统资源。这可能导致连接超时或系统响应速度变慢。减小该值可减少 S3 传输命令消耗的资源。但不利后果是 S3 传输可能需要更长时间才能完成。如果使用了限制带宽的工具，则可能需要减小该值。
- 增大该值 – 在某些情况下，您可能希望 S3 传输根据需要使用尽可能多的网络带宽，以尽可能快地完成任务。在这种情况下，默认的并发请求数可能不足以利用所有可用的网络带宽。增大该值可缩短完成 S3 传输所需的时间。

max_queue_size

指定任务队列中的最大任务数。默认值是 1000。

AWS CLI 在内部使用这样一种模型：将 S3 任务排队，然后由数量受 max_concurrent_requests 限制的使用者执行。任务通常映射到单个 S3 操作。例如，任务可以是 PutObjectTask、GetObjectTask 或 UploadPartTask。任务添加到队列的速度可能比使用者完成任务的速度快得多。为避免无限制增长，任务队列大小设置了特定大小的上限。该设置用于更改该最大数量的值。

您通常不需要更改该设置。该设置还对应于 CLI 知道需要运行的任务数。这意味着，默认情况下 CLI 只能查看前 1000 个任务。在 S3 命令得知执行的任务总数之前，进度线显示总计 ...。增大该值意味着 CLI 可更快得知所需任务的总数（假设排队速度快于任务完成速度）。但不利后果是更大的最大队列大小需要更多的内存。

multipart_threshold

指定 CLI 用于单个文件的分段传输的大小阈值。默认值为 8 MB。

上传、下载或复制文件时，如果文件超出该大小，S3 命令将切换到分段操作。您可以通过以下两种方式之一指定该值：

- 文件大小（以字节为单位）。例如：`1048576`。
- 文件大小及大小后缀。您可以使用 `KB`、`MB`、`GB` 或 `TB`。例如，`10MB`、`1GB`。

  Note

  S3 可能会对可用于分段操作的有效值施加约束。有关更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的 S3 分段上传文档。

multipart_chunksize

指定 CLI 用于单个文件的分段传输的块大小。默认值为 8 MB，最小值为 5 MB。

当文件传输超出 `multipart_threshold` 时，CLI 将文件分成该大小的块。可以使用与 `multipart_threshold` 相同的语法指定该值，即整数形式的字节数，或使用大小和后缀。

max_bandwidth

指定向 Amazon S3 上传数据和从其下载数据可使用的最大带宽。默认为无限制。

这限制了 S3 命令可用于向 S3 传输数据和从 S3 传输数据的最大带宽。该值仅适用于上传和下载；它不适用于复制或删除。值以每秒字节数表示。该值可以指定为：

- 一个整数。例如，`1048576` 将最大带宽使用率设置为每秒 1 兆字节。
- 一个整数，后跟速率后缀。可以使用以下格式指定速率后缀：`KB/s`、`MB/s` 或 `GB/s`。例如，`300KB/s` 和 `10MB/s`。

通常，我们建议您先尝试通过降低 `max_concurrent_requests` 来降低带宽使用率。如果这样做没有将带宽使用率限制到所需速率，接下来您可以使用 `max_bandwidth` 设置进一步限制带宽使用率。这是因为 `max_concurrent_requests` 控制当前运行的线程数。如果您先降低 `max_bandwidth` 但保持较高的 `max_concurrent_requests` 设置，则可能导致线程进行不必要等待，从而造成过多的资源消耗和连接超时。

这些设置都在 `config` 文件中的顶层 `s3` 关键字下设置，如以下 `development` 配置文件示例所示：

```
[profile development]
s3 =
  max_concurrent_requests = 20
  max_queue_size = 10000
  multipart_threshold = 64MB
  multipart_chunksize = 16MB
  max_bandwidth = 50MB/s
  use_accelerate_endpoint = true
  addressing_style = path
```

命名配置文件

AWS CLI 支持使用存储在 `config` 和 `credentials` 文件中的多个命名配置文件中的任何一个。您可以通过在 `aws configure` 中使用 `--profile` 选项或通过向 `config` 和 `credentials` 文件中添加条目来配置其他配置文件。

下面的示例介绍一个有两个配置文件的 `credentials` 文件。第一个在运行没有配置文件的 CLI 命令时使用。第二个在运行有 `--profile user1` 参数的 CLI 命令时使用。

`~/.aws/credentials`（Linux 和 Mac）或 **`%USERPROFILE%\.aws\credentials`** (Windows)

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

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

每个配置文件可以指定不同的凭证（可能来自不同的 IAM 用户），还可以指定不同的 AWS 区域和输出格式。

`~/.aws/config`（Linux 和 Mac）或 **`%USERPROFILE%\.aws\config`** (Windows)

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

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

> **Important**
>
> `credentials` 文件使用的命名格式与 CLI `config` 文件用于命名配置文件的格式不同。仅当在 `config` 文件中配置命名配置文件时，才包含前缀单词“`profile`”。在 `credentials` 文件中创建条目时，请勿 使用单词 `profile`。

通过 AWS CLI 使用配置文件

要使用命名配置文件，请向您的命令添加 `--profile` *`profile-name`* 选项。以下示例列出了使用先前示例文件中 `user1` 配置文件中定义的凭证和设置的所有 Amazon EC2 实例。

```
$ aws ec2 describe-instances --profile user1
```

要为多个命令使用一个命名配置文件，通过在命令行设置 `AWS_PROFILE` 环境变量可以避免在每个命令中指定配置文件。

Linux, OS X, or Unix

```
$ export AWS_PROFILE=user1
```

设置环境变量会更改默认配置文件，直到 Shell 会话结束或直到您将该变量设置为其他值。通过将环境变量放在 shell 的启动脚本中，可使环境变量在未来的会话中继续有效。有关更多信息，请参阅 环境变量 (p. 31)。

Windows

```
C:\> setx AWS_PROFILE user1
```

使用 `set` 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。

使用 `setx` 设置环境变量会更改运行命令后创建的所有命令 Shell 中的值。这不会 影响运行命令时已在运行的任何命令 Shell。关闭并重新启动命令 Shell 可查看这一更改的效果。

环境变量

环境变量提供了另一种指定配置选项和凭证的方法；若要编写脚本或将一个命名配置文件临时设置为默认配置文件，环境变量会很有用。

选项的优先顺序

- 如果您使用本主题中描述的某个环境变量指定选项，则它将在配置文件中覆盖从配置文件加载的任何值。
- 如果您通过在 CLI 命令行上使用参数指定选项，则它将在配置文件中覆盖相应环境变量或配置文件中的任何值。

支持的环境变量

AWS CLI 支持以下环境变量：

- `AWS_ACCESS_KEY_ID` – 指定与 IAM 用户或角色关联的 AWS 访问密钥。
- `AWS_SECRET_ACCESS_KEY` – 指定与访问密钥关联的私有密钥。这基本上是访问密钥的“密码”。
- `AWS_SESSION_TOKEN` – 指定在使用临时安全凭证时需要的会话令牌值。有关更多信息，请参阅 AWS CLI Command Reference 中的代入角色命令的输出部分。
- `AWS_DEFAULT_REGION` – 指定要将请求发送到的 AWS 区域 (p. 21)。
- `AWS_DEFAULT_OUTPUT` – 指定要使用的输出格式 (p. 57)。
- `AWS_PROFILE` – 指定包含要使用的凭证和选项的 CLI 配置文件 (p. 29)的名称。可以是存储在 `credentials` 或 `config` 文件中的配置文件的名称，也可以是值 `default`，后者使用默认配置文件。如果您指定此环境变量，它将在配置文件中覆盖使用名为 `[default]` 的配置文件的行为。
- `AWS_ROLE_SESSION_NAME` – 指定要与角色会话关联的名称。有关更多信息，请参阅指定角色会话名称以便于审核 (p. 40)。
- `AWS_CA_BUNDLE` – 指定要用于 HTTPS 证书验证的证书捆绑包的路径。
- `AWS_SHARED_CREDENTIALS_FILE` – 指定 AWS CLI 用于存储访问密钥的文件的位置。默认路径为 `~/.aws/credentials`。
- `AWS_CONFIG_FILE` – 指定 AWS CLI 用于存储配置文件的文件的位置。默认路径为 `~/.aws/config`。

下面的示例介绍如何为默认用户配置环境变量。这些值将覆盖在命名配置文件中找到的任何值或实例元数据。设置后，您可以通过在 CLI 命令行上指定参数或通过更改/删除环境变量来覆盖这些值。有关优先顺序以及 AWS CLI 如何确定使用哪些凭证的更多信息，请参阅配置设置和优先顺序 (p. 22)。

Linux, OS X, or Unix

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

设置环境变量会更改使用的值，直到 Shell 会话结束或直到您将该变量设置为其他值。通过在 shell 的启动脚本中设置变量，可使变量在未来的会话中继续有效。

Windows 命令提示符

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

```
C:\> setx AWS_DEFAULT_REGION us-west-2
```

使用 `set` 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。使用 `setx` 设置环境变量会更改当前命令提示符会话和运行该命令后创建的所有命令提示符会话中使用的值。它不 影响在运行该命令时已经运行的其他命令 shell。

PowerShell

```
PS C:\> $Env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
PS C:\> $Env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
PS C:\> $Env:AWS_DEFAULT_REGION="us-west-2"
```

如果在 PowerShell 提示符下设置环境变量（如前面的示例所示），则仅保存当前会话持续时间的值。要在所有 PowerShell 和命令提示符会话中使环境变量设置保持不变，请使用控制面板中的系统应用程序来存储该变量。或者，您可以通过将其添加到 PowerShell 配置文件来为将来的所有 PowerShell 会话设置该变量。有关存储环境变量或跨会话保存它们的更多信息，请参阅 PowerShell 文档。

命令行选项

您可以使用以下命令行选项来覆盖一条命令的默认配置设置。虽然您可以指定要使用的配置文件，但无法使用命令行选项直接指定凭证。

--profile ***<string>***

指定用于该命令的命名配置文件 (p. 29)。要设置其他命名配置文件，可以在 `aws configure` 命令中使用 `--profile` 选项。

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

--region ***<string>***

指定要将该命令的 AWS 请求发送到的 AWS 区域。有关可以指定的所有区域的列表，请参阅Amazon Web Services 一般参考中的 AWS 区域和终端节点。

--output ***<string>***

指定用于该命令的输出格式。您可以指定以下任意值：

- **`json`**：输出采用 JSON 字符串的格式。
- **`text`**：输出采用多行制表符分隔的字符串值的格式，如果要将输出传递给文本处理器（如 `grep`、`sed` 或 `awk`），则该格式非常有用。
- **`table`**：输出采用表格形式，使用字符 +|- 以形成单元格边框。它通常以“人性化”格式呈现信息，这种格式比其他格式更容易阅读，但从编程方面来讲不是那么有用。

--endpoint-url ***<string>***

指定要将请求发送到的 URL。对于大多数命令，AWS CLI 会根据所选服务和指定的 AWS 区域自动确定 URL。但是，某些命令需要您指定账户专用 URL。您还可以配置一些 AWS 服务直接在您的私有 VPC 中托管终端节点（然后可能需要指定该终端节点）。

有关每个区域可用的标准服务终端节点的列表，请参阅Amazon Web Services 一般参考中的 AWS 区域和终端节点。

--debug

指定要启用调试日志记录的布尔开关。这包括有关命令操作的额外诊断信息，这些信息在排查命令提供意外结果的原因时非常有用。

--no-paginate

禁用输出自动分页的布尔开关。

--query ***<string>***

指定用于筛选响应数据的 JMESPath 查询。有关更多信息，请参阅 如何使用 --query 选项筛选输出 (p. 61)。

--version

显示正在运行的 AWS CLI 程序的当前版本的布尔开关。

--color ***<string>***

指定对彩色输出的支持。有效值包括 on、off 和 auto。默认值为 auto。

--no-sign-request

对 AWS 服务终端节点的 HTTP 请求禁用签名的布尔开关。这可避免加载凭证。

--ca-bundle ***<string>***

指定验证 SSL 证书时要使用的 CA 证书捆绑包。

--cli-read-timeout ***<integer>***

指定最大套接字读取时间（以秒为单位）。如果该值设置为 0，则套接字读取将无限等待（阻塞），不会超时。

--cli-connect-timeout ***<integer>***

指定最大套接字连接时间（以秒为单位）。如果该值设置为 0，则套接字连接将无限等待（阻塞），不会超时。

将这些选项中的一个或多个作为命令行参数提供时，它会覆盖该单个命令的默认配置或任何相应的配置文件设置。

每个带参数的选项都需要一个空格或等号 (=) 将参数与选项名称分开。如果参数值为包含空格的字符串，则必须使用引号将参数引起来。

常见的命令行选项用法包括在编写脚本时检查多个 AWS 区域中的资源，以及更改输出格式使其易于阅读或使用。例如，如果您不确定实例运行的区域，可以针对每个区域运行 describe-instances 命令，直到找到该区域，如下所示。

```
$ 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                                       |||
```

```
...
```

指定参数值 (p. 48)中详细描述了每个命令行选项的参数类型（例如，字符串、布尔值）。

使用外部进程获取凭证

> Warning
>
> 以下主题讨论从外部进程获取凭证。如果生成凭证的命令可由未经批准的进程或用户访问，则可能存在安全风险。我们建议您使用 CLI 和 AWS 提供的支持的安全替代方案，以降低泄露凭证的风险。请务必保管好 `config` 文件及任何支持文件和工具，以防泄露。

如果您有 AWS CLI 不直接支持的生成或查找凭证的方法，则可以通过在 `config` 文件中配置 `credential_process` 设置来配置 CLI 使用它。

例如，您可以在配置文件中包含类似于以下内容的条目：

```
[profile developer]
credential_process = /opt/bin/awscreds-custom --username helen
```

AWS CLI 完全按照配置文件中指定的方式运行该命令，然后从 STDOUT 读取数据。您指定的命令必须在 STDOUT 上生成符合以下语法的 JSON 输出：

```
{
  "Version": 1,
  "AccessKeyId": "an AWS access key",
  "SecretAccessKey": "your AWS secret access key",
  "SessionToken": "the AWS session token for temporary credentials",
  "Expiration": "ISO8601 timestamp when the credentials expire"
}
```

截至撰写本文之时，`Version` 密钥必须设置为 `1`。随时间推移和该结构的发展，该值可能会增加。

`Expiration` 密钥是采用 ISO8601 格式的时间戳。如果工具的输出中不存在 `Expiration` 键，则 CLI 假定凭证是不刷新的长期凭证。否则，将其视为临时凭证，并通过在其过期前重新运行 `credential_process` 命令来自动刷新凭证。

> Note
>
> AWS CLI 不 缓存外部进程凭据，这一点不同于代入角色凭证。如果需要缓存，则必须在外部进程中实现。

外部进程可以返回非零返回代码，以指示在检索凭证时发生错误。

实例元数据

从 Amazon EC2 实例中运行 AWS CLI 时，可以简化向命令提供凭证的过程。每个 Amazon EC2 实例都包含 AWS CLI 能够直接查询临时凭证的元数据。要提供这些元数据，请创建一个对所需资源有访问权限的 AWS Identity and Access Management (IAM) 角色，然后在 Amazon EC2 实例启动时向其附加该角色。

启动实例并进行检查，看是否已安装了 AWS CLI（在 Amazon Linux 上是预安装的）。如有必要，安装 AWS CLI。您仍必须配置默认区域，以免在每个命令中指定它。

要在命名配置文件中指定要使用托管 Amazon EC2 实例配置文件中可用的凭证，请在配置文件中指定以下行：

```
credential_source = Ec2InstanceMetadata
```

以下示例说明如何通过在 Amazon EC2 实例配置文件中引用 `marketingadminrole` 角色来代入该角色：

```
[profile marketingadmin]
role_arn = arn:aws-cn:iam::123456789012:role/marketingadminrole
credential_source = Ec2InstanceMetadata
```

您可以通过运行 `aws configure` 来设置区域和默认输出格式，而无需通过按两次 Enter 跳过前两条提示来指定凭证。

```
$ 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
```

向实例附加 IAM 角色后，AWS CLI 可以自动并且安全地从实例元数据检索凭证。有关更多信息，请参阅 IAM 用户指南 中的向 Amazon EC2 实例中运行的应用程序授予访问 AWS 资源的权限。

使用 HTTP 代理

要通过代理服务器访问 AWS，您可以使用代理服务器使用的 DNS 域名或 IP 地址和端口号配置 `HTTP_PROXY` 和 `HTTPS_PROXY` 环境变量。

> **Note**
>
> 以下示例显示了全部使用大写字母的环境变量名称。但是，如果您指定一个变量两次 - 一次使用大写字母，一次使用小写字母，则以使用小写字母的变量为准。我们建议您只定义变量一次，以避免混淆和意外行为。

以下示例显示如何使用代理的显式 IP 地址或解析为代理 IP 地址的 DNS 名称。两种情况都可以后跟冒号和应将查询发送到的端口号。

Linux, OS X, or Unix

```
$ export HTTP_PROXY=http://10.15.20.25:1234
$ export HTTP_PROXY=http://proxy.example.com:1234
$ export HTTPS_PROXY=http://10.15.20.25:5678
$ export HTTPS_PROXY=http://proxy.example.com:5678
```

Windows

```
C:\> setx HTTP_PROXY http://10.15.20.25:1234
C:\> setx HTTP_PROXY=http://proxy.example.com:1234
C:\> setx HTTPS_PROXY=http://10.15.20.25:5678
C:\> setx HTTPS_PROXY=http://proxy.example.com:5678
```

代理身份验证

AWS CLI 支持 HTTP 基本身份验证。在代理 URL 中指定用户名和密码，如下所示：

Linux, OS X, or Unix

```
$ export HTTP_PROXY=http://username:password@proxy.example.com:1234
```

```
$ export HTTPS_PROXY=http://username:password@proxy.example.com:5678
```

Windows

```
C:\> setx HTTP_PROXY http://username:password@proxy.example.com:1234
C:\> setx HTTPS_PROXY=http://username:password@proxy.example.com:5678
```

Note

AWS CLI 不支持 NTLM 代理。如果使用 NTLM 或 Kerberos 协议代理，则可以通过身份验证代理（如 Cntlm）进行连接。

对 Amazon EC2 实例使用代理

如果是在使用附加 IAM 角色启动的 Amazon EC2 实例上配置代理，请确保排除用于访问实例元数据的地址。为此，请将 `NO_PROXY` 环境变量设置为实例元数据服务的 IP 地址 169.254.169.254。该地址保持不变。

Linux, OS X, or Unix

```
$ export NO_PROXY=169.254.169.254
```

Windows

```
C:\> setx NO_PROXY 169.254.169.254
```

在 AWS CLI 中使用 IAM 角色

AWS Identity and Access Management (IAM) 角色是一种授权工具，可让 IAM 用户获得额外（或不同）的权限或者获取使用其他 AWS 账户执行操作的权限。

通过在 `~/.aws/credentials` 文件中为 IAM 角色定义配置文件，您可以配置 AWS Command Line Interface (AWS CLI) 以使用该角色。

以下示例显示了一个名为 `marketingadmin` 的角色配置文件。如果使用 `--profile marketingadmin`（或使用 AWS_PROFILE 环境变量 (p. 31)指定它）运行命令，则 CLI 使用配置文件 `user1` 中定义的凭证代入 Amazon 资源名称 (ARN) 为 `arn:aws-cn:iam::123456789012:role/marketingadminrole` 的角色。您可以运行分配给该角色的权限所允许的任何操作。

```
[marketingadmin]
role_arn = arn:aws-cn:iam::123456789012:role/marketingadminrole
source_profile = user1
```

然后，您可以指定一个指向单独的命名配置文件的 `source_profile`，此配置文件包含 IAM 用户凭证及使用该角色的权限。在上一个示例中，`marketingadmin` 配置文件使用 `user1` 配置文件中的凭证。当您指定某个 AWS CLI 命令将使用配置文件 `marketingadmin` 时，CLI 会自动查找链接的 `user1` 配置文件的凭证，并使用它们为指定的 IAM 角色请求临时凭证。CLI 在后台使用 sts:AssumeRole 操作来完成该操作。然后，使用这些临时凭证来运行请求的 CLI 命令。指定的角色必须附加有允许运行请求的 CLI 命令的 IAM 权限策略。

如果要在 Amazon EC2 实例或 Amazon ECS 容器中运行 CLI 命令，可以使用附加到实例配置文件或容器的 IAM 角色。如果未指定配置文件或未设置环境变量，则将直接使用该角色。这让您能够避免在实例上存储长时间生存的访问密钥。您也可以使用这些实例或容器角色仅获取其他角色的凭证。为此，请使用

`credential_source`（而不是 `source_profile`）指定如何查找凭证。`credential_source` 属性支持以下值：

- `Environment` – 从环境变量检索源凭证。
- `Ec2InstanceMetadata` – 使用附加到 Amazon EC2 实例配置文件的 IAM 角色。
- `EcsContainer` – 使用附加到 Amazon ECS 容器的 IAM 角色。

以下示例显示通过引用 Amazon EC2 实例配置文件来使用同一个 `marketingadminrole` 角色：

```
[profile marketingadmin]
role_arn = arn:aws-cn:iam::123456789012:role/marketingadminrole
credential_source = Ec2InstanceMetadata
```

当您调用角色时，您可以要求其他选项，例如使用多重身份验证、外部 ID（供第三方公司用于访问其客户的资源）以及指定可更容易地在 AWS CloudTrail 日志中进行审核的唯一角色会话名称。

小节目录

配置和使用角色

在使用指定 IAM 角色的配置文件运行命令时，AWS CLI 将使用源配置文件的凭证调用 AWS Security Token Service (AWS STS) 并为指定角色请求临时凭证。源配置文件中的用户必须具有为指定配置文件中的角色调用 `sts:assume-role` 的权限。该角色必须具有允许源配置文件中的用户使用角色的信任关系。检索角色的临时凭证然后使用临时凭证的过程通常称为代入角色。

您可以通过执行 AWS Identity and Access Management 用户指南 中的创建向 IAM 用户委派权限的角色下的过程，在 IAM 中创建一个您希望用户代入的具有该权限的新角色。如果该角色和源配置文件的 IAM 用户在同一个账户中，在配置角色的信任关系时，您可以输入自己的账户 ID。

在创建角色后，请修改信任关系以允许 IAM 用户（或 AWS 账户中的用户）代入该角色。

以下示例显示了一个可附加到角色的信任策略。该策略允许账户 123456789012 中的任何 IAM 用户代入该角色，前提 是该账户的管理员向该用户显式授予了 `sts:assumerole` 权限。

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

信任策略不会实际授予权限。账户管理员必须通过附加具有适当权限的策略才能将代入角色的权限委派给各个用户。以下示例显示了一个可附加到 IAM 用户的策略，该策略仅允许用户代入 `marketingadminrole`

角色。有关授予用户代入角色的访问权限的更多信息，请参阅 IAM 用户指南 中的向用户授予切换角色的权限。

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

IAM 用户无需拥有任何附加权限即可使用角色配置文件运行 CLI 命令。相反，运行命令的权限来自附加到角色 的权限。您可以将权限策略附加到角色，以指定可以针对哪些 AWS 资源执行哪些操作。有关向角色附加权限（与向 IAM 用户附加权限的操作相同）的更多信息，请参阅 IAM 用户指南 中的更改 IAM 用户的权限。

您已正确配置角色配置文件、角色权限、角色信任关系和用户权限，可以通过调用 `--profile` 选项在命令行中使用该角色了。例如，下面的命令使用附加到 `marketingadmin` 角色（由本主题开头的示例定义）的权限调用 Amazon S3 `ls` 命令。

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

要对多个调用使用角色，您可以从命令行设置当前会话的 `AWS_DEFAULT_PROFILE` 环境变量。定义该环境变量后，就不必对每个命令都指定 `--profile` 选项。

Linux, OS X, or Unix

```
$ export AWS_PROFILE=marketingadmin
```

Windows

```
C:\> setx AWS_PROFILE marketingadmin
```

有关配置 IAM 用户和角色的更多信息，请参阅 IAM 用户指南 中的用户和组和角色。

使用多重验证

为了提高安全性，当用户尝试使用角色配置文件进行调用时，您可以要求用户提供从多重验证 (MFA) 设备（一种 U2F 设备）或移动应用程序生成的一次性密钥。

首先，您可以选择将与 IAM 角色有关的信任关系修改为需要 MFA。这可以防止任何人在未首先使用 MFA 进行身份验证的情况下使用该角色。有关示例，请参阅下面示例中的 `Condition` 行。此策略允许名为 `anika` 的 IAM 用户代入策略所附加的角色，但前提是她使用 MFA 进行身份验证。

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

```
  ]
}
```

其次，为角色配置文件添加一行，用来指定用户的 MFA 设备的 ARN。以下示例 config 文件条目显示两个角色配置文件，它们都使用访问密钥为 IAM 用户 anika 请求角色 cli-role 的临时凭证。用户 anika 有权代入角色，这是由角色的信任策略授予的。

```
[profile role-without-mfa]
region = us-west-2
role_arn= arn:aws:iam::128716708097:role/cli-role
source_profile=cli-user

[profile role-with-mfa]
region = us-west-2
role_arn= arn:aws:iam::128716708097:role/cli-role
source_profile = cli-user
mfa_serial = arn:aws:iam::128716708097:mfa/cli-user

[profile anika]
region = us-west-2
output = json
```

该 mfa_serial 设置可以采取如图所示的 ARN 或硬件 MFA 令牌的序列号。

第一个配置文件 role-without-mfa 不需要 MFA。但是，由于附加到角色的先前示例信任策略需要 MFA，因此使用此配置文件运行命令的任何尝试都将失败。

```
$ aws iam list-users --profile role-without-mfa

An error occurred (AccessDenied) when calling the AssumeRole operation: Access denied
```

第二个配置文件条目 role-with-mfa 标识要使用的 MFA 设备。当用户尝试使用此配置文件运行 CLI 命令时，CLI 会提示用户输入 MFA 设备提供的一次性密码 (OTP)。如果 MFA 身份验证成功，则此命令会执行请求的操作。OTP 未显示在屏幕上。

```
$ aws iam list-users --profile role-with-mfa
Enter MFA code for arn:aws:iam::123456789012:mfa/cli-user:
{
    "Users": [
        {
            ...
```

跨账户角色和外部 ID

通过将角色配置为跨账户角色，您可以让 IAM 用户使用属于不同账户的角色。在创建角色期间，将角色类型设置为 Another AWS account (其他 AWS 账户) (如创建向 IAM 用户委派权限的角色中所述) 。 (可选) 选择 Require MFA (需要 MFA)。Require MFA (需要 MFA) 选项将按照使用多重验证 (p. 38)中所述在信任关系中配置相应条件。

如果使用外部 ID 来加强控制可跨账户使用角色的人员，则还必须将 external_id 参数添加到角色配置文件。通常情况下，仅应在其他账户由公司或组织外部的人员控制时才使用该功能。

```
[profile crossaccountrole]
role_arn = arn:aws-cn:iam::234567890123:role/SomeRole
source_profile = default
mfa_serial = arn:aws-cn:iam::123456789012:mfa/saanvi
external_id = 123456
```

指定角色会话名称以便于审核

当某个角色被许多人共享时，审核会变得比较困难。您希望将调用的每个操作与调用该操作的个人关联。但是，当个人使用角色时，个人代入角色是一项独立于调用操作的行为，您必须手动将这两者关联起来。

通过在用户代入角色时指定唯一的角色会话名称，您可以简化此过程。只需向指定某一角色的 `config` 文件中的每个命名配置文件添加 `role_session_name` 参数，即可实现这一点。`role_session_name` 值将传递给 `AssumeRole` 操作，并成为角色会话 ARN 的一部分。该值也包含在所有已记录操作的 AWS CloudTrail 日志中。

例如，您可以创建基于角色的配置文件，如下所示：

```
[profile namedsessionrole]
role_arn = arn:aws-cn:iam::234567890123:role/SomeRole
source_profile = default
role_session_name = Session_Maria_Garcia
```

这会导致角色会话具有以下 ARN：

```
arn:aws-cn:iam::234567890123:assumed-role/SomeRole/Session_Maria_Garcia
```

此外，所有 AWS CloudTrail 日志都在为每个操作捕获的信息中包含角色会话名称。

通过 Web 身份代入角色

您可以配置一个配置文件，以指示 AWS CLI 应使用 Web 联合身份验证和 Open ID Connect (OIDC) 代入角色。当您在配置文件中指定此选项时，AWS CLI 会自动为您发出相应的 AWS STS `AssumeRoleWithWebIdentity` 调用。

> **Note**
>
> 当您指定使用 IAM 角色的配置文件时，AWS CLI 会发出相应的调用来检索临时凭证。随后，这些凭证将存储在 `~/.aws/cli/cache` 中。指定同一个配置文件的后续 AWS CLI 命令将使用缓存的临时凭证，直到它们过期。这时，AWS CLI 将自动刷新这些凭证。

要通过 Web 联合身份验证检索和使用临时凭证，您可以在共享配置文件中指定以下配置值：

role_arn (p. 36)

指定要代入的角色的 ARN。

web_identity_token_file

指定一个文件的路径，该文件包含由身份提供商提供的 OAuth 2.0 访问令牌或 OpenID Connect ID 令牌。AWS CLI 加载此文件，并将其内容作为 `AssumeRoleWithWebIdentity` 操作的 `WebIdentityToken` 参数传递。

role_session_name (p. 40)

指定应用于此代入角色会话的可选名称。

以下是使用 Web 身份配置文件配置代入角色所需的最少量配置的示例配置：

```
# In ~/.aws/config

[profile web-identity]
role_arn=arn:aws:iam:123456789012:role/RoleNameToAssume
web_identity_token_file=/path/to/a/token
```

您也可以使用环境变量 (p. 31)提供此配置：

AWS_ROLE_ARN

要代入的角色的 ARN。

AWS_WEB_IDENTITY_TOKEN_FILE

Web 身份令牌文件的路径。

AWS_ROLE_SESSION_NAME

应用于此代入角色会话的名称。

注意

这些环境变量当前仅适用于使用 Web 身份提供商的代入角色，而不适用于常规代入角色提供商配置。

清除缓存凭证

当您使用一个角色时，AWS CLI 会在本地缓存临时凭证，直到这些凭证过期。当您下次尝试使用它们时，AWS CLI 会尝试代表您续订这些凭证。

如果您的角色的临时凭证已吊销，它们不会自动续订，并且使用它们的尝试将失败。但是，您可以删除缓存以强制 AWS CLI 检索新凭证。

Linux, OS X, or Unix

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

Windows

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

命令完成

在类 Unix 系统上，AWS CLI 包含一项命令完成功能，让您可以使用 Tab 键完成部分键入的命令。在大多数系统上，该功能不是自动安装的，需要手动配置。

要配置命令完成，您必须具有两项信息：所使用的 Shell 的名称和 `aws_completer` 脚本的位置。

Amazon Linux

默认情况下，在运行 Amazon Linux 的 Amazon EC2 实例上自动配置和启用命令完成。

小节目录

识别 Shell

如果不确定所使用的 Shell，可以使用以下命令之一进行识别：

echo $SHELL – 显示 Shell 的程序文件名称。这通常会与所使用的 Shell 的名称匹配，除非您在登录后启动了不同的 Shell。

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

ps – 显示为当前用户运行的进程。Shell 将是其中之一。

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

定位 AWS 完成标签

AWS 完成标签的位置可能随所用安装方法而异。

程序包管理器 – `pip`、`yum`、`brew` 和 `apt-get` 等程序通常在标准路径位置安装 AWS 完成标签（或其符号链接）。在这种情况下，`which` 命令可以为您定位完成标签。

如果在没有 `--user` 命令的情况下使用 `pip`，则可能会看到以下路径。

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

如果您在 pip install 命令中使用了 `--user` 参数，则通常可以在 `$HOME` 文件夹下的 `local/bin` 文件夹中找到完成标签。

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

捆绑安装程序 – 如果根据上一节中的说明使用捆绑安装程序，AWS 完成标签将位于安装目录的 `bin` 子文件夹中。

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

如果所有 else 都失败，可以使用 `find` 在整个文件系统中搜索 AWS 完成标签。

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

将补全程序的文件夹添加到您的路径中

要让 AWS 补全程序成功运行，必须先将其添加到计算机的路径中。

1. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 echo $SHELL。

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

- Bash– .bash_profile、.profile 或 .bash_login
- Zsh– .zshrc
- Tcsh– .tcshrc、.cshrc 或 .login

2. 在配置文件脚本末尾添加与以下示例类似的导出命令。将 */usr/local/aws/bin* 替换为您在上一部分中找到的文件夹。

```
export PATH=/usr/local/aws/bin:$PATH
```

3. 将配置文件重新加载到当前会话中，以使更改生效。将 *.bash_profile* 替换为您在第一部分中找到的 shell 脚本的名称。

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

启用命令完成

运行命令以启用命令完成。用来启用完成功能的命令取决于所使用的 Shell。您可以将命令添加到外壳程序的 RC 文件中，以便在每次打开一个新外壳程序时运行它。在每个命令中，将路径 /usr/local/aws/bin 替换为上一部分中在您的系统上找到的那个。

- **bash** – 使用内置命令 complete。

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

  将命令添加到 ~/.bashrc 中，以便在每次打开一个新外壳程序时运行它。您的 ~/.bash_profile 应指定 ~/.bashrc 的来源，以确保该命令也在登录外壳程序中运行。
- **tcsh** – tcsh 的完成采用字类型和样式来定义完成行为。

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

  将命令添加到 ~/.tschrc 中，以便在每次打开一个新外壳程序时运行它。
- **zsh** – 源 bin/aws_zsh_completer.sh。

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

  AWS CLI 使用 bash 兼容性自动完成 (bashcompinit) 实现 zsh 支持。有关更多详细信息，请参阅aws_zsh_completer.sh 的顶部。

  将命令添加到 ~/.zshrc 中，以便在每次打开一个新外壳程序时运行它。

测试命令完成

启用命令完成后，输入部分命令并按 Tab 查看可用命令。

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

使用 AWS CLI

本部分将介绍 AWS Command Line Interface (AWS CLI) 中提供的许多常见功能和选项。

Note

默认情况下，AWS CLI 通过在 TCP 端口 443 上使用 HTTPS，将请求发送到 AWS 服务。要成功使用 AWS CLI，您必须能够在 TCP 端口 443 上建立出站连接。

主题

使用 AWS CLI 获取帮助

使用 AWS Command Line Interface (AWS CLI) 时，您可以获得任何命令的帮助。为此，只需在命令名称末尾键入 `help`。

例如，以下命令显示常规 AWS CLI 选项和可用顶层命令的帮助。

```
$ aws help
```

以下命令显示可用的特定于 Amazon Elastic Compute Cloud (Amazon EC2) 的命令。

```
$ aws ec2 help
```

以下示例显示 Amazon EC2 `DescribeInstances` 操作的详细帮助。帮助包括对其输入参数、可用筛选条件以及作为输出包含的内容的描述。它还包含说明如何键入命令的常见变体的示例。

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

每个命令的帮助分为六个部分：

名称

命令的名称。

```
NAME
```

```
       describe-instances -
```

描述

命令调用的 API 操作的描述。

```
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.
...
```

摘要

使用命令及其选项的基本语法。如果某个选项显示在方括号中，则表示该选项是可选的、具有默认值或具有可替换使用的替代选项。

```
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]
```

例如，describe-instances 具有描述当前账户和 AWS 区域中的所有 实例的默认行为。您可以选择指定 instance-ids 列表来描述一个或多个实例。dry-run 是不接受值的可选布尔标志。要使用布尔标志，请指定其中一个显示的值，在本例中为 --dry-run 或 --no-dry-run。同样，--generate-cli-skeleton 不使用值。如果某个选项的使用存在条件，则在 OPTIONS 部分中描述这些条件，或在示例中显示这些条件。

选项

对摘要中显示的每个选项的描述。

```
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.
...
```

示例

一些示例，用于显示命令及其选项的使用。如果您需要的命令或用例没有示例可用，请使用本页面上的反馈链接请求一个示例，或在命令的帮助页上的 AWS CLI 命令参考中请求一个示例。

```
    EXAMPLES
    To describe an Amazon EC2 instance

    Command:

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

    To describe all instances with the instance type m1.small

    Command:

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

    To describe all instances with an Owner tag

    Command:

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

输出

来自 AWS 的响应中包含的每个字段和数据类型的描述。

对于 `describe-instances`，输出是预留对象的列表，每个列表都包含若干字段和对象，这些字段和对象包含与其关联的实例的相关信息。此信息来自 Amazon EC2 使用的预留数据类型的 API 文档。

```
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.
...
```

当 AWS CLI 将输出呈现为 JSON 时，输出将成为预留对象的数组，类似于以下示例。

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

每个预留对象都包含一些描述预留的字段和一组实例对象，每个实例对象又带有用来描述它的字段（如 `PublicDnsName`）和对象（如 `State`）。

Windows 用户

您可以通过管道 (|) 将 help 命令的输出发送到 `more` 命令以便每次查看一页帮助文件。按空格键或 PgDn 键可查看文档的更多内容，按 **`q`** 可退出。

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

AWS CLI 文档

AWS CLI Command Reference 还包含所有 AWS CLI 命令的帮助内容。将显示这些说明以方便在手机、平板电脑或桌面屏幕进行浏览和查看。

Note

帮助文件包含无法通过命令行查看或导航至的链接。您可以使用在线AWS CLI Command Reference查看这些链接并与其交互。

API 文档

AWS CLI 中的所有命令对应于对 AWS 服务的公用 API 发出的请求。具有公用 API 的每项服务都有一个 API 参考，可从 AWS 文档网站上该服务的主页找到。API 参考的内容因 API 的构造方式以及所用协议而有所不

同。API 参考通常包含有关该 API 支持的操作、发送到该服务和从该服务发送的数据以及该服务可能报告的任何错误状况的详细信息。

API 文档的各部分

- Actions (操作) – 有关每个操作及其参数（包括对长度或内容以及默认值的约束）的详细信息。它列出了此操作可能发生的错误。每个操作对应于 AWS CLI 中的一个子命令。
- Data Types (数据类型) – 有关命令可能需要作为参数或在响应请求时要返回的结构的详细信息。
- Common Parameters (常用参数) – 有关由服务的所有操作共享的参数的详细信息。
- Common Errors (常见错误) – 有关可能由服务的任意操作返回的错误的详细信息。

每个部分的名称和可用性可能根据具体服务而不同。

特定于服务的 CLI

与以前创建单个 AWS CLI 以处理所有服务不同，有些服务还有单独的 CLI。这些特定于服务的 CLI 具有单独的文档，该服务的文档页面包含指向该文档的链接。特定于服务的 CLI 的文档不适用于 AWS CLI。

AWS CLI 中的命令结构

AWS Command Line Interface (AWS CLI) 在命令行上使用多部分结构，各部分必须以如下顺序指定：

1. 对 aws 计划的基本调用。
2. 顶级命令，这通常对应于 AWS CLI 支持的 AWS 服务。
3. 用于指定要执行的操作的子命令。
4. 常规 CLI 选项或操作所需的参数。您可以按任意顺序指定这些项，只要它们跟在前三个部分之后。如果多次指定某个排他参数，则仅应用最后一个值。

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

参数可采用各种类型的输入值，如数字、字符串、列表、映射和 JSON 结构。支持的内容取决于您指定的命令和子命令。

为 AWS CLI 指定参数值

AWS Command Line Interface (AWS CLI) 中使用的很多参数都是简单的字符串或数值，如下面示例中的 `my-key-pair` 密钥对名称。

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

没有任何空格字符的字符串可以用引号引起来。但是，必须将包含一个或多个空格字符的字符串用引号引起来。在 Linux、macOS 或 Unix PowerShell 中使用单引号 (' ')。在 Windows 命令提示符中使用双引号 (" ")，如以下示例所示。

PowerShell，Linux, OS X, or Unix

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

Windows 命令提示符

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

您可以用等号 (=) 而不是空格将参数名称和值分开。这通常仅在参数的值以连字符开头时有必要。

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

主题

通用参数类型

本节介绍一些通用参数类型以及典型的所需格式。如果您不知道如何设置特定命令的参数格式，请在命令名称后键入 **help** 来查看帮助，例如：

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

每个子命令的帮助都介绍了其功能、选项、输出和示例。选项部分包括每个选项的名称和说明，并在括号中给出了选项的参数类型。

String – 字符串参数可以包含 ASCII 字符集中的字母数字字符、符号和空格。包含空格的字符串必须用引号引起来。我们建议您不要使用标准空格字符以外的符号或空格，因为它可能会导致意外结果。

一些字符串参数可接受来自文件的二进制数据。有关示例，请参阅二进制文件 (p. 53)。

Timestamp – 时间戳根据 ISO 8601 标准设置格式。这些有时称为“DateTime”或“Date”参数。

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

可接受的格式包括：

- *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss.sssTZD (UTC)*，例如，2014-10-01T20:30:00.000Z
- *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss.sssTZD######*，例如，2014-10-01T12:30:00.000-08:00
- *YYYY*-*MM*-*DD*，例如，2014-10-01
- 以秒为单位的 Unix 时间，例如 1412195400。这有时称为 Unix 纪元时间，表示自 1970 年 1 月 1 日午夜 (UTC) 以来经历的秒数。

List – 以空格分隔的一个或多个字符串。如果任何字符串项目包含空格，则必须用引号括起该项目。

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

Boolean – 打开或关闭选项的二进制标志。例如，`ec2 describe-spot-price-history` 有一个布尔 `--dry-run` 参数，如果指定该参数，则针对服务验证查询而不实际运行查询。

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

输出指示命令格式是否正确。此命令还包含一个 `--no-dry-run` 参数版本，可以用来显式指示命令应正常运行。不过不是必须包含此参数，因为这是默认行为。

Integer – 无符号整数。

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

Blob – 二进制对象。Blob 参数采用包含二进制数据的本地文件的路径。此路径不应包含任何协议标识符，例如 `http://` 或 `file://`。指定的路径被解释为相对于当前工作目录。

例如，适用于 `aws s3api put-object` 的 `--body` 参数是一个 blob。

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

Map – 使用 JSON 或 CLI 的速记语法 (p. 65)指定的一系列密钥值对。以下 JSON 示例使用 map 参数 `--key` 从名为 my-table 的 Amazon DynamoDB 表中读取项目。此参数在嵌套的 JSON 结构中指定名为 id 且数值为 1 的主键。

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

对参数使用 JSON

JSON 对于指定复杂的命令行参数非常有用。例如，下面的命令列出实例类型为 `m1.small` 或 `m1.medium` 并在 `us-west-2c` 可用区中的所有 Amazon EC2 实例。

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

或者，您可以将筛选条件的等效列表指定为 JSON 数组。方括号用于创建以逗号分隔的 JSON 对象的数组。每个对象均为以逗号分隔开的密钥值对列表（在此实例中，“Name”和“Values”都是密钥）。

“Values”密钥右侧的值本身就是数组。即使数组只包含一个值字符串，也是必需的。

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

但是，仅在指定一个以上的筛选器时需要最外层的括号。上一个命令的 JSON 格式的单个筛选器版本如下所示。

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

对于某些操作，您必须将数据格式设置为 JSON。例如，要向 `--block-device-mappings` 命令中的 `ec2 run-instances` 参数传递参数，您需要将块储存设备信息的格式设置为 JSON。

此示例显示的 JSON 指定一个在启动实例时在 `/dev/sdb` 中映射的 20 GiB Amazon Elastic Block Store (Amazon EBS) 设备。

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

要连接多个设备，请在数组中列出对象，如下一个示例中所示。

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

您可以直接在命令行输入 JSON（请参阅将引号和字符串结合使用 (p. 51)），也可以将它保存为一个从命令行引用的文件（请参阅从文件加载参数 (p. 52)）。

当传入大块数据时，先将 JSON 保存为一个文件，然后从命令行引用它可能更为简单。文件中的 JSON 数据更容易读取、编辑和与他人共享。后面的部分将介绍这一方法。

有关 JSON 的更多信息，请参阅 JSON.org、Wikipedia 的 JSON 条目和 RFC4627 - JSON 的应用程序/json 媒体类型。

将引号和字符串结合使用

在命令行中输入 JSON 格式参数的方式因操作系统而异。

Linux、macOS 或 Unix

使用单引号 (' ') 括住 JSON 数据结构，如下例所示：

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

PowerShell

PowerShell 需要单引号 (' ') 括住 JSON 数据结构，还需要反斜杠 (\) 对 JSON 结构中的每个双引号 (") 进行转义，如下例所示：

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

Windows 命令提示符

Windows 命令提示符要求使用双引号 (" ") 括住 JSON 数据结构。然后，您必须对 JSON 数据结构中的每个双引号 (") 本身进行转义（前面有一个反斜杠 [\] 字符），如下例所示：

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

只有最外层双引号不进行转义。

如果参数值本身是一个 JSON 文档，请对该嵌入 JSON 文档中的引号进行转义。例如，`attribute` 的 `aws sqs create-queue` 参数可以使用 `RedrivePolicy` 键。`--attributes` 参数取值为一个 JSON 文档，而该文档包含 `RedrivePolicy`，后者也将 JSON 文档作为其值。嵌入到外层 JSON 的内层 JSON 必须进行转义。

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

从文件加载参数

有些参数需要文件名作为变量，AWS CLI 将从这些变量中加载数据。其他参数允许您将参数值指定为在命令行上键入的文本或从文件中读取的文本。无论文件是必需还是可选的，该文件都必须正确编码才能被 AWS CLI 理解。该文件的编码必须与读取系统的默认区域设置相匹配。这可以通过使用 Python `locale.getpreferredencoding()` 方法来确定。

Note

默认情况下，Windows PowerShell 将文本输出为 UTF-16，这与许多 Linux 系统使用的 UTF-8 冲突。我们建议您将 `-Encoding ascii` 与 PowerShell `Out-File` 命令一起使用，以确保生成的文件可以被 AWS CLI 读取。

有时，从文件加载参数值（而不是尝试将其全部键入为命令行参数值）很方便，例如当参数为复杂的 JSON 字符串时。要指定包含该值的文件，请按以下格式指定文件 URL：

```
file://complete/path/to/file
```

前两个斜杠“/”字符是规范的一部分。如果所需的路径以“/”开头，则结果为三个斜杠字符：`file:///folder/file`。

URL 提供包含实际参数内容的文件的路径。

Note

对于本就要求指定 URI 的参数（例如标识 AWS CloudFormation 模板 URI 的参数），将自动禁用该行为。

您也可以通过将以下行添加到 CLI 配置文件来自行禁用该行为：

```
cli_follow_urlparam = false
```

以下示例中的文件路径被解读为相对于当前工作目录。

Linux, OS X, 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
C:\> aws ec2 describe-instances --filters file://C:\temp\filter.json
```

`file://` 前缀选项支持包含“`~/`”、“`./`”和“`../`”的 Unix 式扩展。在 Windows 上，“`~/`”表达式将展开到您的用户目录（存储在 `%USERPROFILE%` 环境变量中）。例如，在 Windows 10 上，您通常在 `C:\Users\`*`User Name`*`\` 下有一个用户目录。

作为另一个 JSON 文档的值嵌入的 JSON 文档仍必须进行转义。

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

attributes.json

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

二进制文件

对于将二进制数据用作参数的命令，请使用 `fileb://` 前缀指定该数据为二进制内容。接受二进制数据的命令包括：

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

以下示例使用 Linux 命令行工具生成一个二进制 256 位 AES 密钥，然后将该密钥提供给 Amazon S3 以对上载的文件服务器端进行加密。

```
$ 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\""
}
```

远程文件

AWS CLI 还支持使用 `http://` 或 `https://` URL 从 Internet 上托管的文件中加载参数。下面的示例引用存储在 Amazon S3 存储桶中的一个文件。这将允许您从任何计算机访问参数文件，但它的确要求容器可公开访问。

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

前面的示例假定 `filename.json` 文件包含以下 JSON 数据。

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

有关引用包含更复杂 JSON 格式参数的文件的更多示例，请参阅将 IAM 托管策略附加到 IAM 用户 (p. 87)。

从 JSON 输入文件生成 CLI 骨架和输入参数

大多数 AWS Command Line Interface (AWS CLI) 命令支持使用 `--cli-input-json` 参数接受从文件输入的所有参数的功能。

这些相同的命令帮助提供 `--generate-cli-skeleton` 以使用您可以编辑和填写的所有参数生成文件。然后，您可以带有 `--cli-input-json` 参数运行命令并指向填充的文件。

> **Important**
>
> 有多个 AWS CLI 命令不直接映射到各个 AWS API 操作，例如 `aws s3` 命令。此类命令不支持此页面上讨论的 `--generate-cli-skeleton` 或 `--cli-input-json` 参数。如果您对特定命令是否支持这些参数有任何疑问，请运行以下命令，将 *service* 和 *command* 名称替换为您感兴趣的名称：
>
> ```
> $ aws service command help
> ```
>
> 输出包含 `Synopsis` 部分，其中显示了指定的命令支持的参数。

`--generate-cli-skeleton` 参数将导致命令无法运行，而是生成和显示您可以自定义的参数模板，然后用作以后命令的输入。生成的模板包含命令支持的所有参数。

例如，如果您运行以下命令，它会为 Amazon Elastic Compute Cloud (Amazon EC2) 命令 run-instances 生成参数模板。

```
$ 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
}
```

生成并使用参数骨架文件

1. 运行带有 `--generate-cli-skeleton` 参数的命令，并将输出定向到某一文件以保存它。

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

2. 在文本编辑器中打开参数骨架文件，并删除任何不需要的参数。例如，您可以将其缩减到以下内容。

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

   在本示例中，我们将 `DryRun` 参数保留设置为 true 以使用 EC2 的空运行功能，这让您可以安全地测试命令而不实际创建或修改任何资源。

3. 使用适合您的场景的值填入剩余的值。在本例中，我们提供要使用的 AMI 的实例类型、密钥名称、安全组和标识符。此示例假定默认区域。AMI `ami-dfc39aef` 是 `us-west-2` 区域中的 64 位 Amazon Linux 映像。如果您使用不同的区域，您必须查找正确 AMI ID 来使用。

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

4. 通过使用 `file://` 前缀将 JSON 文件传递到 `--cli-input-json` 参数，使用填写的参数运行命令。AWS CLI 将路径解释为相对于当前工作目录，因此，将直接在当前工作目录中查找以下仅显示文件名而不带路径的示例。

```
$ 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.
```

   空运行错误表明，JSON 格式正确且参数值有效。如果输出中报告了任何其他问题，请解决这些问题并重复以上步骤，直至显示“请求应已成功”消息。

5. 现在，您可以将 `DryRun` 参数设置为 `false` 以禁用空运行。

```
{
```

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

6. 现在，当您运行此命令时，`run-instances` 实际上会启动 EC2 实例并显示成功启动所生成的详细信息。

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

控制 AWS CLI 的命令输出

此部分介绍控制 AWS Command Line Interface (AWS CLI) 输出的不同方式。

主题

如何选择输出格式

AWS CLI 支持三种不同的输出格式：

- JSON (`json`)
- 制表符分隔的文本 (`text`)
- ASCII 格式的表 (`table`)

正如配置 (p. 20)主题所述，输出格式可通过三种不同方式指定：

- 在 `config` 文件中的命名配置文件中使用 `output` 选项。以下示例将默认输出格式设置为 `text`。

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

- 使用 `AWS_DEFAULT_OUTPUT` 环境变量。对于此命令行会话中的命令，以下输出将格式设置为 `table`，直到更改此变量或会话结束。使用此环境变量将覆盖在 `config` 文件中设置的任何值。

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

- 在命令行上使用 --output 选项。以下示例仅将这一个命令的输出设置为 json。对此命令使用此选项将覆盖任何当前设置的环境变量或 config 文件中的值。

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

AWS CLI 优先顺序规则 (p. 22)适用。例如，使用 AWS_DEFAULT_OUTPUT 环境变量将覆盖在 config 文件中设置的任何值，使用 --output 传递到 AWS CLI 命令的值将覆盖在环境变量或 config 文件中设置的任何值。

json 选项是通过不同语言或 jq（命令行 JSON 处理器）以编程方式处理输出的最佳选择。

table 格式易于阅读。

text 格式适合在传统 Unix 文本工具（如 sed、grep 和 awk）以及 PowerShell 脚本中使用。

可以使用 --query 参数自定义和筛选任何格式的结果。有关更多信息，请参阅 如何使用 --query 选项筛选输出 (p. 61)。

JSON 输出格式

JSON 是 AWS CLI 的默认输出格式。大多数语言都可以使用内置功能或公开提供的库轻松解码 JSON 字符串。正如上一个主题及输出示例所示，--query 选项可提供强大的方法来筛选和格式化 AWS CLI 的 JSON 格式输出。

如果您需要 --query 无法实现的更高级的功能，可以试试命令行 JSON 处理程序 jq。您可以在以下网址下载它并找到正式的教程：http://stedolan.github.io/jq/。

Text 输出格式

text 格式将 AWS CLI 的输出组织为制表符分隔的行。此格式适合在传统 Unix 文本工具（如 grep、sed 和 awk）以及由 PowerShell 执行的文本处理中使用。

Text 输出格式遵循以下所示的基本结构。这些列根据底层 JSON 对象相应的键名称按字母顺序排序。

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

下面是一个文本输出示例。

```
$ aws ec2 describe-volumes --output text
VOLUMES us-west-2a      2013-09-17T00:55:03.000Z        30      snap-f23ec1c8   in-use
 vol-e11a5288    standard
ATTACHMENTS     2013-09-17T00:55:03.000Z        True    /dev/sda1       i-a071c394
 attached        vol-e11a5288
VOLUMES us-west-2a      2013-09-18T20:26:15.000Z        8       snap-708e8348   in-use
 vol-2e410a47    standard
ATTACHMENTS     2013-09-18T20:26:16.000Z        True    /dev/sda1       i-4b41a37c
 attached        vol-2e410a47
```

Important

我们强烈建议，如果您指定 *text* 输出，则也始终使用 *--query* 选项以确保行为一致。这是因为文本格式按基础 JSON 对象的键名称以字母顺序对输出列进行排序，而类似的资源可能没有相同的键名称。例如，基于 Linux 的 EC2 实例的 JSON 表示形式中的元素在基于 Windows 的实例的 JSON 表示形式中可能不存在，反之亦然。此外，资源可能在未来的更新中添加或移除键/值元

素，从而修改列的顺序。因此，可以使用 --query 补充文本输出的功能，以提供对输出格式的完全控制。在以下示例中，命令指定要显示的元素，并使用列表表示法 [key1, key2, ...] 定义列的顺序。这可让您十分放心：正确的键值始终显示在预期的列中。最后请注意，对于不存在的键，AWS CLI 将输出 None 作为键值。

```
$ 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
```

以下示例显示如何将 grep 和 awk 与来自 aws ec2 describe-instances 命令的 text 输出结合使用。第一个命令在文本输出中显示每个实例的可用区、当前状态和实例 ID。第二个命令处理该输出，以仅输出在 us-west-2a 可用区中运行的所有实例的实例 ID。

```
$ 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
```

以下示例更进一步，不仅说明如何筛选输出，还介绍如何使用该输出自动更改每个已停止实例的实例类型。

```
$ 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
```

文本输出也适用于 PowerShell。因为 text 输出中的列使用制表符分隔，所以可以使用 PowerShell 的 `t 分隔符将其拆分为数组。以下命令在第一列 (AvailabilityZone) 与字符串 us-west-2a 匹配的情况下显示第三列 (InstanceId) 的值。

```
PS C:\>aws ec2 describe-instances --query 'Reservations[*].Instances[*].
[Placement.AvailabilityZone, State.Name, InstanceId]' --output text |
%{if ($_.split("`t")[0] -match "us-west-2a") { $_.split("`t")[2]; } }
i-4b41a37c
i-a071c394
i-3045b007
i-6fc67758
```

Tip

如果使用 --query 参数输出文本并将输出筛选到单个字段，则输出是单行制表符分隔值。要将每个值放到单独的行上，可以将输出字段放在括号中，如以下示例所示：

制表符分隔的单行输出：

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

通过将 [GroupName] 放在括号中，让每个值都在自己的行上：

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

Table 输出格式

table 格式以表格形式生成复杂 AWS CLI 输出的易于人阅读的表示。

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

您可以将 --query 选项与表格格式结合使用，以显示从原始输出中预先选择的一系列元素。请注意，字典和列表表示法之间的输出区别：在第一个示例中，列名按字母顺序排序；在第二个示例中，未命名的列按用户指定的顺序排序。有关 --query 选项的更多信息，请参阅 如何使用 --query 选项筛选输出 (p. 61)。

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

如何使用 --query 选项筛选输出

AWS CLI 使用 --query 选项提供内置的基于 JSON 的输出筛选功能。--query 参数接受符合 JMESPath 规范的字符串。

> **Important**
>
> 指定的输出类型（json、text 或 table）影响 --query 选项的运行方式。
>
> - 如果您指定 --output text，则在应用 --query 筛选条件之前，输出采用分页方式，并且 AWS CLI 会一次性地在输出的每个页面上运行查询。这可能会导致意外的额外输出，特别是如果您的筛选条件使用 [0] 之类的对象指定数组元素，因为输出会包含每个页面上的第一个匹配元素。
> - 如果您指定 --output json，则在应用 --query 筛选条件之前，输出会完全处理并转换为 JSON 结构。AWS CLI 仅针对整个输出一次性地运行查询。
>
> 要处理在使用 --output text 时可能产生额外的输出，您可以指定 --no-paginate。这会导致筛选条件仅应用于整个结果集，但它会删除任何分页，因此可能导致很长的输出。您也可以使用其他命令行工具（例如 head 或 tail），额外地筛选输出以仅显示所需的值。

为了演示 --query 的工作原理，我们首先从以下默认 JSON 输出开始，这些输出描述了两个 Amazon Elastic Block Store (Amazon EBS) 卷，它们附加到不同的 Amazon EC2 实例。

```
$ 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
        }
    ]
}
```

我们可以选择使用以下命令从 Volumes 列表中仅显示第一个卷，该命令为数组中的第一个卷编制索引。

```
$ 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
}
```

在下一个示例中，我们使用通配符表示法 [*] 循环访问列表中的所有卷：VolumeId、AvailabilityZone 和 Size。词典表示法要求您为每个 JSON 键提供一个别名，如：{Alias1:JSONKey1,Alias2:JSONKey2}。词典本身是无序的，因此，此种结构中的键别名的顺序可能不一致。

```
$ 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
    }
]
```

在使用词典表示法时，您也可以将键链接起来（如 key1.key2[0].key3）来筛选深度嵌套在结构中的元素。以下示例利用 Attachments[0].InstanceId 键演示此功能，别名指定为简单的 InstanceId。

```
$ 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
    }
]
```

您也可以使用列表表示法筛选多个元素：[key1, key2]。此表达式会将每个对象筛选出的所有属性放入单个排序列表中，不考虑类型。

```
$ 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
    ]
]
```

要按特定字段的值筛选结果，请使用 JMESPath "?" 运算符。以下示例查询仅输出 us-west-2a 可用区中的卷。

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

Note

在指定诸如以上 JMESPath 查询表达式中的“us-west-2”这样的文字值时，必须将该值放在反引号 (`) 中，以便使它能够正确读取。

下面是一些其他示例，说明如何从命令输出中仅获取所需的详细信息。

以下示例列出了 Amazon EC2 卷。该服务在 us-west-2a 可用区中生成所有正在使用的卷的列表。--query 参数进一步将输出限制为只有 Size 值大于 50 的卷，并且仅显示具有用户定义名称的指定字段。

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

以下示例检索满足多个条件的映像的列表。然后，它使用 --query 参数按 CreationDate 对输出进行排序，从而仅选择最新的。然后，它显示这一个映像的 ImageId。

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

以下示例使用 --query 参数在列表中查找特定的项目，然后提取该项目的的信息。此示例列出了与指定的服务终端节点相关联的所有可用区。它从 ServiceDetails 列表中提取具有指定 ServiceName 的项目，然后输出该选定项目的 AvailabilityZones 字段。

```
$ aws --region us-east-1 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-1.ecs`].AvailabilityZones'
[
    [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
    ]
]
```

--query 参数还允许您计算输出中的项目数量。以下示例显示超过 1000 IOPS 的可用卷数。

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

以下示例显示如何列出在指定日期之后创建的所有快照，从而在输出中仅包括几个可用字段。

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

以下示例列出了您创建的五个最新 AMI，从最新到最旧排序。

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

]
```

以下示例仅显示指定 AutoScaling 组中任何运行状况不佳的实例的 `InstanceId`。

```
$ aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name My-AutoScaling-
Group-Name --output text\
--query 'AutoScalingGroups[*].Instances[?HealthStatus==`Unhealthy`].InstanceId'
```

在以下部分中将更详细地解释如何组合使用这三种输出格式。`--query` 选项是十分强大的工具，您可用它来自定义输出的内容和样式。

有关底层 JSON 处理库 JMESPath 的更多示例和完整规范，请查看 http://jmespath.org/specification.html。

将速记语法与 AWS Command Line Interface结合使用

AWS Command Line Interface (AWS CLI) 可以接受 JSON 格式的许多选项。但是，在命令行上输入较大的 JSON 列表或结构会比较繁琐。为了简化此过程，AWS CLI 还支持一种速记语法，允许采用比完整 JSON 格式更简单的方式表示选项参数。

结构参数

通过 AWS CLI 中的速记语法，用户更容易输入平面（非嵌套结构）参数。格式采用以逗号分隔的键值对列表。

Linux, OS X, or Unix

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

PowerShell

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

这两者均等同于以下采用 JSON 格式的示例。

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

各逗号分隔的键值对之间不能有空格。下面的示例 Amazon DynamoDB`update-table` 命令包含采用快速输入语法指定的 `--provisioned-throughput` 选项。

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

该示例等同于以下 JSON 格式的示例。

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

列出参数

您可以使用两种方法以列表形式指定输入参数：JSON 或速记。使用 AWS CLI 速记语法，可更方便地传入含有数字、字符串或非嵌套结构的列表。

下面显示了基本格式，列表中的值用单个空格分隔。

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

该示例等同于以下 JSON 格式的示例。

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

如前所述，您可以用速记语法指定数字列表、字符串列表或非嵌套结构的列表。以下是用于 Amazon Elastic Compute Cloud (Amazon EC2) 的 `stop-instances` 命令示例，其中，`--instance-ids` 选项的输入参数（字符串列表）采用快速输入语法指定。

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

该示例等同于以下 JSON 格式的示例。

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

下面的示例显示 Amazon EC2 `create-tags` 命令，该命令针对 `--tags` 选项使用非嵌套结构的列表。`--resources` 选项指定要添加标签的实例的 ID。

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

该示例等同于以下 JSON 格式的示例。JSON 参数分多行编写以便于阅读。

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

使用 AWS CLI 分页选项

对于可返回项目的大型列表的命令，AWS Command Line Interface (AWS CLI) 添加了三个选项。当 AWS CLI 调用服务的 API 以填充此列表时，您可使用这三个选项控制输出中包括的项目数。

默认情况下，AWS CLI 使用页面大小 1000 并检索所有可用项目。如果您在包含 3500 个对象的 Amazon S3 存储桶上运行 `aws s3api list-objects`，则 CLI 将对 Amazon S3 进行四次调用，以在后台处理服务特定分页逻辑并在最终输出中返回所有 3500 个对象。

如果您在对大量资源运行列表命令时发现问题，则表明默认页面大小 1000 可能过高。这可能会导致对 AWS 服务的调用超过允许的最大时间并生成“超时”错误。您可以使用 `--page-size` 选项来指定 AWS CLI 从对 AWS 服务的每个调用请求数量较少的项目。CLI 仍将检索完整列表，但会在后台执行大量服务 API 调用，并减少每次调用时检索的项目数。这样，各个调用成功的可能性会更高且不会发生超时。更改页面大小不会影响输出；它只影响生成输出所需进行的 API 调用数。

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

要在 AWS CLI 输出中一次包括更少的项目，请使用 `--max-items` 选项。AWS CLI 仍会按上面所述使用该服务处理分页，但只打印您指定的一次检索的项目数：

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

如果项目输出的数量 (`--max-items`) 少于基础 API 调用所返回的项目总数，则输出将包含您可传递到后续命令的 `NextToken` 以检索下一组项目。以下示例显示如何使用上一示例返回的 `NextToken` 值，并使您能够检索接下来的 100 个项目。

> **Note**
>
> 参数 `--starting-token` 不能为空。如果上一个命令未返回 `NextToken` 值，则没有更多项目可返回，您不需要再次调用该命令。

```
$ aws s3api list-objects --bucket my-bucket --max-items 100 --starting-token
 eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==
{
```

```
    "Contents": [
...
```

每次调用指定的 AWS 服务时返回项目的顺序可能不同。如果您为 `--page-size` 和 `--max-items` 指定不同的值，您可能会获得意外结果（项目缺失或重复）。为防止出现这种情况，请对 `--page-size` 和 `--max-items` 使用相同的数字，以同步 AWS CLI 的分页与基础服务的分页。您还可以检索整个列表并在本地执行任何必需的分页操作。

了解 AWS CLI 的返回代码

要确定 AWS CLI 命令的返回代码，请在运行 CLI 命令后立即运行以下命令之一。

Linux/Unix/Mac 系统

```
$ echo $?
```

Windows PowerShell

```
PS> echo $lastexitcode
```

Windows 命令提示符

```
C:\> echo %errorlevel%
```

以下是运行 AWS Command Line Interface (AWS CLI) 命令结束时可能返回的返回代码值。

代码	意义
0	命令成功完成。AWS CLI 或将请求发送到的 AWS 服务未生成错误。
1	一个或多个 S3 传输操作失败。仅限 s3 命令。
2	该返回代码的含义取决于命令。 • 无法解析在命令行上输入的命令。解析失败的原因可能是（但不限于）缺少必需的子命令或参数，或使用了未知的命令或参数。 适用于所有 CLI 命令。 • 在传输过程中，跳过了标记为要进行传输的一个或多个文件。但是，标记为要进行传输的所有其他文件都已成功传输。传输过程中跳过的文件包括：不存在的文件，字符特殊设备、块特殊设备、FIFO 或套接字的文件，以及用户没有读取权限的文件。 仅限 S3 命令。
130	命令被 SIGINT (Ctrl-C) 中断。
255	命令失败。AWS CLI 或将请求发送到的 AWS 服务生成了错误。

要了解有关故障的更多详情，请使用 `--debug` 开关运行命令。这将生成 AWS CLI 用于处理命令的步骤以及每个步骤的结果的详细报告。

通过 AWS CLI 使用 AWS 服务

本节提供说明如何使用 AWS Command Line Interface (AWS CLI) 访问各种 AWS 服务的示例。

有关对每种服务的所有可用命令的完整参考信息，请参阅AWS CLI Command Reference或者使用内置的命令行帮助。有关更多信息，请参阅使用 AWS CLI 获取帮助 (p. 44)。

主题

将 Amazon DynamoDB 与 AWS CLI 结合使用

AWS Command Line Interface (AWS CLI) 为所有 AWS 数据库服务（包括 Amazon DynamoDB）提供支持。您可以使用 AWS CLI 进行临时操作，如创建表。您还可以使用它在实用工具脚本中嵌入 DynamoDB 操作。

要列出 DynamoDB 的 AWS CLI 命令，请使用以下命令。

```
aws dynamodb help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

命令行格式为 Amazon DynamoDB API 名称后接该 API 的参数。AWS CLI 对于参数值支持 CLI 速记语法 (p. 65)以及完全 JSON。

例如，以下命令可创建一个名为 `MusicCollection` 的表。

> **Note**
>
> 为便于阅读，本节中的长命令分行显示。反斜杠字符是 Linux 命令行的行继续符，可让您将多个行复制并粘贴（或输入）到 Linux 提示符处。如果您使用的是不使用反斜杠进行行继续的 shell，请将反斜杠替换为该 shell 的行继续符。或者删除反斜杠并将整个命令放在一行上。

```
$ 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
```

您可以使用类似下面的命令向表中添加新行，如以下示例所示。这些示例使用速记语法和 JSON 的组合。

```
$ 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"
    }
}
```

可能难以在单行命令中编写有效的 JSON。为了使之更简单，AWS CLI 可以读取 JSON 文件。例如，请考虑以下 JSON 代码段，它存储在一个名为 `expression-attributes.json` 的文件中。

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

您可以使用此文件通过 AWS CLI 发出 `query` 请求。在下面的示例中，`expression-attributes.json` 文件的内容用于 `--expression-attribute-values` 参数。

```
$ 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
}
```

有关将 AWS CLI 与 DynamoDB 结合使用的更多信息，请参阅 AWS CLI Command Reference中的 DynamoDB。

除了 DynamoDB 之外，您可以将 AWS CLI 和 DynamoDB Local 结合使用。DynamoDB Local 是模拟 DynamoDB 服务的小客户端数据库和服务器。通过 DynamoDB Local 可编写使用 DynamoDB API 的应用程序，而无需实际操作 DynamoDB Web 服务中的任何表或数据。而所有 API 操作均重新路由到本地数据库。这样可节省预配置吞吐量、数据存储和数据传输费用。

有关 DynamoDB Local 及如何将它与 AWS CLI 结合使用的更多信息，请参阅 Amazon DynamoDB 开发人员指南 中的以下部分：

- DynamoDB Local
- 结合使用 AWS CLI 和 DynamoDB Local

将 Amazon EC2 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Elastic Compute Cloud (Amazon EC2) 的功能。要列出 Amazon EC2 的 AWS CLI 命令，请使用以下命令。

```
aws ec2 help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

本主题显示执行 Amazon EC2 常见任务的 AWS CLI 命令。

主题

创建、显示和删除 Amazon EC2 密钥对

您可以使用 AWS Command Line Interface (AWS CLI) 创建、显示和删除 Amazon EC2 的密钥对。您可以使用密钥对连接到 Amazon EC2 实例。

在创建实例时您必须向 Amazon EC2 提供密钥对，然后在连接到实例时使用该密钥对进行身份验证。

> Note
>
> 以下示例假定您已配置了默认凭证 (p. 71)。

主题

创建密钥对

要创建密钥对，请使用 create-key-pair 命令以及 `--query` 选项和 `--output text` 选项，以通过管道将私有密钥直接传输到文件。

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

对于 Windows PowerShell，`> file` 重定向会默认采用 UTF-8 编码，这种编码不能用于某些 SSH 客户端。因此，您必须通过管道将输出传输到 `out-file` 命令和显式将编码设置为 `ascii` 来转换输出。

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

生成的 `MyKeyPair.pem` 文件类似于以下内容。

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

您的私有密钥不存储在 AWS 中，并且只有在创建后才能检索。以后您无法恢复它。而如果您丢失了私有密钥，则必须创建新的密钥对。

如果您要从 Linux 计算机连接到您的实例，建议您使用以下命令设置您的私有密钥文件的权限，以确保只有您可以读取该文件。

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

显示密钥对

“指纹”是从密钥对生成的，您可以使用指纹验证您本地计算机上的私有密钥是否与 AWS 中存储的公有密钥匹配。

指纹是取自私有密钥的 DER 编码副本的 SHA1 哈希。在创建密钥对时将捕获此值，此值与公有密钥一起存储在 AWS 中。您可以在 Amazon EC2 控制台中或通过运行 AWS CLI 命令 `aws ec2 describe-key-pairs` 查看指纹。

以下示例显示 `MyKeyPair` 的指纹。

```
$ 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"
        }
    ]
```

```
}
```

有关密钥和指纹的更多信息，请参阅Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 密钥对。

删除您的密钥对

要删除密钥对，请运行以下命令，将 *MyKeyPair* 替换为要删除的密钥对的名称。

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

创建、配置和删除 Amazon EC2 的安全组

您可以为您的 Amazon Elastic Compute Cloud (Amazon EC2) 实例创建安全组，安全组本质上用作一个防火墙，具有可确定哪些网络流量可以进入和离开的规则。您可以创建要在 virtual private cloud (VPC) 或 EC2-Classic 共享扁平化网络中使用的安全组。有关 EC2-Classic 和 EC2-VPC 之间差异的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的支持的平台。

您可以使用 AWS Command Line Interface (AWS CLI) 创建一个安全组，向现有安全组添加规则，以及删除安全组。

> **Note**
>
> 下面所示的示例假定您已配置了默认凭证 (p. 71)。

主题

正在创建安全组

您可以创建与 VPC 或 EC2-Classic 关联的安全组。

EC2-VPC

以下示例说明如何为指定的 VPC 创建安全组。

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

要查看安全组的初始信息，请运行 describe-security-groups 命令。您不能仅通过其 `vpc-id` 而非其名称引用 EC2-VPC 安全组。

```
$ 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

以下示例说明如何为 EC2-Classic 创建安全组。

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

要查看 my-sg 的初始信息，请运行 describe-security-groups 命令。对于 EC2-Classic 安全组，您可以通过其名称进行引用。

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

为安全组添加规则

当您运行 Amazon EC2 实例时，您必须启用安全组中的规则，以便为连接到映像的方法启用入站网络流量。

例如，如果您要启动 Windows 实例，您通常会添加规则以允许 TCP 端口 3389 (RDP) 上的入站流量，从而支持远程桌面协议 (RDP)。如果您要启动 Linux 实例，则通常会添加规则以允许 TCP 端口 22 上的入站流量，从而支持 SSH 连接。

使用 authorize-security-group-ingress 命令向安全组添加规则。此命令的必需参数是您计算机的公有 IP 地址，或您的计算机连接到的网络（以地址范围形式）（采用 CIDR 表示法）。

EC2-VPC

以下示例说明如何将适用于 RDP 的规则（TCP 端口 3389）添加到 ID 为 sg-903004f8 的 EC2-VPC 安全组。此示例假定客户端计算机在 CIDR 范围 203.0.113.0/24 中的某个位置有一个地址。

您可以首先确认您的公有地址显示为包含在 CIDR 范围 203.0.113.0/24 中。

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

确认该信息后，您可以通过运行 authorize-security-group-ingress 命令将此范围添加到您的安全组。

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

以下命令添加另一个规则以对同一安全组中的实例启用 SSH。

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

要查看对安全组所做的更改，请运行 describe-security-groups 命令。

```
$ 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

以下命令将适用于 RDP 的规则添加到名为 *my-sg* 的 EC2-Classic 安全组。

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

以下命令将适用于 SSH 的另一条规则添加到同一个安全组。

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

要查看对安全组所做的更改，请运行 describe-security-groups 命令。

```
$ 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"
        }
    ]
}
```

删除安全组

要删除安全组，请运行 delete-security-group 命令。

Note

如果安全组当前已附加到环境，则无法删除。

EC2-VPC

以下命令删除一个 EC2-VPC 安全组。

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

EC2-Classic

以下命令删除名为 `my-sg` 的 EC2-Classic 安全组。

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

启动、列出和终止 Amazon EC2 实例

您可以使用 AWS Command Line Interface (AWS CLI) 启动、列出和终止 Amazon Elastic Compute Cloud (Amazon EC2) 实例。您需要密钥对 (p. 71)和安全组 (p. 73)。您还需要选择 Amazon 系统映像 (AMI)

并记下 AMI ID。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的查找合适的 AMI。

如果您启动不在 AWS 免费套餐范围内的实例，那么在启动实例后您将需要付费，费用按实例的运行时间计算，即使实例处于闲置状态也需付费。

Note

以下示例假定您已配置了默认凭证 (p. 71)。

主题

启动实例

要使用所选的 AMI 启动 Amazon EC2 实例，请使用 run-instances 命令。您可以将实例启动到 virtual private cloud (VPC)，或者如果您的账户支持，会启动到 EC2-Classic。

最初，您的实例显示为 `pending` 状态，但在几分钟后将更改为 `running` 状态。

EC2-VPC

以下示例在 VPC 的指定子网中启动 `t2.micro` 实例。将*##*参数值替换为您自己的值。

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

```
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "SubnetId": "subnet-6e7f829e",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
                    "Status": "in-use",
                    "SourceDestCheck": true,
                    "VpcId": "vpc-1a2b3c4d",
                    "Description": "Primary network interface",
                    "NetworkInterfaceId": "eni-a7edb1c9",
                    "PrivateIpAddresses": [
                        {
                            "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                            "Primary": true,
                            "PrivateIpAddress": "10.0.1.114"
                        }
                    ],
                    "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                    "Attachment": {
                        "Status": "attached",
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "AttachmentId": "eni-attach-52193138",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    },
                    "Groups": [
                        {
                            "GroupName": "my-sg",
                            "GroupId": "sg-903004f8"
                        }
                    ],
                    "SubnetId": "subnet-6e7f829e",
                    "OwnerId": "123456789012",
                    "PrivateIpAddress": "10.0.1.114"
                }
            ],
            "SourceDestCheck": true,
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
```

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

EC2-Classic

如果您的账户支持，您可以使用以下命令在 EC2-Classic 中启动 `t1.micro` 实例。将*##*参数值替换为您自己的值。

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

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

向实例添加块储存设备

每个启动的实例都具有关联的根设备卷。您可以使用块储存设备映射来指定实例启动时要连接的其他 Amazon Elastic Block Store (Amazon EBS) 卷或实例存储卷。

要向实例添加块储存设备，请在使用 `run-instances` 时指定 `--block-device-mappings` 选项。

以下示例参数配置大小为 20 GB 的标准 Amazon EBS 卷，并使用标识符 `/dev/sdf` 将其映射到您的实例。

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

以下示例基于现有快照添加映射到 `/dev/sdf` 的 Amazon EBS 卷。快照表示加载到卷的映像。当您指定快照时，无需指定卷大小；它将足够大可容纳您的映像。但是，如果您确定指定大小，则大小必须大于或等于快照的大小。

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

以下示例向实例添加两个卷。可用于您的实例的卷的数目取决于其实例类型。

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

以下示例创建映射 (`/dev/sdj`)，但未为实例预配置卷。

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

有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的块储存设备映射。

向实例添加标签

标签是您为 AWS 资源分配的标记。它允许您向您可用于各种目的的资源添加元数据。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的标记您的资源。

以下示例显示如何使用 create-tags 命令，将带有密钥名称“Name”和值“MyInstance”的标签添加到指定的实例。

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

连接到您的实例

当您的实例运行时，您可以连接到该实例，然后像使用您面前的计算机一样使用该实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的 Amazon EC2 实例。

列出实例

您可以使用 AWS CLI 列出您的实例并查看有关这些实例的信息。您可以列出所有实例，或根据您感兴趣的实例对结果进行筛选。

以下示例说明如何使用 describe-instances 命令。

以下命令将列表筛选到只仅限您的 t2.micro 实例，并仅为每个匹配项输出 InstanceId 值。

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

以下命令列出具有标签 Name=MyInstance 的任何实例。

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

以下命令列出使用以下任何 AMI 启动的实例：ami-x0123456、ami-y0123456 和 ami-z0123456。

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

终止实例

终止实例将删除此实例。在您终止之后，您将无法重新连接到此实例。

一旦实例的状态变为 shutting-down 或 terminated，您即停止为该实例付费。如果您希望稍后重新连接到实例，请使用 stop-instances，而不是 terminate-instances。想要了解更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例。

当您使用完该实例后，您可以使用命令 terminate-instances 将其删除。

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

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

将 Amazon S3 Glacier 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon S3 Glacier 的功能。要列出 S3 Glacier 的 AWS CLI 命令，请使用以下命令。

```
aws glacier help
```

本主题显示执行 S3 Glacier 常见任务的 AWS CLI 命令。这些示例演示如何使用 AWS CLI 将大型文件上传到 Glacier，方法是将它拆分为较小的部分并从命令行上传它们。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

> Note
>
> 本教程使用一些通常预安装在类 Unix 操作系统（包括 Linux 和 OS X）上的命令行工具。Windows 用户可通过安装 Cygwin 并从 Cygwin 终端运行命令来使用相同的工具。如有可执行相同功能的 Windows 本机命令和实用工具，将会注明。

主题

创建 Amazon S3 Glacier 文件库

使用 `create-vault` 命令创建文件库。

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

> Note
>
> 所有 S3 Glacier 命令都需要一个账户 ID 参数。使用连字符 (`--account-id -`) 以使用当前账户。

准备要上传的文件

创建一个用于测试上传的文件。以下命令将创建一个正好包含 3 MiB 随机数据的名为 *largefile* 的文件。

Linux, OS X, 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 是一个实用工具，该实用工具将大量字节从输入文件复制到输出文件。上一个示例使用系统设备文件 /dev/urandom 作为随机数据的源。fsutil 在 Windows 中执行相似的功能。

Windows

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

接下来，将文件拆分为 1 MiB（1,048,576 字节）的区块。

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

> **Note**
>
> HJ-Split 是一个免费的文件拆分器，适用于 Windows 和很多其他平台。

启动文件分段上传和上传

使用 initiate-multipart-upload 命令在 Amazon S3 Glacier 中创建分段上传。

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

S3 Glacier 需要每个部分的大小以字节为单位（本例中以 1 MiB 为单位）、您的文件库名称和一个账户 ID，用来配置分段上传。操作完成时，AWS CLI 会输出一个上传 ID。将上传 ID 保存到 shell 变量以待将来使用。

Linux, OS X, or Unix

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

Windows

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

接下来，使用 upload-multipart-part 命令上传三个部分的每个部分。

```
$ 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

上一个示例使用美元符号 ($) 在 Linux 上引用 `UPLOADID` shell 变量的内容。在 Windows 命令行上，在变量名称的任一侧使用百分号（例如，`%UPLOADID%`）。

在上传各个部分时，您必须指定其字节范围，以便 Glacier 可以按正确的顺序重组它。由于每个部分的大小为 1,048,576 字节，因此第一个部分占用 0-1048575 字节，第二个部分占用 1048576-2097151 字节，第三个部分占用 2097152-3145727 字节。

完成上传

Amazon S3 Glacier 需要原始文件的树形哈希，以确认所有上传的部分都已完整地到达 AWS。

要计算树形哈希，必须将文件拆分为 1 MiB 的部分并计算每个部分的二进制 SHA-256 哈希。然后，将哈希列表拆分成对，合并每对中的两个二进制哈希，并采用结果的哈希。重复此步骤，直到只剩下一个哈希。如果任一级别出现奇数数量的哈希，请将其提升到下一级别而无需修改。

在使用命令行实用工具时，正确计算树形哈希的关键是以二进制的形式存储每个哈希，并且仅在最后一步将其转换为十六进制。对树中的任何哈希的十六进制版本进行合并或哈希处理将导致错误结果。

Note

Windows 用户可使用 `type` 命令来代替 `cat`。OpenSSL 适用于 Windows，可在 OpenSSL.org 找到。

计算树形哈希

1. 将原始文件拆分为 1 MiB 的部分（如果您还没有这样做）。

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

2. 计算并存储每个区块的二进制 SHA-256 哈希。

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

3. 合并前两个哈希，并采用结果的二进制哈希。

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

4. 将区块 `aa` 和 `ab` 的父哈希与区块 `ac` 的哈希合并并对结果进行哈希处理，此时将输出十六进制。将结果存储在 shell 变量中。

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

最后，使用 complete-multipart-upload 命令完成上传。此命令采用原始文件的大小（以字节为单位）、最终树形哈希值（十六进制形式）以及您的账户 ID 和文件库名称。

```
$ 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"
}
```

您也可以使用 describe-vault 命令查看文件库的状态。

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

Note

基本上每天都会更新一次文件库的状态。有关更多信息，请参阅使用文件库。

现在可以安全删除您创建的块和哈希文件。

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

有关分段上传的更多信息，请参阅 Amazon S3 Glacier 开发人员指南中的分段上传大型档案和计算校验和。

从 AWS CLI 使用 AWS Identity and Access Management

您可以使用 AWS Command Line Interface (AWS CLI) 访问 AWS Identity and Access Management (IAM) 的功能。要列出 IAM 的 AWS CLI 命令，请使用以下命令。

```
aws iam help
```

本主题显示执行 IAM 常见任务的 AWS CLI 命令。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

主题

创建 IAM 用户和组

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令创建 IAM 组和新的 IAM 用户，然后将该用户添加到该组中。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

创建 IAM 组并向其中添加新 IAM 用户

1. 使用 `create-group` 命令创建组。

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

2. 使用 `create-user` 命令创建用户。

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

3. 使用 `add-user-to-group` 命令将用户添加到组中。

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

4. 要验证 `MyIamGroup` 组包含 `MyUser`，请使用 `get-group` 命令。

```
$ aws iam get-group --group-name MyIamGroup
{
    "Group": {
        "GroupName": "MyIamGroup",
        "CreateDate": "2018-12-14T03:03:52Z",
        "GroupId": "AGPAJNUJ2W4IJVEXAMPLE",
        "Arn": "arn:aws-cn:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    },
    "Users": [
        {
```

```
            "UserName": "MyUser",
            "Path": "/",
            "CreateDate": "2018-12-14T03:13:02Z",
            "UserId": "AIDAJY2PE5XUZ4EXAMPLE",
            "Arn": "arn:aws-cn:iam::123456789012:user/MyUser"
        }
    ],
    "IsTruncated": "false"
}
```

将 IAM 托管策略附加到 IAM 用户

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令将 IAM 策略附加到 IAM 用户。此示例中的策略为用户提供“高级用户访问”。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

将 IAM 托管策略附加到 IAM 用户

1. 确定要附加的策略的 ARN。以下命令使用 `list-policies` 查找具有名称 `PowerUserAccess` 的策略的 ARN。然后，它会将该 ARN 存储在环境变量中。

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

2. 要附加策略，请使用 `attach-user-policy` 命令，并引用存放策略 ARN 的环境变量。

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

3. 通过运行 `list-attached-user-policies` 命令验证策略已附加到此用户。

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

其他资源

有关更多信息，请参阅访问管理资源。该主题提供权限和策略概述的链接以及用于访问 Amazon S3、Amazon EC2 和其他服务的策略示例的链接。

为 IAM 用户设置初始密码

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令为 IAM 用户设置初始密码。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

以下命令使用 create-login-profile 为指定用户设置初始密码。当用户首次登录时，用户需要将密码更改为只有用户知道的内容。

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

您可以使用 update-login-profile 命令更改 IAM 用户的密码。

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

为 IAM 用户创建访问密钥

本主题介绍如何使用 AWS Command Line Interface (AWS CLI) 命令为 IAM 用户创建一组访问密钥。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

您可以使用 create-access-key 命令为 IAM 用户创建访问密钥。访问密钥是一组安全凭证，由访问密钥 ID 和私有密钥组成。

IAM 用户一次只能创建两个访问密钥。如果您尝试创建第三组，则命令返回 LimitExceeded 错误。

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

使用 delete-access-key 命令为 IAM 用户删除访问密钥。使用访问密钥 ID 指定要删除的访问密钥。

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

将 Amazon S3 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Simple Storage Service (Amazon S3) 的功能。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

AWS CLI 提供两个层级的命令来访问 Amazon S3：

- s3 层级由高级别命令构成，这些命令简化常见任务的执行，如创建、操作和删除对象及存储桶。
- 通过公开对所有 Amazon S3 API 操作的直接访问，s3api 层级的行为与其他 AWS 服务完全相同。它允许您执行单凭以下层级的高级别命令无法完成的高级操作。

要获得每个层级中提供的所有命令的列表，请在 aws s3api 或 aws s3 命令中使用 help 参数。

```
$ aws s3 help
```

```
$ aws s3api help
```

Note

AWS CLI 支持使用 Amazon S3 提供的服务器端 COPY操作从 Amazon S3 复制、移动和同步到 Amazon S3。这意味着，您的文件保留在云中，不会下载到客户端计算机，然后备份到 Amazon S3。
虽然这样的操作完全在云中执行，但只使用 HTTP 请求和响应所需的带宽。

有关使用 Amazon S3 的示例，请参阅本部分中的以下主题。

主题

通过 AWS CLI 使用高级别 (s3) 命令

本主题介绍如何使用高级别 aws s3 命令管理 Amazon S3 存储桶和对象。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

管理存储桶

高级别 aws s3 命令支持常用的存储桶操作，如创建、列出和删除存储桶。

创建存储桶

使用 s3 mb 命令创建存储桶。存储桶名称必须全局唯一，并且应符合 DNS 标准。存储桶名称可以包含小写字母、数字、连字符和点号。存储桶名称只能以字母或数字开头和结尾，连字符或点号后不能跟点号。

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

列出存储桶

使用 s3 ls 命令列出您的存储桶。下面是一些常见使用情况示例。

下面的命令列出所有存储桶。

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

下面的命令列出一个存储桶中的所有对象和文件夹（在 S3 中称为“前缀”）。

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

先前的输出显示在前缀 path/ 下有一个名为 MyFile1.txt 的文件。

您可以通过在命令中包含特定前缀将输出筛选到此前缀。下面的命令列出 *bucket-name*/path 中的对象（即 *bucket-name* 中按前缀 path/ 筛选后的对象）。

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

删除存储段

要删除存储桶，请使用 `s3 rb` 命令。

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

默认情况下，存储桶必须为空，此操作才能成功。要删除非空存储桶，需要包含 `--force` 选项。

以下示例删除存储桶中的所有对象和子文件夹，然后删除存储桶。

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

> **Note**
>
> 如果您使用的是受版本控制的存储桶，即其中包含以前删除“但仍保留”的对象，则此命令不 允许您删除该存储桶。您必须先删除所有内容。

管理对象

高级别 `aws s3` 命令可以方便地管理 Amazon S3 对象。这些对象命令包括 `s3 cp`、`s3 ls`、`s3 mv`、`s3 rm` 和 `s3 sync`。

`cp`、`ls`、`mv` 和 `rm` 命令的用法与它们在 Unix 中的对应命令相同，使您可以跨本地目录和 Amazon S3 存储桶无缝工作。`sync` 命令同步一个存储桶与一个目录或两个存储桶中的内容。

> **Note**
>
> 如果对象很大，所有涉及向 Amazon S3 存储桶（`s3 cp`、`s3 mv` 和 `s3 sync`）上传对象的高级命令都会自动执行分段上传。
> 使用这些命令时，无法恢复失败的上传。如果分段上传由于超时而失败，或者通过按 Ctrl+C 手动取消，AWS CLI 将会清除创建的所有文件并中止上传。此过程可能耗时数分钟。
> 如果进程被 kill 命令中断或者由于系统故障而中断，则正在进行的分段上传将保留在 Amazon S3 中，必须在 AWS 管理控制台中手动清除，或者使用 s3api abort-multipart-upload 命令来清除。

`cp`、`mv` 和 `sync` 命令包括一个 `--grants` 选项，可用来向指定用户或组授予对对象的权限。使用以下语法将 `--grants` 选项设置为权限列表。

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

每个值都包含以下元素：

- *`Permission`* – 指定授予的权限，可将其设置为 `read`、`readacl`、`writeacl` 或 `full`。
- *`Grantee_Type`* – 指定如何标识被授权者，可将其设置为 `uri`、 `emailaddress` 或 `id`。
- *`Grantee_ID`* – 根据 *`Grantee_Type`* 指定被授权者。
  - `uri` – 组 URI。有关更多信息，请参阅谁是被授权者？
  - `emailaddress` – 账户的电子邮件地址。
  - `id` – 账户的规范 ID。

有关 Amazon S3 访问控制的更多信息，请参阅访问控制。

下面的示例将一个对象复制到一个存储桶中。它授予所有人对对象的 read 权限，向 user@example.com 的关联账户授予 full 权限（read、readacl 和 writeacl）。

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

还可以为上传到 Amazon S3 的对象指定非默认存储类（REDUCED_REDUNDANCY 或 STANDARD_IA）。为此，请使用 --storage-class 选项。

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

s3 sync 命令使用如下语法。可能的源-目标组合有：

- 本地文件系统到 Amazon S3
- Amazon S3 到本地文件系统
- Amazon S3 到 Amazon S3

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

下面的示例将 my-bucket 中名为 path 的 Amazon S3 文件夹中的内容与当前工作目录同步。s3 sync 将更新与目标中的同名文件具有不同大小或修改时间的任何文件。输出显示在同步期间执行的特定操作。请注意，此操作将子目录 MySubdirectory 及其内容与 s3://my-bucket/path/MySubdirectory 递归同步。

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

通常，s3 sync 仅在源和目标之间复制缺失或过时的文件或对象。不过，您还可以提供 --delete 选项来从目标中删除源中不存在的文件或对象。

下面的示例对上一示例进行了扩展，显示了其工作方式。

```
// 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
```

可以使用 --exclude 和 --include 选项指定规则来筛选要在同步操作期间复制的文件或对象。默认情况下，指定文件夹中的所有项都包含在同步中。因此，仅当需要指定 --exclude 选项的例外情况（也就是

说，--include 实际上意味着“不排除”）时，才需要使用 --include。这些选项按指定顺序应用，如下例所示。

```
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 和 --include 选项也可以与要在 s3 sync 操作（包括 --delete 选项）期间删除的文件或对象。在这种情况下，参数字符串必须指定要在目标目录或存储桶上下文中包含或排除在删除操作中的文件。下面是一个示例。

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

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

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

s3 sync 命令还可以接受 --acl 选项，使用该选项可以设置对复制到 Amazon S3 中的文件的访问权限。--acl 选项接受 private、public-read 和 public-read-write 值。

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

如上文所述，s3 命令集包括 cp、mv、ls 和 rm，它们的用法与它们在 Unix 中的对应命令相同。下面是一些示例。

```
// 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
```

当 `--recursive` 选项与 `cp`、`mv` 或 `rm` 一起用于目录或文件夹时，命令会遍历目录树，包括所有子目录。与 `--exclude` 命令相同，这些命令也接受 `--include`、`--acl` 和 `sync` 选项。

通过 AWS CLI 使用 API 级 (s3api) 命令

API 级命令（包含在 `s3api` 命令集中）提供对 Amazon Simple Storage Service (Amazon S3) API 的直接访问，可以执行高级别 `s3` 命令中未公开的某些操作。这些命令等同于对服务功能提供 API 级访问的其他 AWS 服务的命令。

本主题提供了若干示例，以演示如何使用映射到 Amazon S3 API 的低级别命令。此外，您可以在 CLI 参考指南的 s3api 部分中找到每个 S3 API 的示例。

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

应用自定义 ACL

对于高级命令，您可以使用 `--acl` 选项对 Amazon S3 对象应用预定义的访问控制列表 (ACL)，但不能使用该命令设置存储桶范围的 ACL。但是，您可以通过 API 级命令 put-bucket-acl 执行此操作。

下面的示例说明如何向两个 AWS 用户（user1@example.com 和 user2@example.com）授予完全控制权限，并向所有人授予读取权限。“everyone”的标识符来自作为参数传递的特殊 URI。

```
$ 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.cn/groups/global/AllUsers"'
```

有关如何构建 ACL 的更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 PUT Bucket acl。CLI 中的 `s3api` ACL 命令（如 `put-bucket-acl`）使用相同的简化参数表示法。

配置日志记录策略

API 命令 `put-bucket-logging` 配置存储桶日志记录策略。

在下面的示例中，已向 AWS 用户 user@example.com 授予对日志文件的完全控制权限，而向所有用户授予了读取访问权限。请注意，还需要使用 `put-bucket-acl` 命令向 Amazon S3 的日志传输系统（由 URI 指定）授予必要的权限，以读取和向存储桶写入日志。

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

上一个命令中的文件 `logging.json` 具有以下内容。

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

将 Amazon SNS 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Simple Notification Service (Amazon SNS) 的功能。要列出 Amazon SNS 的 AWS CLI 命令，请使用以下命令。

```
aws sns help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

本主题显示执行 Amazon SNS 常见任务的 CLI 命令。

主题

创建主题

要创建主题，请使用 `create-topic` 命令并指定要分配给该主题的名称。

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

记下响应的 `TopicArn`，您随后将用它来发布消息。

订阅主题

要订阅主题，请使用 subscribe 命令。

以下示例为 notification-endpoint 指定 email 协议和电子邮件地址。

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

AWS 通过向您在 subscribe 命令中指定的地址发送电子邮件，立即发送确认电子邮件。电子邮件具有以下文本。

```
You have chosen to subscribe to the topic:
arn:aws-cn: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
```

收件人单击确认订阅链接后，收件人的浏览器显示通知消息，信息类似于以下内容。

```
Subscription confirmed!

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

Your subscription's id is:
arn:aws-cn:sns:us-west-2:123456789012:my-topic:1328f057-de93-4c15-512e-8bb22EXAMPLE

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

向主题发布

要将消息发送给某一主题的所有订阅者，请使用 publish 命令。

以下示例向指定主题的所有订阅者发送消息“Hello World!”。

```
$ aws sns publish --topic-arn arn:aws-cn:sns:us-west-2:123456789012:my-topic --
message "Hello World!"
{
    "MessageId": "4e41661d-5eec-5ddf-8dab-2c867EXAMPLE"
}
```

在本示例中，AWS 将包含文本“Hello World!”的电子邮件发送到 saanvi@example.com。

取消订阅主题

要取消订阅某个主题并停止接收向该主题发布的消息，请使用 unsubscribe 命令并指定您要取消订阅的主题的 ARN。

```
$ aws sns unsubscribe --subscription-arn arn:aws-cn:sns:us-west-2:123456789012:my-
topic:1328f057-de93-4c15-512e-8bb22EXAMPLE
```

要验证您已成功取消订阅，请使用 list-subscriptions 命令以确认该 ARN 不再显示在列表中。

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

删除主题

要删除主题，请运行 delete-topic 命令。

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

要验证 AWS 已成功删除主题，请使用 list-topics 命令以确认该主题不再显示在列表中。

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

将 Amazon SWF 与 AWS CLI 结合使用

您可以使用 AWS Command Line Interface (AWS CLI) 访问 Amazon Simple Workflow Service (Amazon SWF) 的功能。

要列出 Amazon SWF 的 AWS CLI 命令，请使用以下命令。

```
aws swf help
```

在运行任何命令之前，请设置默认证书。有关更多信息，请参阅 配置 AWS CLI (p. 20)。

本主题显示执行 Amazon SWF 常见任务的 CLI 命令。

主题

按类别列出 Amazon SWF 命令

您可以使用 AWS Command Line Interface (AWS CLI) 在 Amazon Simple Workflow Service (Amazon SWF) 中创建、显示和管理工作流。

本部分列出 AWS CLI 中 Amazon SWF 命令的参考主题（按功能类别分组）。

有关字母顺序的 命令列表，请参阅 AWS CLI Command Reference 的 Amazon SWF 部分，或者使用以下命令。

```
$ aws swf help
```

您也可以通过在命令名称后放入 `help` 指令以获取单个命令的帮助。下面是一个示例。

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

主题

与活动相关的命令

活动工作者使用 `poll-for-activity-task` 获取新活动任务。工作者从 Amazon SWF 收到活动任务后即执行该任务，如果成功，则使用 `respond-activity-task-completed` 进行响应，如果失败，则使用 `respond-activity-task-failed` 进行响应。

下面是由活动工作者执行的命令：

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

与决策者相关的命令

决策者使用 `poll-for-decision-task` 获取决策任务。决策者从 Amazon SWF 收到决策任务后，检查其工作流程执行历史记录并决定接下来要做什么。它调用 `respond-decision-task-completed` 以完成该决策任务，并提供零个或多个后续决策。

以下是由决策者执行的命令：

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

与工作流程执行相关的命令

对工作流程可执行以下命令：

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

与管理相关的命令

尽管可以从 Amazon SWF 控制台执行管理任务，您也可以使用本节中的命令自动执行各种功能或构建您自己的管理工具。

活动管理

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

工作流程管理

- register-workflow-type

- deprecate-workflow-type

域管理

- register-domain
- deprecate-domain

有关这些域管理命令的更多信息和示例，请参阅使用 AWS CLI处理 Amazon SWF 域 (p. 99)。

工作流执行管理

- request-cancel-workflow-execution
- terminate-workflow-execution

可见性命令

尽管可以从 Amazon SWF 控制台执行可见性操作，您也可以使用本节中的命令构建您自己的控制台或管理工具。

活动可见性

- list-activity-types
- describe-activity-type

工作流程可见性

- list-workflow-types
- describe-workflow-type

工作流程执行可见性

- describe-workflow-execution
- list-open-workflow-executions
- list-closed-workflow-executions
- count-open-workflow-executions
- count-closed-workflow-executions
- get-workflow-execution-history

域可见性

- list-domains
- describe-domain

有关这些域可见性命令的更多信息和示例，请参阅使用 AWS CLI处理 Amazon SWF 域 (p. 99)。

任务列表可见性

- count-pending-activity-tasks
- count-pending-decision-tasks

使用 AWS CLI处理 Amazon SWF 域

您可以使用 AWS Command Line Interface (AWS CLI) 管理您的 Amazon Simple Workflow Service (Amazon SWF) 域。

主题

列出您的域

要列出已为您的 AWS 账户注册的 Amazon SWF 域，可以使用 `swf list-domains`。您必须包含 `--registration-status` 并指定 `REGISTERED` 或 `DEPRECATED`。

下面是一个最简单的示例。

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

Note

有关使用 `DEPRECATED` 的示例，请参阅弃用域 (p. 100)。

获取有关域的信息

要获取有关特定域的详细信息，请使用 `swf describe-domain` 。有一个必需参数 `--name`，此参数用于指定您要获取其信息的域的名称。例如：

```
$ aws swf describe-domain --name ExampleDomain
{
    "domainInfo": {
        "status": "REGISTERED",
        "name": "ExampleDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "1"
    }
}
```

注册域

要注册新域，请使用 `swf register-domain`。

有两个必需参数：--name 和 --workflow-execution-retention-period-in-days，前者指定要注册的域名，后者使用一个整数指定在该域上保留工作流程执行数据的天数，最长 90 天（有关更多信息，请参阅 Amazon SWF 常见问题）。

如果您为此值指定零 (0)，则保留期自动设置为最长持续时间。否则，在指定的天数过后，不会保留工作流程执行数据。以下示例说明如何注册新域。

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

该命令不返回任何输出，但您可以使用 swf describe-domain 或 swf list-domains 查看新域。例如：

```
$ aws swf describe-domain --name MyNeatNewDomain
{
    "domainInfo": {
        "status": "REGISTERED",
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
    }
}
```

弃用域

要弃用域（您仍可以看到它，但不能在它上面创建新工作流程执行或注册类型），请使用 swf deprecate-domain。它只有一个必需参数 --name，此参数用于指定要弃用的域的名称。

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

与 register-domain 一样，不会返回任何输出。不过，如果您使用 list-domains 查看已注册的域，则会看到该域不会再显示出来。您还可以使用 --registration-status DEPRECATED。

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

另请参阅

- AWS CLI Command Reference 中的 deprecate-domain
- AWS CLI Command Reference 中的 describe-domain
- AWS CLI Command Reference 中的 list-domains
- AWS CLI Command Reference 中的 register-domain

排查 AWS CLI 错误

一般故障排除技巧：使用 --debug 选项

当 CLI 报告一个您不能立即理解的错误，或者产生您不期望的结果时，您首先要做的是获得关于该错误的更多详细信息。通过再次运行该命令并在命令行末尾包含 --debug 选项，即可完成此操作。这会使 AWS CLI 报告有关完成以下过程所需的每个步骤的详细信息：处理命令、将请求发送到 AWS 服务器、接收响应并将响应处理为您所看到的输出。输出中的详细信息可帮助您确定发生错误的步骤，以及可提供有关是什么触发错误的线索的上下文。

您可以将输出发送到一个要捕获它的文本文件以供日后查看，或者按要求将输出发送给 AWS 技术支持。

以下是使用和不使用 --debug 选项运行命令的示例：

```
$ aws iam list-groups --profile MyTestProfile
{
    "Groups": [
        {
            "Path": "/",
            "GroupName": "MyTestGroup",
            "GroupId": "AGPA0123456789EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/MyTestGroup",
            "CreateDate": "2019-08-12T19:34:04Z"
        }
    ]
}
```

当您包含 --debug 选项时，详细信息包括（或其他）：

- 查找凭证
- 解析提供的参数
- 构建发送到 AWS 服务器的请求
- 发送到 AWS 的请求的内容
- 原始响应的内容
- 带格式的输出

```
$ aws iam list-groups --profile MyTestProfile --debug
2019-08-12 12:36:18,305 - MainThread - awscli.clidriver - DEBUG - CLI version: aws-
cli/1.16.215 Python/3.7.3 Linux/4.14.133-113.105.amzn2.x86_64 botocore/1.12.205
2019-08-12 12:36:18,305 - MainThread - awscli.clidriver - DEBUG - Arguments entered to CLI:
 ['iam', 'list-groups', '--debug']
2019-08-12 12:36:18,305 - MainThread - botocore.hooks - DEBUG - Event session-initialized:
 calling handler <function add_scalar_parsers at 0x7fdf173161e0>
2019-08-12 12:36:18,305 - MainThread - botocore.hooks - DEBUG - Event session-initialized:
 calling handler <function register_uri_param_handler at 0x7fdf17dec400>
2019-08-12 12:36:18,305 - MainThread - botocore.hooks - DEBUG - Event session-initialized:
 calling handler <function inject_assume_role_provider_cache at 0x7fdf17da9378>
2019-08-12 12:36:18,307 - MainThread - botocore.credentials - DEBUG - Skipping environment
 variable credential check because profile name was explicitly set.
2019-08-12 12:36:18,307 - MainThread - botocore.hooks - DEBUG - Event session-initialized:
 calling handler <function attach_history_handler at 0x7fdf173ed9d8>
2019-08-12 12:36:18,308 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/
ec2-user/venv/lib/python3.7/site-packages/botocore/data/iam/2010-05-08/service-2.json
```

```
2019-08-12 12:36:18,317 - MainThread - botocore.hooks - DEBUG - Event building-command-
table.iam: calling handler <function add_waiters at 0x7fdf1731a840>
2019-08-12 12:36:18,320 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/
ec2-user/venv/lib/python3.7/site-packages/botocore/data/iam/2010-05-08/waiters-2.json
2019-08-12 12:36:18,321 - MainThread - awscli.clidriver - DEBUG - OrderedDict([('path-
prefix', <awscli.arguments.CLIArgument object at 0x7fdf171ac780>), ('marker',
 <awscli.arguments.CLIArgument object at 0x7fdf171b09e8>), ('max-items',
 <awscli.arguments.CLIArgument object at 0x7fdf171b09b0>)])
2019-08-12 12:36:18,322 - MainThread - botocore.hooks - DEBUG - Event building-
argument-table.iam.list-groups: calling handler <function add_streaming_output_arg at
 0x7fdf17316510>
2019-08-12 12:36:18,322 - MainThread - botocore.hooks - DEBUG - Event building-argument-
table.iam.list-groups: calling handler <function add_cli_input_json at 0x7fdf17da9d90>
2019-08-12 12:36:18,322 - MainThread - botocore.hooks - DEBUG - Event building-argument-
table.iam.list-groups: calling handler <function unify_paging_params at 0x7fdf17328048>
2019-08-12 12:36:18,326 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/
ec2-user/venv/lib/python3.7/site-packages/botocore/data/iam/2010-05-08/paginators-1.json
2019-08-12 12:36:18,326 - MainThread - awscli.customizations.paginate - DEBUG - Modifying
 paging parameters for operation: ListGroups
2019-08-12 12:36:18,326 - MainThread - botocore.hooks - DEBUG - Event building-argument-
table.iam.list-groups: calling handler <function add_generate_skeleton at 0x7fdf1737eae8>
2019-08-12 12:36:18,326 - MainThread - botocore.hooks - DEBUG - Event
 before-building-argument-table-parser.iam.list-groups: calling handler
 <bound method OverrideRequiredArgsArgument.override_required_args of
 <awscli.customizations.cliinputjson.CliInputJSONArgument object at 0x7fdf171b0a58>>
2019-08-12 12:36:18,327 - MainThread - botocore.hooks - DEBUG - Event
 before-building-argument-table-parser.iam.list-groups: calling handler
 <bound method GenerateCliSkeletonArgument.override_required_args of
 <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at
 0x7fdf171c5978>>
2019-08-12 12:36:18,327 - MainThread - botocore.hooks - DEBUG - Event operation-
args-parsed.iam.list-groups: calling handler functools.partial(<function
 check_should_enable_pagination at 0x7fdf17328158>, ['marker', 'max-items'], {'max-
items': <awscli.arguments.CLIArgument object at 0x7fdf171b09b0>}, OrderedDict([('path-
prefix', <awscli.arguments.CLIArgument object at 0x7fdf171ac780>), ('marker',
 <awscli.arguments.CLIArgument object at 0x7fdf171b09e8>), ('max-items',
 <awscli.customizations.paginate.PageArgument object at 0x7fdf171c58d0>), ('cli-
input-json', <awscli.customizations.cliinputjson.CliInputJSONArgument object at
 0x7fdf171b0a58>), ('starting-token', <awscli.customizations.paginate.PageArgument
 object at 0x7fdf171b0a20>), ('page-size', <awscli.customizations.paginate.PageArgument
 object at 0x7fdf171c5828>), ('generate-cli-skeleton',
 <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at
 0x7fdf171c5978>)]))
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.path-prefix: calling handler <awscli.paramfile.URIArgumentHandler
 object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.marker: calling handler <awscli.paramfile.URIArgumentHandler object at
 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.max-items: calling handler <awscli.paramfile.URIArgumentHandler object
 at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.cli-input-json: calling handler <awscli.paramfile.URIArgumentHandler
 object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.starting-token: calling handler <awscli.paramfile.URIArgumentHandler
 object at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event load-cli-
arg.iam.list-groups.page-size: calling handler <awscli.paramfile.URIArgumentHandler object
 at 0x7fdf1725c978>
2019-08-12 12:36:18,328 - MainThread - botocore.hooks - DEBUG - Event
 load-cli-arg.iam.list-groups.generate-cli-skeleton: calling handler
 <awscli.paramfile.URIArgumentHandler object at 0x7fdf1725c978>
2019-08-12 12:36:18,329 - MainThread - botocore.hooks - DEBUG
 - Event calling-command.iam.list-groups: calling handler
```

```
 <bound method CliInputJSONArgument.add_to_call_parameters of
 <awscli.customizations.cliinputjson.CliInputJSONArgument object at 0x7fdf171b0a58>>
2019-08-12 12:36:18,329 - MainThread - botocore.hooks - DEBUG -
 Event calling-command.iam.list-groups: calling handler <bound
 method GenerateCliSkeletonArgument.generate_json_skeleton of
 <awscli.customizations.generatecliskeleton.GenerateCliSkeletonArgument object at
 0x7fdf171c5978>>
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for
 credentials via: assume-role
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for
 credentials via: assume-role-with-web-identity
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - DEBUG - Looking for
 credentials via: shared-credentials-file
2019-08-12 12:36:18,329 - MainThread - botocore.credentials - INFO - Found credentials in
 shared credentials file: ~/.aws/credentials
2019-08-12 12:36:18,330 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/
ec2-user/venv/lib/python3.7/site-packages/botocore/data/endpoints.json
2019-08-12 12:36:18,334 - MainThread - botocore.hooks - DEBUG - Event choose-service-name:
 calling handler <function handle_service_name_alias at 0x7fdf1898eb70>
2019-08-12 12:36:18,337 - MainThread - botocore.hooks - DEBUG - Event creating-client-
class.iam: calling handler <function add_generate_presigned_url at 0x7fdf18a028c8>
2019-08-12 12:36:18,337 - MainThread - botocore.regions - DEBUG - Using partition endpoint
 for iam, us-west-2: aws-global
2019-08-12 12:36:18,337 - MainThread - botocore.args - DEBUG - The s3 config key is not a
 dictionary type, ignoring its value of: None
2019-08-12 12:36:18,340 - MainThread - botocore.endpoint - DEBUG - Setting iam timeout as
 (60, 60)
2019-08-12 12:36:18,341 - MainThread - botocore.loaders - DEBUG - Loading JSON file: /home/
ec2-user/venv/lib/python3.7/site-packages/botocore/data/_retry.json
2019-08-12 12:36:18,341 - MainThread - botocore.client - DEBUG - Registering retry handlers
 for service: iam
2019-08-12 12:36:18,342 - MainThread - botocore.hooks - DEBUG - Event before-parameter-
build.iam.ListGroups: calling handler <function generate_idempotent_uuid at 0x7fdf189b10d0>
2019-08-12 12:36:18,342 - MainThread - botocore.hooks - DEBUG - Event before-
call.iam.ListGroups: calling handler <function inject_api_version_header_if_needed at
 0x7fdf189b2a60>
2019-08-12 12:36:18,343 - MainThread - botocore.endpoint - DEBUG - Making
 request for OperationModel(name=ListGroups) with params: {'url_path': '/',
 'query_string': '', 'method': 'POST', 'headers': {'Content-Type': 'application/x-
www-form-urlencoded; charset=utf-8', 'User-Agent': 'aws-cli/1.16.215 Python/3.7.3
 Linux/4.14.133-113.105.amzn2.x86_64 botocore/1.12.205'}, 'body': {'Action':
 'ListGroups', 'Version': '2010-05-08'}, 'url': 'https://iam.amazonaws.com/', 'context':
 {'client_region': 'aws-global', 'client_config': <botocore.config.Config object at
 0x7fdf16e9a4a8>, 'has_streaming_input': False, 'auth_type': None}}
2019-08-12 12:36:18,343 - MainThread - botocore.hooks - DEBUG - Event request-
created.iam.ListGroups: calling handler <bound method RequestSigner.handler of
 <botocore.signers.RequestSigner object at 0x7fdf16e9a470>>
2019-08-12 12:36:18,343 - MainThread - botocore.hooks - DEBUG - Event choose-
signer.iam.ListGroups: calling handler <function set_operation_specific_signer at
 0x7fdf18996f28>
2019-08-12 12:36:18,343 - MainThread - botocore.auth - DEBUG - Calculating signature using
 v4 auth.
2019-08-12 12:36:18,343 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
POST
/

content-type:application/x-www-form-urlencoded; charset=utf-8
host:iam.amazonaws.com
x-amz-date:20190812T193618Z

content-type;host;x-amz-date
5f776d91EXAMPLE9b8cb5eb5d6d4a787a33ae41c8cd6eEXAMPLEca69080e1e1f
2019-08-12 12:36:18,344 - MainThread - botocore.auth - DEBUG - StringToSign:
AWS4-HMAC-SHA256
20190812T193618Z
20190812/us-east-1/iam/aws4_request
```

```
ab7e367eEXAMPLE2769f178ea509978cf8bfa054874b3EXAMPLE8d043fab6cc9
2019-08-12 12:36:18,344 - MainThread - botocore.auth - DEBUG - Signature:
d85a0EXAMPLEb40164f2f539cdc76d4f294fe822EXAMPLE18ad1ddf58a1a3ce7
2019-08-12 12:36:18,344 - MainThread - botocore.endpoint - DEBUG - Sending http request:
 <AWSPreparedRequest stream_output=False, method=POST, url=https://iam.amazonaws.com/,
 headers={'Content-Type': b'application/x-www-form-urlencoded; charset=utf-8',
 'User-Agent': b'aws-cli/1.16.215 Python/3.7.3 Linux/4.14.133-113.105.amzn2.x86_64
 botocore/1.12.205', 'X-Amz-Date': b'20190812T193618Z', 'Authorization': b'AWS4-HMAC-
SHA256 Credential=AKIA01234567890EXAMPLE-east-1/iam/aws4_request, SignedHeaders=content-
type;host;x-amz-date, Signature=d85a07692aceb401EXAMPLEa1b18ad1ddf58a1a3ce7EXAMPLE',
 'Content-Length': '36'}>
2019-08-12 12:36:18,344 - MainThread - urllib3.util.retry - DEBUG - Converted retries
 value: False -> Retry(total=False, connect=None, read=None, redirect=0, status=None)
2019-08-12 12:36:18,344 - MainThread - urllib3.connectionpool - DEBUG - Starting new HTTPS
 connection (1): iam.amazonaws.com:443
2019-08-12 12:36:18,664 - MainThread - urllib3.connectionpool - DEBUG - https://
iam.amazonaws.com:443 "POST / HTTP/1.1" 200 570
2019-08-12 12:36:18,664 - MainThread - botocore.parsers - DEBUG - Response headers: {'x-
amzn-RequestId': '74c11606-bd38-11e9-9c82-559da0adb349', 'Content-Type': 'text/xml',
 'Content-Length': '570', 'Date': 'Mon, 12 Aug 2019 19:36:18 GMT'}
2019-08-12 12:36:18,664 - MainThread - botocore.parsers - DEBUG - Response body:
b'<ListGroupsResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">\n
 <ListGroupsResult>\n    <IsTruncated>false</IsTruncated>\n    <Groups>\n
  <member>\n        <Path>/</Path>\n        <GroupName>MyTestGroup</GroupName>
\n        <Arn>arn:aws:iam::123456789012:group/MyTestGroup</Arn>\n
 <GroupId>AGPA1234567890EXAMPLE</GroupId>\n        <CreateDate>2019-08-12T19:34:04Z</
CreateDate>\n      </member>\n    </Groups>\n  </ListGroupsResult>\n  <ResponseMetadata>\n
    <RequestId>74c11606-bd38-11e9-9c82-559da0adb349</RequestId>\n  </ResponseMetadata>\n</
ListGroupsResponse>\n'
2019-08-12 12:36:18,665 - MainThread - botocore.hooks - DEBUG - Event needs-
retry.iam.ListGroups: calling handler <botocore.retryhandler.RetryHandler object at
 0x7fdf16e9a780>
2019-08-12 12:36:18,665 - MainThread - botocore.retryhandler - DEBUG - No retry needed.
2019-08-12 12:36:18,665 - MainThread - botocore.hooks - DEBUG - Event after-
call.iam.ListGroups: calling handler <function json_decode_policies at 0x7fdf189b1d90>
{
    "Groups": [
        {
            "Path": "/",
            "GroupName": "MyTestGroup",
            "GroupId": "AGPA123456789012EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/MyTestGroup",
            "CreateDate": "2019-08-12T19:34:04Z"
        }
    ]
}
```

运行 aws 时我收到“找不到命令”错误

可能的原因：安装期间未更新操作系统“路径”

此错误意味着操作系统找不到 AWS CLI 程序。安装可能不完整。

如果您使用 `pip` 安装 AWS Command Line Interface (AWS CLI)，您可能需要将包含 `aws` 程序的文件夹添加到操作系统的 `PATH` 环境变量，或更改其模式以使其可执行。

需要将 `aws` 可执行文件添加到操作系统的 `PATH` 环境变量中。按照相应过程中的步骤操作：

- Windows – 将 AWS CLI 可执行文件添加到命令行路径 (p. 13)
- macOS – 将 AWS CLI 可执行文件添加到 macOS 命令行路径 (p. 16)

- Linux – 将 AWS CLI 可执行文件添加到命令行路径 (p. 9)

我收到“拒绝访问”错误

可能的原因：CLI 程序文件没有“运行”权限

在 Linux 或 MacOS 上，确保 aws 程序具有调用用户的运行权限。通常，这些权限将设置为 755。

要添加用户的运行权限，请运行以下命令，并将 *~/.local/bin/aws* 替换为您计算机上的路径：

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

可能的原因：您的 IAM 身份无权执行该操作

当您运行 CLI 命令时，使用将您与 IAM 用户或角色关联的凭证代表您执行 AWS 操作。附加到该 IAM 用户或角色的策略必须向您授权权限，才能调用与通过 AWS CLI 运行的命令相对应的 API 操作。

大多数命令会通过一个与命令名匹配的名称来调用单个操作；但是，像 aws s3 sync 这样的自定义命令会调用多个 API。您可以查看命令通过使用 --debug 选项调用哪些 API。

如果您确定用户或角色具有策略所分配的适当权限，则确保您的 CLI 命令使用的是您预期的凭证。请参阅下一节中有关凭证的内容 (p. 105)以验证 CLI 使用的是否是您预期的凭证。

有关将权限分配给 IAM 用户和角色的信息，请参阅IAM 用户指南中的访问管理概述：权限和策略。

我收到“凭证无效”错误

可能的原因：AWS CLI 从意外位置读取了凭证

AWS CLI 读取凭证的位置可能与您的预期不同。您可以运行 aws configure list 以确认使用哪些凭证。

以下示例说明如何检查用于默认配置文件的凭证。

```
$ 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
```

以下示例说明如何检查命名配置文件的凭证。

```
$ aws configure list --profile saanvi
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                    saanvi           manual    --profile
access_key         **************** shared-credentials-file
secret_key         **************** shared-credentials-file
    region                us-west-2      config-file    ~/.aws/config
```

可能的原因：您计算机的时钟不同步

如果您使用有效的凭证，您的时钟可能不同步。在 Linux, OS X, or Unix 上，运行 `date` 以检查时间。

```
$ date
```

如果您的系统时钟在几分钟内不正确，则使用 `ntpd` 进行同步。

```
$ sudo service ntpd stop
$ sudo ntpdate time.nist.gov
$ sudo service ntpd start
$ ntpstat
```

在 Windows 上，使用控制面板中的日期和时间选项来配置系统时钟。

我收到“签名不匹配”错误

当 AWS CLI 运行命令时，它会向 AWS 服务器发送加密请求以执行适当的 AWS 服务操作。您的凭证（访问密钥和私有密钥）参与了加密过程，使 AWS 能够对发出请求的人员进行身份验证。有多种因素可能会干扰此过程的正常执行：

可能的原因：您的时钟与 AWS 服务器不同步

为了帮助防范重播攻击，在加密/解密过程中可能会使用当前时间。如果客户端和服务器的时间不一致超出允许的时间量，则该过程可能会失败，并且请求会被拒绝。当您在其时钟与主机时钟不同步的虚拟机中运行命令时，也可能发生此错误。一个可能的原因是，当虚拟机休眠时，唤醒后需要一些时间才能将时钟与主机重新同步。

在 Linux, OS X, or Unix 上，运行 `date` 以检查时间。

```
$ date
```

如果您的系统时钟在几分钟内不正确，则使用 `ntpd` 进行同步。

```
$ sudo service ntpd stop
$ sudo ntpdate time.nist.gov
$ sudo service ntpd start
$ ntpstat
```

在 Windows 上，使用控制面板中的日期和时间选项来配置系统时钟。

可能的原因：您的操作系统错误地处理了包含某些特殊字符的 AWS 私有密钥

如果您的 AWS 私有密钥包含某些特殊字符（例如 –、+、/ 或 %），则某些操作系统变体会不正确地处理该字符串，并导致对该私有密钥字符串的解释不正确。

如果使用其他工具或脚本（例如，在创建新实例期间在其上构建凭证文件的工具）处理访问密钥和私有密钥，则这些工具和脚本可能有自己对特殊字符的处理，这会使它们转换为 AWS 不再识别的内容。

简单的解决方案是重新生成私有密钥，以获得一个不包含特殊字符的私有密钥。

AWS CLI 用户指南文档历史记录

下表介绍了自 2018 年 1 月以来对 AWS Command Line Interface 用户指南 文档的重要补充。如需对此文档更新的通知，您可以订阅 RSS 源。

update-history-change	update-history-description	update-history-date
新的 MFA 部分	增加了一个新部分，其中介绍如何使用多重验证和角色访问 CLI。	May 3, 2019
更新了“配置 CLI”部分	对 CLI 配置说明和过程进行了重大改进和补充。	March 7, 2019
更新了“安装 CLI”部分	对 CLI 安装说明和过程进行了重大改进和补充。	March 7, 2019
更新了“使用 CLI”部分	对使用说明和过程进行了重大改进和补充。	March 7, 2019