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 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, macOS, or Unix 中运行命令。
- Windows 命令行 – 在 Microsoft Windows 上，在 PowerShell 或 Windows 命令提示符中运行命令。
- 远程 – 通过远程终端（如 PuTTY 或 SSH）或者使用 Amazon EC2 系统管理器在 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 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、工具包或 适用于 Windows PowerShell 的 AWS 工具。

- 适用于 Windows 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 Toolkit for Eclipse
- AWS Toolkit for Visual Studio
- 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 – 命令提示符显示为美元符号后跟一个空格（“$ ”）。请勿在键入命令时包含提示符。
- 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，格式化为 `computer output`。

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，请在命令行处键入 **`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并按 ENTER。然后，以显示的格式输入区域名称，按 ENTER，然后最后一次按 ENTER 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"
}
```

要使用该示例，请输入命令的完整文本（提示符后突出显示的文本）并按 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 访问密钥 ID）和应替换的参数（如组名）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 Command Line Interface

安装 AWS CLI 的方式

- pip (p. 3)
- 使用虚拟环境 (p. 3)
- 使用捆绑安装程序 (p. 3)

要求

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

> **Note**
>
> 较旧版本的 Python 可能无法兼容所有 AWS 服务。如果在安装或使用 &CLI; 时看到 `InsecurePlatformWarning` 或弃用通知，请更新到最新版本。

使用 *pip* 安装 AWS CLI：

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

当前 AWS CLI 版本

经常更新 AWS CLI 以支持新服务和命令。要了解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` 和受支持的 Python 版本，则可以使用以下命令安装 AWS CLI：

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

`--upgrade` 选项通知 `pip` 升级已安装的任何必要组件。`--user` 选项通知 `pip` 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。

在虚拟环境中安装 AWS CLI

如果在尝试通过 `pip` 安装 AWS CLI 时遇到问题，可以在虚拟环境中安装 AWS CLI (p. 13)，从而隔离工具及其依赖项；或者使用与平时使用的 Python 不同的版本。

使用安装程序安装 AWS CLI

若要在 Linux, macOS, or Unix 上进行离线或自动安装，请尝试捆绑安装程序 (p. 14)。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. 8)。这两种方法都简化了初始安装，并对新版本 AWS CLI 发布后升级更加困难的情况做出权衡。

安装后需要执行的步骤

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. 6)
- Windows – 将 AWS CLI 可执行文件添加到命令行路径 (p. 10)
- macOS – 将 AWS CLI 可执行文件添加到命令行路径 (p. 12)

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

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

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。

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

如果需要卸载 AWS CLI，请使用 `pip uninstall`。

```
pip uninstall awscli
```

如果您没有 Python 和 `pip`，则使用适合您的操作系统的过程：

针对每个环境的详细说明

- 在 Linux 上安装 AWS Command Line Interface (p. 4)
- 在 Microsoft Windows 上安装 AWS Command Line Interface (p. 8)
- 在 macOS 上安装 AWS Command Line Interface (p. 10)
- 在虚拟环境中安装 AWS Command Line Interface (p. 13)
- 使用捆绑安装程序安装 AWS CLI (Linux, macOS, or Unix) (p. 14)

在 Linux 上安装 AWS Command Line Interface

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 及其相关组件。

> Important
>
> 存储库中的 `awscli` 程序包可供其他程序包管理器（如 APT 和 yum）使用，但是，除非通过 `pip` 或使用捆绑安装程序 (p. 14)获得该程序包，否则不一定是最新版本。

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。

```
pip --version
```

如果您没有 pip，请检查以查看安装的是哪个版本的 Python。

```
python --version
```

或者

```
python3 --version
```

如果还没有 Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+，则必须安装 Python (p. 6)。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. 脚本将会下载，并将安装 `pip` 的最新版本以及另一个名为 `setuptools` 的必需程序包。使用 Python 运行脚本：

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

3. 将可执行文件的路径添加到您的 PATH 变量中：`~/.local/bin`

   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. 在配置文件脚本的末尾添加导出命令。

   ```
   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。

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。

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

使用 Pip 安装 AWS CLI

使用 `pip` 安装 AWS CLI。

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

验证 AWS CLI 是否已正确安装。

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

如果出现错误，请参阅排查 AWS CLI 错误 (p. 87)。

要升级到最新版本，请重新运行安装命令：

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

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。

Example AWS CLI 安装位置 - 带 pip (用户模式) 的 Linux

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

如果您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 `which python`。

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

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

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

如果这是在安装 Pip (p. 5) 的步骤 3 中添加到 Path 的文件夹，则不必再执行任何操作。否则，请再次执行步骤 3a 到 3c 将该文件夹添加到 Path。

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
```

   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 衍生物 (如 Ubuntu) 上，请使用 `APT`：

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

   - 在 Red Hat 及其衍生物上，请使用 `yum`：

```
sudo yum install python
```

   - 在 SUSE 及其衍生物上，请使用 `zypper`：

```
sudo zypper install python3
```

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 3.6.2
```

在 Amazon Linux 上安装 AWS Command Line Interface

该 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/1.11.84 Python/3.6.2 Linux/ botocore/1.5.47
```

您可以使用 `sudo yum update` 获取 yum 存储库中可用的最新版本，但这可能不是最新版本。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 Command Line Interface (p. 4)。

在 Amazon Linux 上升级 AWS CLI（根）

1. 使用 `pip install` 安装最新版本的 AWS CLI。

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

2. 使用 `aws --version` 验证新版本。

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

如果您没有根特权，请以用户模式安装 AWS CLI。

在 Amazon Linux 上升级 AWS CLI（用户）

1. 使用 `pip install` 安装最新版本的 AWS CLI。

```
sudo pip 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 `~/.bashrc` 末尾以维护会话之间的更改。

3. 使用 `aws --version` 验证新版本。

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

在 Microsoft Windows 上安装 AWS Command Line Interface

可在 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 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

MSI 安装程序

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。如果您倾向于经常更新，请考虑使用 pip (p. 9)，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）。请在此版本的 Windows 中使用 pip (p. 9) 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。

CLI 默认情况下安装到 `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（如果您不确定命令提示符安装在何处，请打开开始菜单并搜索 `cmd` 以启动命令提示符）。

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

键入命令时，请勿包含提示符符号（上面的“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. 10) 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，请按照上面的详细说明，再次下载和运行 MSI 安装程序。

卸载

要卸载 AWS CLI，请打开控制面板并选择程序和功能。选择名为 AWS Command Line Interface 的条目，并单击卸载启动卸载程序。收到提示时，请确认您要卸载 AWS CLI。

您还可以使用以下命令，从命令行启动程序和功能程序：

```
appwiz.cpl
```

在 Windows 上安装 Python、pip 和 AWS CLI

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

安装 Python 3 和 pip (Windows)

1. 从 Python.org 的下载页面下载 Python 3 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。

随 pip 安装 AWS CLI (Windows)

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

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

3. 使用 pip 安装 AWS CLI：

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

4. 验证 AWS CLI 是否已正确安装：

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

要升级到最新版本，请重新运行安装命令：

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

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，将 aws 程序添加到操作系统的 PATH 环境变量。对于 MSI 安装，此操作将自动执行，但如果 aws 命令不可用，则您可能需要手动设置它。

- Python 3 和 pip – `C:\Program Files\Python37\Scripts\`
- Python 3 和 pip --user 选项 – `%USERPROFILE%\AppData\Local\Programs\Python\Python37\Scripts`
- MSI 安装程序（64 位）– `C:\Program Files\Amazon\AWSCLI`
- MSI 安装程序（32 位）– `C:\Program Files (x86)\Amazon\AWSCLI`

修改您的 PATH 变量 (Windows)

1. 按 Windows 键并键入 **environment variables**。
2. 选择 Edit environment variables for your account。
3. 选择 PATH，然后选择 Edit。
4. 向 Variable value 字段添加路径，中间用分号隔开。例如：*C:\existing\path*;***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。

在 macOS 上安装 AWS Command Line Interface

要在 macOS 上安装 AWS CLI，建议使用捆绑安装程序。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

- 使用 pip 在 macOS 上安装 AWS CLI (p. 12)
- 将 AWS CLI 可执行文件添加到命令行路径 (p. 12)

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 Command Line Interface (p. 4)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

1. 下载 AWS CLI 捆绑安装程序。

   ```
   curl "https://s3.amazonaws.com/aws-cli/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. 运行安装程序。

   ```
   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 程序的绝对路径）来运行安装脚本。例如:

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

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
```

使用 pip 在 macOS 上安装 AWS CLI

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

```
pip --version
```

在 macOS 上安装 AWS CLI

1. 从 Python.org 的下载页面下载并安装 Python 3.6。
2. 下载并运行 Python 打包权威机构提供的 `pip` 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 `pip` 安装 AWS CLI。

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

4. 验证 AWS CLI 是否已正确安装。

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

   如果未找到该程序，请将它添加到命令行路径 (p. 12)。

要升级到最新版本，请重新运行安装命令：

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

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，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 3.7 和 pip（用户模式）的 macOS

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

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/python3.7
```

`pip` 将程序安装到 Python 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, macOS, 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. 将配置文件加载到当前会话。

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

在虚拟环境中安装 AWS Command Line Interface

您可以通过在虚拟环境中安装 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. 创建虚拟环境并为其命名。

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

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, macOS, or Unix)

在 Linux, macOS, or Unix 上，可以使用捆绑安装程序来安装 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, macOS, 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 Command Line Interface (p. 4)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

1. 下载 AWS CLI 捆绑安装程序。

```
$ curl "https://s3.amazonaws.com/aws-cli/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
```

不使用 Sudo 安装 AWS CLI (Linux, macOS, or Unix)

如果您没有 sudo 权限，或打算仅为当前用户安装 AWS CLI，则可使用以上命令的修改版本：

```
$ 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 PATH 环境变量中包含 ~/bin，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

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 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

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，并将它们存储在名为 `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，就会使用该配置文件。

访问密钥/凭证

`AWS Access Key ID` 和 `AWS Secret Access Key` 是您的 AWS 凭证。它们与 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`。

快速配置和多个配置文件

如果使用上面显示的命令，则结果为名为 `default` 的单个配置文件。您也可以使用 `--profile` 选项指定配置文件的名称，以创建附加配置。

```
aws configure --profile user2
AWS Access Key ID [None]: AKIAI44QH8DHBEXAMPLE
AWS Secret Access Key [None]: je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]: text
```

然后，运行命令时，可以省略 `--profile` 选项，这样会使用 `default` 配置文件中存储的设置：

```
aws s3 ls
```

或者，您也可以指定 `--profile` *`profilename`* 来使用该名称下存储的设置：

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

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，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. 22) – 您可以在命令行上指定 --region、--output 和 --profile 作为参数。
2. 环境变量 (p. 21) – 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，则使用这些变量。
3. CLI 凭证文件 (p. 19) – 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, macOS, 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. 19) – 这是运行命令 aws configure 时更新的文件之一。该文件位于 ~/.aws/config（在 Linux, macOS, 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 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

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, macOS, or Unix

```
ls  ~/.aws
```

Windows

```
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. 21)。

在 Config 中存储证书

AWS CLI 也可以从 `config` 文件读取凭证。如果您需要将所有配置文件设置保存在一个文件中，是可以做到的。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。
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，您会发现其他警告。

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. 20)。

支持以下设置：

aws_access_key_id – AWS 访问密钥。

aws_secret_access_key – AWS 私有密钥。

aws_session_token – AWS 会话令牌。只有在使用临时安全凭证时才需要会话令牌。

region – 使用该配置文件时，将请求发送到的默认 AWS 区域。

output (p. 18) – 该配置文件的默认输出格式。

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。在不指定配置文件的情况下运行 CLI 命令时，使用第一个；在指定 `--profile user1` 参数的情况下运行 CLI 命令时，使用第二个。

~/.aws/credentials

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

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

每个配置文件使用不同的证书（可能来自不同的 IAM 用户），并且使用不同的区域和输出格式：

~/.aws/config

```
[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“profile”。配置 credentials 文件时，不要使用 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。下面的示例列出了使用前面示例文件中的 user1 配置文件的所有 Amazon EC2 实例：

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

如果要为多个命令使用一个指定的配置文件，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, macOS, or Unix

```
export AWS_PROFILE=user2
```

Windows

```
set AWS_PROFILE=user2
```

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。下节进一步介绍变量。

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。

Important

设置以下环境变量之一会覆盖所有其他指定该选项的方式（但将选项指定为命令行参数的方式除外）。

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. 18)。
- AWS_DEFAULT_OUTPUT – 指定要使用的输出格式。

- `AWS_PROFILE` – 指定包含要使用的凭证和选项的 CLI 配置文件 (p. 20)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。
- `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（默认为 `~/.aws/credentials`）。
- `AWS_CONFIG_FILE` – 指定 AWS CLI 用于存储配置文件的位置（默认为 `~/.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，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。

Linux, macOS, or Unix

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

Windows

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

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

指定用于该命令的命名配置文件 (p. 20)。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

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

指定用于该命令的输出格式。您可以指定以下任意值：

--endpoint-url

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。

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。每个选项都采用一个字符串参数，以及一个空格或等号 (=) 将参数与选项名称分隔开来。如果参数值包含空格，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. 38)中详细描述了每个命令行选项的参数类型（字符串、布尔值等）。

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 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。

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

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

Linux, macOS, or Unix

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

Windows

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

在这些示例中，`http://a.b.c.d:n` 和 `http://w.x.y.z:m` 是 HTTP 和 HTTPS 代理的 IP 地址和端口号。

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, macOS, or Unix

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

Windows

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

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

对 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 。

Linux, macOS, or Unix

```
export NO_PROXY=
```

Windows

```
set NO_PROXY=
```

代入 IAM 角色

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/config` 文件中为某个角色定义配置文件，您可以配置 AWS Command Line Interface 以使用 IAM 角色。下面的示例显示了一个名为 `marketingadmin` 的配置文件，当您运行指定 `marketingadmin` 配置文件的命令时，将代入该配置文件。

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

该角色必须链接到一个单独的命名配置文件，此配置文件包含 IAM 用户凭证及代入该角色的权限。在上面的示例中，marketingadmin 配置文件使用 source-profile 字段链接到 user1 配置文件。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。然后，使用这些临时凭证来运行该 CLI 命令。指定的角色必须附加有允许运行该 CLI 命令的 IAM 权限策略。

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代入角色的信任关系。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。下面的示例显示了一个允许账户 123456789012 中的任何 IAM 用户代入角色的信任关系，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: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。下面的示例仅允许附加的 IAM 用户代入 marketingadmin 角色：

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

```
}
```

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，运行命令所需的权限来自附加到角色 的权限。但是，如果您希望用户能够在不 使用角色的情况下访问 AWS 资源，则必须向 IAM 用户附加授予这些资源权限的其他内联或托管策略。

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
```

如果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_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, macOS, or Unix

```
export AWS_PROFILE=marketingadmin
```

Windows

```
set 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设备（一种 U2F 设备）或移动应用程序生成的一次性密钥。

首先，将与 IAM 角色有关的信任关系修改为需要多重验证。有关示例，请参阅下面示例中的 `Condition` 行：

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

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

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。

跨账户角色

通过将角色配置为跨账户角色，您可以让 IAM 用户代入属于不同账户的角色。在创建角色期间，将角色类型设置为其他 AWS 账户（如创建向 IAM 用户委派权限的角色中所述），并可根据需要选择需要 MFA。需要 MFA 选项将按照使用多重验证 (p. 26)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代入角色的人员，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:iam::234567890123:role/SomeRole
source_profile = default
mfa_serial = arn:aws:iam::123456789012:mfa/saanvi
external_id = 123456
```

清除缓存凭证

当您代入一个角色时，AWS CLI 会在本地缓存临时凭证，直到这些凭证过期。如果您的角色的临时凭证已撤销，则您可以删除缓存以强制 AWS CLI 检索新凭证。

Linux, macOS, or Unix

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

Windows

```
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 的安装目录。这通常会与所使用的 Shell 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

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。

```
$ which aws_completer
/usr/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
```

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。

- bash – 使用内置命令 complete。

  ```
  complete -C '/usr/local/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模式来定义完成行为。

  ```
  > 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/bin/aws_zsh_completer.sh
```

  AWS CLI 使用 bash 兼容性自动完成 (bashcompinit) 实现 zsh 支持。有关进一步详细信息，请参阅 aws_zsh_completer.sh。

  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，键入部分命令并按 Tab 查看可用命令。

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

使用 AWS Command Line Interface 在 Amazon EC2 中部署开发环境

本教程详细说明如何使用 AWS CLI 在 Amazon EC2 中设置开发环境。它包括简略版的安装和配置说明，并且可在 Windows、Linux, macOS, or Unix 上自始至终运行。

步骤

安装 AWS CLI

您可以使用安装程序 (Windows) 或使用 `pip`（一种适用于 Python 的程序包管理器）来安装 AWS CLI。

Windows

1. 下载 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
2. 运行下载的 MSI 安装程序。
3. 按显示的说明执行操作。

Linux, macOS, or Unix

这些步骤要求您正常安装 Python 2 版本 2.6.5+ 或 Python 3 版本 3.3+。如果您在使用下列步骤时遇到问题，请参阅 AWS Command Line Interface 用户指南中的完整安装说明。

1. 从 pip 网站下载并运行安装脚本：

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

2. 使用 `pip` 安装 AWS CLI：

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

配置AWS CLI

在命令行运行 `aws configure` 以设置您的证书和设置。

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

AWS CLI 会提示您输入以下信息：

- AWS 访问密钥 ID 和 AWS 秘密访问密钥 – 这些是您的账户凭证。如果您没有密钥，请参阅 Amazon Web Services 一般参考 中的如何获取安全凭证？。
- 默认区域名称 – 这是您希望默认对其进行调用的区域的名称。
- 默认输出格式 – 此格式可以是 json、text 或 table。如果不指定输出格式，将使用 json。

运行命令以验证您的凭证是否配置正确，以及您能否连接到 AWS。

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

为 EC2 实例创建安全组和密钥对

下一步是设置启动可使用 SSH 访问的 EC2 实例的先决条件。有关 Amazon EC2 功能的更多信息，请转至 Amazon EC2 用户指南（适用于 Linux 实例）

创建安全组、密钥对和角色

1. 首先，创建一个新的安全组并添加一个规则以允许 SSH 的通过端口 22 的传入流量。如果您对该区域使用默认 VPC，则可以省略 `--vpc-id` 参数，否则指定您在其中启动实例的 VPC 的 ID。为提高安全性，请将 `0.0.0.0/0` CIDR 范围替换为您要从中连接到实例的网络的范围。

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

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

在启动实例时记下安全组 ID，以便以后使用。

2. 接下来，创建一个密钥对以便能连接到实例。此命令会将密钥内容保存到一个名为 `devenv-key.pem` 的文件中。

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

Windows

在 Windows 命令提示符中，用双引号 (而非单引号)。

3. 在 Linux 上，您还需要更改文件模式，以便仅您有权访问密钥文件。

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

启动并连接到实例

最后，您已准备好启动一个实例并连接到该实例。

启动并连接到实例

1. 使用您在上一步中创建的安全组的 ID 运行以下命令。`--image-id` 参数指定 Amazon EC2 用于引导实例的 Amazon 系统映像 (AMI)。您可以使用 Amazon EC2 控制台查找您的区域和操作系统的映像 ID。如果您对默认 VPC 使用默认子网，则可以省略 `--subnet-id` 参数，否则指定您在其中启动实例的子网的 ID。

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

2. 启动实例需要一些时间。实例启动并运行后，您需要实例的公有 IP 地址来连接它。使用以下命令获取公有 IP 地址：

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

3. 要连接到实例，请将公有 IP 地址和私有密钥用于首选终端程序。在 Linux, macOS, or Unix 上，您可以在命令行使用以下命令执行此操作：

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

如果您在尝试连接到实例时收到类似 Permission denied (publickey) 的错误，请检查以下内容是否正确：

- 密钥 – 指定的密钥必须位于指示的路径，并且必须是私钥而不是公钥。密钥的权限必须限制为所有者。
- 用户 – 用户名必须与关联到用于启动实例的 AMI 的默认用户名匹配。对于 Ubuntu AMI，它为 `ubuntu`。对于 Amazon Linux AMI，它为 `ec2-user`。

- 实例 – 实例的公有 IP 地址或 DNS 名称。验证该地址是否为公有地址以及端口 22 是否对实例的安全组上的本地计算机开放。

您还可使用 `-v` 选项查看与错误相关的其他信息。

Windows 上的 SSH

在 Windows 上，您可以使用此处提供的 PuTTY 终端应用程序。从下载页面获取 `putty.exe` 和 `puttygen.exe`。
使用 `puttygen.exe` 将私有密钥转换为 PuTTY 所需的 `.ppk` 文件。启动 `putty.exe`，在 Host Name 字段中输入实例的公有 IP 地址，并将连接类型设置为 SSH。
在 Category 面板中，导航到 Connection > SSH > Auth，单击 Browse 以选择您的 `.ppk` 文件，然后单击 Open 进行连接。

4. 终端将提示您接受服务器的公有密钥。键入 `yes` 并单击 Enter 以完成连接。

您现已完成以下操作：配置安全组，创建密钥对，启动 EC2 实例并连接到该实例，而这些操作都是使用命令行执行的。

使用 AWS Command Line Interface

此部分介绍在 AWS Command Line Interface中使用的常见功能和调用模式。

> Note
>
> AWS CLI 通过 HTTPS 对服务进行 API 调用。必须在 TCP 端口 443 上启用出站连接才能执行调用。

主题

使用 AWS Command Line Interface获取帮助

要在使用 AWS CLI 时获取帮助，您只需在命令末尾添加 `help`。例如，以下命令会列出常规 AWS CLI 选项和可用顶层命令的帮助。

```
$ aws help
```

以下命令会列出 Amazon EC2 的可用子命令。

```
$ aws ec2 help
```

下一个示例列出 EC2 `DescribeInstances` 操作的详细帮助，包括对它的输入参数、筛选条件和输出的描述。如果您不确定如何编写某个命令，请查看帮助的示例部分。

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

每个命令的帮助分为六个部分：

Name – 命令的名称。

```
NAME
       describe-instances -
```

Description – 命令调用的 API 操作的描述，取自于命令的服务的 API 文档。

```
DESCRIPTION
       Describes one or more of your instances.
```

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

Synopsis – 命令及其选项的列表。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]
```

`describe-instances` 具有描述当前账户和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，则应在 `OPTIONS` 部分中描述这些条件，或在示例中显示这些条件。

Options – 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.
...
```

Examples – 一些示例，用于显示命令及其选项的使用方法。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 a Owner tag

    Command:
```

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

Output – 来自 AWS 的响应中返回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.
...
```

当输出由 AWS CLI 以 JSON 格式呈现时，它将成为预留对象的数组，类似于以下内容：

```
{
    "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

通过管道将 help 命令的输出发送到 `more` 以便每次查看一页帮助文件。按空格键或 Page Down 键可查看文档的更多内容，按 q 可退出。

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

AWS CLI 文档

AWS CLI Command Reference 提供了所有 AWS CLI 命令的帮助文件的内容，这些内容已经过编译并在线提供，可轻松实现在移动设备、平板电脑和台式机屏幕上导航和查看。

帮助文件有时包含无法从命令行视图查看或点开的链接；这些链接保留在在线 AWS CLI 参考中。

API 文档

AWS CLI 中的所有子命令对应于对服务的公用 API 进行的调用。反过来，具有公用 API 的每个服务都有一组 API 参考文档，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可能的错误状况的详细信息。

API 文档的各部分

- Actions – 有关特定于某个操作的参数（包括对长度或内容的约束）和错误的详细信息。操作对应于 AWS CLI 中的子命令。
- Data Types – 可能包含有关由子命令返回的对象数据的其他信息。
- Common Parameters – 有关由服务的所有操作使用的参数的详细信息。
- Common Errors – 有关由服务的所有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 用于处理所有服务不同，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 Command Line Interface中的命令结构

AWS CLI 在命令行中使用多部分结构。这种结构一开始是对 aws 的基本调用。下一部分指定一个顶级命令，通常表示 AWS CLI 中支持的 AWS 服务。每个 AWS 服务均拥有指定要执行的操作的附加子命令。一个操作的常规 CLI 选项或特定参数可在命令行中以任何顺序指定。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。

为 AWS Command Line Interface指定参数值

很多参数都是简单的字符串或数值，如以下示例中的 my-key-pair 密钥对名称：

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

不带任何空格字符的字符串可以用引号引起来，也可以不用引号。但是，包含一个或多个空格字符的字符串必须用引号引起来。在 Linux, macOS, or Unix 和 Windows PowerShell 中使用单引号 (')，在 Windows 命令提示符中使用双引号 (")，如下例所示。

Windows PowerShell、Linux, macOS, or Unix

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

Windows 命令处理程序

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

您也可以使用等号来代替空格。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服务要求它们应符合的格式。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** 来查看手册，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。不建议使用标准空格字符以外的符号和空格，否则在使用 AWS CLI 时可能会导致问题。

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. 42)。

Timestamp – 将根据 ISO 8601 标准设置时间戳格式。这些有时称为“DateTime”或“Date”类型参数。

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

可接受的格式包括：

- YYYY-MM-DDThh:mm:ss.sssTZD (UTC)，例如，2014-10-01T20:30:00.000Z
- YYYY-MM-DDThh:mm:ss.sssTZD（带偏移量），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

List – 以空格分隔的一个或多个字符串。

```
$ 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，则针对服务验证命令而不实际运行查询。

```
$ 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://`。

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 或速记语法 (p. 51)指定的一系列密钥值对。以下示例使用 map 参数 `--key` 从名为 my-table 的 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"
        }
    }
}
```

下一节将更详细地介绍 JSON 参数。

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 EC2 实例。

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

以下示例在 JSON 数组中指定筛选器的等效列表。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）。

请注意，“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"]
  }
]
```

另一方面，仅在指定一个以上的筛选器时需要最外层的括号。上述命令的 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"]}'
```

有些操作需要将数据格式设置为 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 Elastic Block Store 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（请参阅引用字符串 (p. 41)），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. 41)）。

当传入大块数据时，将 JSON 保存为一个文件并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。下一部分将介绍这一方法。

有关 JSON 的更多信息，请参阅 Wikipedia - JSON 和 RFC4627 - JSON 的应用程序/json 媒体类型。

引用字符串

在命令行中输入 JSON 格式参数的方式因操作系统而异。Linux, macOS, or Unix 和 Windows PowerShell 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"}}]'
```

而 Windows 命令提示符使用双引号 (") 括住 JSON 数据结构。此外，JSON 数据结构中的每个双引号 (") 本身都需要一个反斜杠 (\) 转义字符，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\"}}]"
```

Windows PowerShell 需要使用单引号 (') 来括住 JSON 数据结构，还需要使用反斜杠 (\) 来对 JSON 结构中的每个双引号 (") 进行转义，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\"}}]'
```

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。`RedrivePolicy` 的值是一个 JSON 文档，必须对其进行转义：

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

从文件加载参数

要避免在命令行中对 JSON 字符串进行转义的需要，请从文件中加载该 JSON。通过使用 `file://` 前缀提供文件路径来从本地文件加载参数，如以下示例所示。

Linux, macOS, or Unix

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

// Read from a file in /tmp
```

```
$ aws ec2 describe-instances --filters file:///tmp/filter.json
```

Windows

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

`file://` 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 7 上，您通常在 `C:\Users\`*User Name*`\` 下有一个用户目录。

作为参数键的值提供的 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: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 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，但要求文件存储在可公开访问的位置。

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

在前面的示例中，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 格式参数的文件的其他示例，请参阅为 IAM 用户设置 IAM 策略 (p. 71)。

生成 CLI 框架和 CLI 输入 JSON 参数

大多数 AWS CLI 命令支持 --generate-cli-skeleton 和 --cli-input-json 参数，可使用这些参数在 JSON 中存储参数并从文件中读取参数（而不是在命令行上键入参数）。

生成将概述可为操作指定的所有参数的 CLI 框架输出 JSON。

将 -generate-cli-skeleton 与 aws ec2 run-instances 结合使用

1. 执行带 --generate-cli-skeleton 选项的 run-instances 命令可查看 JSON 框架。

```
$ aws ec2 run-instances --generate-cli-skeleton
{
    "DryRun": true,
    "ImageId": "",
    "MinCount": 0,
    "MaxCount": 0,
    "KeyName": "",
    "SecurityGroups": [
        ""
    ],
    "SecurityGroupIds": [
        ""
    ],
    "UserData": "",
    "InstanceType": "",
    "Placement": {
        "AvailabilityZone": "",
        "GroupName": "",
        "Tenancy": ""
    },
    "KernelId": "",
    "RamdiskId": "",
    "BlockDeviceMappings": [
        {
            "VirtualName": "",
            "DeviceName": "",
            "Ebs": {
                "SnapshotId": "",
                "VolumeSize": 0,
                "DeleteOnTermination": true,
                "VolumeType": "",
                "Iops": 0,
                "Encrypted": true
            },
```

```
            "NoDevice": ""
        }
    ],
    "Monitoring": {
        "Enabled": true
    },
    "SubnetId": "",
    "DisableApiTermination": true,
    "InstanceInitiatedShutdownBehavior": "",
    "PrivateIpAddress": "",
    "ClientToken": "",
    "AdditionalInfo": "",
    "NetworkInterfaces": [
        {
            "NetworkInterfaceId": "",
            "DeviceIndex": 0,
            "SubnetId": "",
            "Description": "",
            "PrivateIpAddress": "",
            "Groups": [
                ""
            ],
            "DeleteOnTermination": true,
            "PrivateIpAddresses": [
                {
                    "PrivateIpAddress": "",
                    "Primary": true
                }
            ],
            "SecondaryPrivateIpAddressCount": 0,
            "AssociatePublicIpAddress": true
        }
    ],
    "IamInstanceProfile": {
        "Arn": "",
        "Name": ""
    },
    "EbsOptimized": true
}
```

2. 将输出定向到文件可本地保存框架：

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

3. 在文本编辑器中打开框架并删除不使用的任何参数：

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

   将 `DryRun` 参数设置为 true 可使用 EC2 的空运行功能，可利用此功能在不创建资源的情况下测试配置。

4. 在默认区域中填写实例类型、密钥名称、安全组和 AMI 的值。在此示例中，`ami-dfc39aef` 是 `us-west-2` 区域中的 64 位 Amazon Linux 映像。

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

5. 使用 `file://` 前缀将 JSON 配置传递给 `--cli-input-json` 参数：

```
$ 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，直至显示空运行错误。

6. 将 DryRun 参数设置为 false 可禁用空运行功能。

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

7. 再次运行 `run-instances` 命令可启动实例：

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

控制 AWS Command Line Interface的命令输出

此部分介绍控制 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. 17)主题所述，输出格式可通过三种不同方式指定：

- 在配置文件中使用 `output` 选项。以下示例将输出设置为 `text`：

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

- 使用 `AWS_DEFAULT_OUTPUT` 环境变量。例如：

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

- 在命令行上使用 `--output` 选项。例如：

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

> **Note**
>
> 如果以多种方式指定输出格式，则通用 AWS CLI 优先规则 (p. 19)适用。例如，使用 `AWS_DEFAULT_OUTPUT` 环境变量将用 `output` 覆盖配置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或配置文件中设置的任何值。

JSON 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 格式便于人阅读，text 格式适合在传统 Unix 文本处理工具（例如 `sed`、`grep` 和 `awk`）中以及 Windows PowerShell 脚本中使用。

如何使用 --query 选项筛选输出

AWS CLI 使用 `--query` 选项提供内置输出筛选功能。为演示其工作方式，我们首先来看看下面的默认 JSON 输出，它描述了连接到不同 EC2 实例的两个 EBS (Elastic Block Storage) 卷。

```
$ 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
        }
    ]
}
```

首先，我们可以使用以下命令仅显示 Volumes 列表中的第一个卷。

```
$ 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
}
```

现在，我们使用通配符表示法 [*] 循环访问整个列表，并筛选出三个元素：VolumeId、AvailabilityZone 和 Size。请注意，词典表示法要求您为每个键提供一个别名，如：{Alias1:Key1,Alias2:Key2}。词典本身是无序的，因此，此种结构中的键别名的顺序在某些情况下可能不一致。

```
$ 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，您也可以使用串联的键（如 key1.key2[0].key3）来筛选深度嵌套在结构中的元素。下面的示例通过 Attachments[0].InstanceId 键（其别名为简单的 InstanceId）对此进行演示。

```
$ 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]。这样做会对每个对象将筛选出的所有属性格式化为一个排序 列表，而不管类型如何。

```
$ 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。

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，请访问 http://jmespath.org/specification.html。

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 `sed`、`grep` 和 `awk`）以及 Windows PowerShell 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
```

我们强烈建议将文本输出与 *`--query`* 选项一起使用以确保行为一致。这是因为 text 格式按字母顺序对输出列进行排序，而类似的资源可能并不总是具有相同的一组键。例如，Linux EC2 实例的 JSON 表示形式中的元素在 Windows 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，以支持对输出格式的完全控制。以下示例中的命令预先选择要显示的元素，并使用列表表示法 `[key1, key2, ...]` 定义列的顺序。这可令用户十分放心：正确的键/值将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
```

以下示例演示 `grep` 和 `awk` 如何与 `aws ec2 describe-instances` 命令的文本输出一起使用。第一个命令在文本输出中显示每个实例的可用区、状态和实例 ID。第二个命令仅输出在 `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
```

下一个命令显示所有已停止的实例的类似示例，并执行进一步处理，以自动更改每个已停止的实例的实例类型。

```
$ 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
```

Text 输出同样适用于 Windows PowerShell。因为 AWS CLI 的文本输出使用制表符分隔，因此在 PowerShell 中可很轻松地使用 `t 分隔符分隔为阵列。以下命令在第一列 (InstanceId) 与 AvailabilityZone–2 匹配的情况下显示第三列 (us-west-2a) 的值。

```
> 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
```

Table 输出格式

table 格式生成易于阅读的 AWS CLI 输出表示。示例如下：

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

```
||+--------------------------+-----------------------+-------------+--------------
+------------+----------------+||
||                                                    Volumes
                        ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|| AvailabilityZone |        CreateTime         | Size  |  SnapshotId    |  State  |
 VolumeId     | VolumeType   ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
||  us-west-2a      |  2013-09-18T20:26:15.000Z |  8    |  snap-708e8348 |  in-use |
 vol-2e410a47  |  standard    ||
|+------------------+--------------------------+-------+----------------+---------
+----------------+--------------+|
|||                                                Attachments
                       |||
||+--------------------------+-----------------------+-------------+--------------
+------------+----------------+||
|||        AttachTime         |  DeleteOnTermination   |   Device    | InstanceId   |
 State    |   VolumeId     |||
||+--------------------------+-----------------------+-------------+--------------
+------------+----------------+||
|||  2013-09-18T20:26:16.000Z |  True                  |  /dev/sda1  |  i-4b41a37c  |
 attached  |  vol-2e410a47  |||
||+--------------------------+-----------------------+-------------+--------------
+------------+----------------+||
```

--query 选项可与 table 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，字典和列表表示法中的输出的区别：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。

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

将速记语法与 AWS Command Line Interface结合使用

尽管 AWS Command Line Interface可以使用 JSON 格式的非标量选项参数，但在命令行上键入较大的 JSON 列表或结构会比较繁琐。为了解决此问题，AWS CLI 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, macOS, or Unix

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

的 AWS 工具

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

该示例等同于以下 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 DynamoDB `update-table` 命令示例包含采用速记语法指定的 `--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

可用两种方法以列表形式指定输入参数：JSON 和快速输入。使用 AWS CLI 的速记语法，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 EC2 的 `stop-instances` 命令示例，其中，`--instance-ids` 选项的输入参数（字符串列表）采用速记语法指定。

```
$ 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"]'
```

接下来是 Amazon EC2 `create-tags` 命令的示例，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 Command Line Interface 的分页选项

对于可返回项目的大型列表的命令，AWS CLI 添加了三个选项。当 CLI 调用服务的 API 以填充此列表时，您可使用这三个选项修改 CLI 的分页行为。

默认情况下，CLI 使用页面大小 1000 并检索所有可用项目。例如，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。

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可能过高，导致对 AWS 服务的调用超时。您可使用 `--page-size` 选项指定小一些的页面大小来解决此问题。CLI 仍将检索完整列表，但会在后台执行大量调用，以便减少每次调用时检索的项目数：

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

要减少检索的项目数，请使用 `--max-items` 选项。CLI 将以相同方式处理分页，但只会显示您指定的数量的项目：

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

如果项目输出的数量 (`--max-items`) 少于项目的总量，则输出将包含您可在后续命令中传递的 `NextToken` 以检索下一组项目：

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

每次调用服务时返回项目的顺序可能不同。如果您在页面中间指定了一个“下一个”标记，则可能会看到意外的结果。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 CLI 的分页和服务分页。您还可以检索整个列表并在本地执行任何必需的分析操作。

使用 Amazon Web Services

此部分提供使用 AWS Command Line Interface访问 AWS 服务的示例。这些示例旨在演示如何使用 AWS CLI 执行管理任务。

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 Command Line Interface获取帮助 (p. 34)。

主题

将 Amazon DynamoDB 与 AWS Command Line Interface配合使用

AWS Command Line Interface (AWS CLI) 提供对 Amazon DynamoDB 的支持。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。

命令行格式为 Amazon DynamoDB API 名称后接该 API 的参数。AWS CLI 支持参数值的速记语法以及 JSON。

例如，以下命令创建一个名为 `MusicCollection` 表。

> **Note**
>
> 为便于阅读，本节中的长命令分行显示。利用反斜杠字符，可以将多个行复制并粘贴（或键入）到 Linux 终端。如果您使用的 shell 未使用反斜杠对字符进行转义，请将反斜杠替换为其他转义字符，或者删除反斜杠并将整条命令放在一个行中。

```
$ 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
```

以下命令将新项目添加到表中。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"
    }
}
```

在命令行上，难以编写有效的 JSON；然而，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：

Example expression-attributes.json

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

您现在可以使用 AWS CLI 发出 `Query` 请求。在本示例中，`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 用于 DynamoDB 的更多文档，请转到 https://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html。

除了 DynamoDB 之外，您还可以结合使用 AWS CLI 和 DynamoDB Local。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 中的任何表或数据。所有 API 操作均重新路由到 DynamoDB Local。当应

用程序创建表或修改数据时，这些更改会写入本地数据库。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

通过 AWS Command Line Interface使用 Amazon EC2

您可以使用 AWS CLI 访问 Amazon EC2 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. 17)。

有关 Amazon EC2 的常见任务的示例，请参阅以下主题。

主题

使用密钥对

您可以使用 AWS CLI 创建、显示和删除密钥对。在您启动和连接到 Amazon EC2 实例时，必须指定密钥对。

> Note
>
> 在尝试示例命令之前，请设置默认证书。

主题

创建密钥对

要创建名为 `MyKeyPair` 的密钥对，请在 create-key-pair 命令中使用 `--query` 选项和 `--output text` 选项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
```

请注意，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，您必须在 `out-file` 命令中显式指定 ASCII 编码。

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

得到的 MyKeyPair.pem 文件与下面类似：

```
-----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。

如果您在 Linux 计算机上使用 SSH 客户端连接到您的实例，请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。此值存储在 AWS 中，可在 EC2 管理控制台中查看，或通过调用 `aws ec2 describe-key-pairs` 查看。例如，您可以使用以下命令查看 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》中的 Amazon EC2 密钥对页面。

删除您的密钥对

要删除 MyKeyPair，请使用 delete-key-pair 命令，如下所示：

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

使用安全组

您可以创建在 EC2-Classic 或 EC2-VPC 中使用的安全组。有关 EC2-Classic 和 EC2-VPC 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 CLI 创建安全组，在其中添加规则，以及删除安全组。

> Note
>
> 在尝试示例命令之前，请设置默认证书。

主题

正在创建安全组

要创建名为 `my-sg` 的安全组，请使用 create-security-group 命令。

EC2-VPC

以下命令为指定的 VPC 创建名为 `my-sg` 的安全组：

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

要查看 `my-sg` 的初始信息，请使用 describe-security-groups 命令，如下所示。请注意，您不能通过名称引用 EC2-VPC 的安全组。

```
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

以下命令为 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，请使用 describe-security-groups 命令，如下所示：

```
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

如果您要启动 Windows 实例，则必须添加规则以允许 TCP 端口 3389 (RDP) 上的入站流量。如果您要启动 Linux 实例，则必须添加规则以允许 TCP 端口 22 (SSH) 上的入站流量。使用 authorize-security-group-ingress 命令可向您的安全组添加规则。此命令的必需参数之一是您的计算机的公有 IP 地址（采用 CIDR 表示法）。

> **Note**
>
> 您可以通过一项服务来获取本地计算机的公有 IP 地址。例如，我们提供以下服务：https://checkip.amazonaws.com/。要查找另一项可提供您的 IP 地址的服务，请使用搜索短语“what is my IP address”。如果您正通过 ISP 或从防火墙后面连接，没有静态 IP 地址，您需要找出客户端计算机使用的 IP 地址范围。

EC2-VPC

以下命令将适用于 RDP 的规则添加到 ID 为 sg-903004f8 的安全组：

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

以下命令将适用于 SSH 的规则添加到 ID 为 sg-903004f8 的安全组：

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

要查看对 my-sg 所做的更改，请使用 describe-security-groups 命令，如下所示：

```
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 的规则添加到安全组 my-sg：

```
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 的规则添加到 my-sg 的安全组：

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

要查看对 my-sg 所做的更改，请使用 describe-security-groups 命令，如下所示：

```
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，请使用 delete-security-group 命令。请注意，如果安全组已附加到环境，则无法删除。

EC2-VPC

以下命令删除 ID 为 sg-903004f8 的安全组：

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

EC2-Classic

以下命令删除名为 my-sg 的安全组：

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

使用 Amazon EC2 实例

您可以使用 AWS CLI 启动、列出和终止实例。您需要有密钥对和安全组；有关通过 AWS CLI 创建密钥对和安全组的信息，请参阅使用密钥对 (p. 56)和使用安全组 (p. 58)。您还需要选择亚马逊系统映像 (AMI) 并记下其 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
>
> 在尝试示例命令之前，请设置默认证书。

主题

启动实例

要使用所选的 AMI 启动单个 Amazon EC2 实例，请使用 run-instances 命令。根据您的账户支持的平台，您可以在 EC2-Classic 或 EC2-VPC 中启动该实例。

最初，您的实例处于 pending 状态，但在几分钟后将进入 running 状态。

EC2-VPC

以下命令在指定子网中启动一个 t2.micro 实例：

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-
name MyKeyPair --security-group-ids sg-xxxxxxxx --subnet-id subnet-xxxxxxxx
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "PrivateIpAddress": "10.0.1.114",
            "ProductCodes": [],
            "VpcId": "vpc-1a2b3c4d",
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": ip-10-0-1-114.ec2.internal,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "SubnetId": "subnet-6e7f829e",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
                    "Status": "in-use",
                    "SourceDestCheck": true,
                    "VpcId": "vpc-1a2b3c4d",
                    "Description": "Primary network interface",
                    "NetworkInterfaceId": "eni-a7edb1c9",
                    "PrivateIpAddresses": [
                        {
                            "PrivateDnsName": "ip-10-0-1-114.ec2.internal",
                            "Primary": true,
                            "PrivateIpAddress": "10.0.1.114"
                        }
```

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

EC2-Classic

以下命令在 EC2-Classic 中启动一个 `t1.micro` 实例：

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t1.micro --key-
name MyKeyPair --security-groups my-sg
{
```

```
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Groups": [
        {
            "GroupName": "my-sg",
            "GroupId": "sg-903004f8"
        }
    ],
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": null,
            "Platform": "windows",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2013-07-19T02:42:39.000Z",
            "ProductCodes": [],
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "PrivateDnsName": null,
            "KeyName": "MyKeyPair",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg",
                    "GroupId": "sg-903004f8"
                }
            ],
            "ClientToken": null,
            "InstanceType": "t1.micro",
            "NetworkInterfaces": [],
            "Placement": {
                "Tenancy": "default",
                "GroupName": null,
                "AvailabilityZone": "us-west-2b"
            },
            "Hypervisor": "xen",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                        "AttachTime": "2013-07-19T02:42:39.000Z"
                    }
                }
            ],
            "Architecture": "x86_64",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "RootDeviceName": "/dev/sda1",
            "VirtualizationType": "hvm",
            "RootDeviceType": "ebs",
            "Tags": [
                {
                    "Value": "MyInstance",
                    "Key": "Name"
                }
            ],
```

```
            "AmiLaunchIndex": 0
        }
    ]
}
```

向实例添加块储存设备映射

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 EBS 卷或实例存储卷。

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

以下示例将添加映射到 `/dev/sdf` 的标准 Amazon EBS 卷，其大小为 20 GB。

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

以下示例基于快照添加映射到 `/dev/sdf` 的 Amazon EBS 卷。指定快照时，无需指定卷大小，但如果指定卷大小，其必须大于或等于快照的大小。

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

以下示例添加两个实例存储卷。请注意，可用于您的实例的实例存储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\"}]"
```

以下示例省略了由 AMI 指定的用于启动实例的设备映射 (`/dev/sdj`)：

```
--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。

向实例添加名称标签

要向实例添加标签 `Name=MyInstance`，请使用 create-tags 命令，如下所示：

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

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。

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。

> **Note**
>
> 在尝试示例命令之前，请设置默认证书。

以下示例说明如何使用 describe-instances 命令。

Example 1：列出具有指定实例类型的实例

以下命令列出您的 t2.micro 实例。

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

Example 2：列出具有指定标签的实例

以下命令列出具有标签 Name=MyInstance 的实例。

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

Example 3：列出使用指定映像启动的实例

以下命令列出从以下 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

终止实例可有效地删除实例；无法在终止实例后重新连接到实例。一旦实例的状态变为 shutting-down 或 terminated，您即停止为该实例付费。

在使用完该实例后，请使用 terminate-instances 命令，如下所示：

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

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。

将 Amazon S3 Glacier 与 AWS Command Line Interface配合使用

您可以将大型文件上传到 Glacier，方法是将它拆分为较小的部分并从命令行上传它们。本主题介绍使用 AWS CLI 创建文件库、拆分文件以及配置并执行到 Glacier 的分段上传的过程。

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 Glacier 文件库

使用 `aws glacier create-vault` 命令创建文件库。以下命令创建名为 `myvault` 的文件库。

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

Note

所有 glacier 命令都需要一个账户 ID 参数。请使用连字符指定当前账户。

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（3 x 1024 x 1024 字节）随机数据的文件。

Linux, macOS, or Unix

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

`dd` 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。以上示例使用设备文件 `/dev/urandom` 作为随机数据的源。`fsutil` 在 Windows 中执行相似的功能：

Windows

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

接下来，将文件拆分为 1 MiB（1048576 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 `aws glacier initiate-multipart-upload` 命令在 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"
}
```

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, macOS, or Unix

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

Windows

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

接下来，使用 `aws glacier upload-multipart-part` 命令上传每个部分。

```
$ 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

以上示例使用美元符号（“$”）对 `UPLOADID` shell 变量解除引用。在 Windows 命令行上，请使用两个百分比符号（例如，`%UPLOADID%`）。

在上传各个部分时，您必须指定其字节范围，以便 Glacier 按适当的顺序将其重组。由于每个部分的大小为 1048576 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

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全到达 AWS。要计算树形哈希，请将文件拆分为 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 `aws glacier 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 `aws glacier describe-vault` 查看文件库的状态：

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

Note

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部分和哈希文件：

```
$ 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。

AWS Command Line Interface中的 AWS Identity and Access Management

本节介绍一些与 AWS Identity and Access Management (IAM) 相关的常见任务以及如何使用 AWS Command Line Interface 执行这些任务。

此处显示的命令假设您已设置了默认证书和默认区域。

主题

创建新 IAM 用户和组

本节介绍如何创建新 IAM 组和新 IAM 用户，然后将该用户添加到该组中。

创建 IAM 组并向其中添加新 IAM 用户

1. 首先，使用 `create-group` 命令创建组。

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

```
    }
}
```

2. 然后，使用 create-user 命令创建用户。

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

3. 最后，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": "2012-12-20T03:03:52Z",
        "GroupId": "AKIAI44QH8DHBEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/MyIamGroup",
        "Path": "/"
    },
    "Users": [
        {
            "UserName": "MyUser",
            "Path": "/",
            "CreateDate": "2012-12-20T03:13:02Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/MyUser"
        }
    ],
    "IsTruncated": "false"
}
```

您还可以使用 AWS 管理控制台 查看 IAM 用户和组。

为 IAM 用户设置 IAM 策略

以下命令显示如何将 IAM 策略分配给 IAM 用户。此处指定的策略为用户提供“高级用户访问”。此策略与 IAM 控制台中提供的高级用户访问策略模板相同。在此示例中，策略保存为文件 MyPolicyFile.json：

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

要指定策略，请使用 put-user-policy 命令。

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

使用 list-user-policies 命令验证策略已分配给用户。

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

其他资源

有关更多信息，请参阅用于了解权限和策略的资源。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

下面的示例演示如何使用 create-login-profile 命令为 IAM 用户设置初始密码。

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

使用 update-login-profile 命令为 IAM 用户更新密码。

为 IAM 用户创建安全凭证

下面的示例使用 create-access-key 命令为 IAM 用户创建安全凭证。一组安全凭证包含一个访问密钥 ID 和一个私有密钥。请注意，一个 IAM 用户在任意给定时间最多只能有两组证书。如果您尝试创建第三组证书，create-access-key 命令会返回“LimitExceeded”错误。

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

使用 delete-access-key 命令为 IAM 用户删除一组证书。使用访问密钥 ID 指定要删除的证书。

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

将 Amazon S3 与 AWS Command Line Interface配合使用

AWS CLI 提供两个层级的命令来访问 Amazon S3。

- 第一个层级名为 s3，由高级别命令构成，这些命令用于频繁使用的操作，如创建、操作和删除对象及存储桶。
- 第二个层级名为 s3api，用于公开所有 Amazon S3 操作，包括修改存储桶访问控制列表 (ACL)、使用跨源的资源共享 (CORS) 或日志记录策略。它允许您执行单凭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 s3` 或 `aws s3api` 命令中使用 `help` 参数：

```
$ aws s3 help
```

或者

```
$ aws s3api help
```

> **Note**
>
> AWS CLI 支持从 Amazon S3 到 Amazon S3 进行复制、移动和同步。这些操作使用 Amazon S3 提供的服务端 COPY 操作：您的文件保存在云中，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 Command Line Interface使用高级别 s3 命令

本节介绍如何使用高级别 `aws s3` 命令管理 Amazon S3 存储桶和对象。

管理存储桶

高级别 `aws s3` 命令支持常用存储桶操作，如创建、删除和列出存储桶。

创建存储桶

使用 `aws s3 mb` 命令可以创建新存储桶。存储桶名称必须唯一，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
```

删除存储桶

要删除存储桶，请使用 `aws 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。

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

这将先删除存储桶中的所有对象和子文件夹，然后删除存储桶。

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，则此命令将不 允许您删除该存储桶。

列出存储桶

要列出所有存储桶或其内容，请使用 `aws s3 ls` 命令。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
2013-07-11 17:08:50 my-bucket
2013-07-24 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（前缀）。

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

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

管理对象

您还可以使用高级 `aws s3` 命令方便地管理 Amazon S3 对象。这些对象命令包括 `aws s3 cp`、`aws s3 ls`、`aws s3 mv`、`aws s3 rm` 和 `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（`aws s3 cp`、`aws s3 mv` 和 `aws 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。您可以使用以下语法对权限列表设置 `--grants` 选项。

```
--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* – 指定被授权者的标识方法，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/groups/
global/AllUsers full=emailaddress=user@example.com
```

要为上传到 Amazon S3 的对象指定非默认存储类（REDUCED_REDUNDANCY 或 STANDARD_IA），请使用 --storage-class 选项：

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

sync 命令的形式如下。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，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
```

使用 --exclude 和 --include 选项可以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，指定目录中的所有项都包含在同步中。因此，仅当指定 --include 选项的例外情况（例如，--exclude 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 --delete 选项一起使用来筛选要在同步操作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
```

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。该选项接受 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/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 Command Line Interface使用 API 级 (s3api) 命令

API 级命令（包含在 s3api 命令集中）提供对 Amazon S3 API 的直接访问，可以执行高级命令中未公开的某些操作。本节介绍 API 级命令并提供一些示例。有关更多 Amazon S3 示例，请参阅 s3api 命令行参考并从列表中选择可用命令。

自定义 ACL

借助高级命令，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)，但不能设置存储桶范围的 ACL。可以使用 API 级命令 put-bucket-acl 执行此操作。下面的示例向两个 AWS 用户（user1@example.com 和 user2@example.com）授予完全控制权限，向所有人授予读取权限。

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

有关自定义 ACL 的详细信息，请参阅 PUT Bucket acl。s3api ACL 命令（如 put-bucket-acl）使用相同的简化参数表示法。

日志记录策略

API 命令 put-bucket-logging 配置存储桶日志记录策略。下面的示例为 MyBucket 设置日志记录策略。AWS 用户 user@example.com 将对日志文件拥有完全控制权限，所有用户都将拥有访问权限。请注

意，必须使用 `put-bucket-acl` 命令才能向 Amazon S3 的日志传输系统授予必要的权限（write-acp 和 read-acp）。

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

logging.json

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

将 AWS Command Line Interface与 Amazon SNS 结合使用

本节介绍一些与 Amazon Simple Notification Service (Amazon SNS) 相关的常见任务以及如何使用 AWS Command Line Interface 执行这些任务。

主题

创建主题

以下命令创建名为 **my-topic** 的主题：

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

```
}
```

记下 TopicArn，您随后将用它来发布消息。

订阅主题

以下命令使用电子邮件协议和通知终端节点的电子邮件地址来订阅主题：

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

电子邮件将发送到 subscribe 命令中列出的电子邮件地址。电子邮件包含以下文本：

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

单击确认订阅后，“已确认订阅！”通知消息在浏览器中显示的信息应类似于以下内容：

```
Subscription confirmed!

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

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

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

向主题发布

以下命令向主题发布消息：

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

包含文本“Hello World!”的电子邮件将发送到 emailusername@example.com

取消订阅主题

以下命令取消主题订阅：

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

要验证是否取消了主题订阅，请键入以下内容：

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

删除主题

以下命令删除主题：

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

要验证是否删除了主题，请键入以下内容：

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

将 Amazon Simple Workflow Service 与 AWS Command Line Interface配合使用

您可以使用 AWS CLI 访问 Amazon Simple Workflow Service (Amazon SWF) 的功能。

有关命令列表以及如何在 Amazon SWF 中使用域的信息，请参见以下主题。

主题

按类别列出 Amazon SWF 命令

本节在 AWS CLI 中列出 Amazon SWF 命令的参考主题。这里的命令是按功能类别 列出的。

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
```

要获取特定命令的帮助，请在命令名称后使用 `help` 指令。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 Command Line Interface处理 Amazon SWF 域 (p. 82)。

工作流程执行管理

- 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 Command Line Interface处理 Amazon SWF 域 (p. 82)。

任务列表可见性

- count-pending-activity-tasks
- count-pending-decision-tasks

使用 AWS Command Line Interface处理 Amazon SWF 域

本节介绍如何使用 AWS CLI 执行常见 Amazon SWF 域任务。

主题

列出域

要列出已为您的账户注册的 Amazon SWF 域，可以使用 `swf list-domains`。只有一个必需参数：`--registration-status`，可将此参数设置为 `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. 85)。您可能会想到，它会返回您拥有的任何已弃用的域。

设置页面大小以限制结果

如果您有许多域，则可以设置 `--maximum-page-size` 参数以限制返回的结果数目。如果您获取的结果多于指定的最大数目，那么您将收到一个 `nextPageToken`，您可将其发送到对 `list-domains` 的下一个调用以检索其他条目。

下面是使用 `--maximum-page-size` 的示例：

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

Note

返回给您的 `nextPageToken` 的长度要大很多。此值只是一个用于说明的示例。

再次执行调用时，如果在 `nextPageToken` 参数中提供 `--next-page-token` 的值，那么您将会得到另外一页结果：

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

当没有要检索的其他结果页时，nextPageToken 不会在结果中返回。

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，将不会保留工作流程执行数据。

下面是注册新域的示例：

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

注册域时，不会返回任何内容 ("")，但您可以使用 swf list-domains 或 swf describe-domain 查看新域。例如：

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

```
    ]
}
```

下面是使用 swf describe-domain 的示例：

```
$ 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。

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

您可通过将 --registration-status DEPRECATED 与 list-domains 结合使用来查看已弃用的域。

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

您还可以使用 describe-domain 获取有关已弃用域的信息。

```
$ aws swf describe-domain --name MyNeatNewDomain
{
    "domainInfo": {
        "status": "DEPRECATED",
```

```
        "name": "MyNeatNewDomain"
    },
    "configuration": {
        "workflowExecutionRetentionPeriodInDays": "0"
    }
}
```

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

在使用 pip 进行安装后，您可能需要将 aws 可执行文件添加到操作系统的 PATH 环境变量，或更改其模式以使其可执行。

错误：aws：找不到命令

需要将 aws 可执行文件添加到操作系统的 PATH 环境变量中。

- Windows – 将 AWS CLI 可执行文件添加到命令行路径 (p. 10)
- macOS – 将 AWS CLI 可执行文件添加到命令行路径 (p. 12)
- Linux – 将 AWS CLI 可执行文件添加到命令行路径 (p. 6)

如果 aws 在您的 PATH 中并且您仍然看到此错误，可能是没有正确的文件模式。请尝试直接运行。

```
$ ~/.local/bin/aws --version
```

错误：权限被拒绝

确保 aws 脚本具有可执行的文件模式。例如：755。

运行 chmod +x 可使文件成为可执行文件。

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

错误：AWS 无法验证提供的凭证

AWS CLI 读取凭证的位置可能与您的预期不同。运行 aws configure list 以确认使用的凭证是否正确。

```
$ 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
```

如果正在使用的凭证正确，您的时钟可能不同步。在 Linux, macOS, or Unix 上，运行 data 以检查时间。

```
date
```

如果您的系统时钟已关闭，请使用 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。

错误：调用 *CreateKeyPair* 操作时出现错误 (UnauthorizedOperation)：未授权您执行此操作。

您的 IAM 用户或角色需要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。