

AWS Command Line Interface

Kiran

Table of Contents

[1. AWS Command Line Interface](#)

[1.1. Introduction:](#)

[1.2. Versions:](#)

[1.3. Installation:](#)

[1.4. Configuration:](#)

[1.5. Syntax:](#)

1. AWS Command Line Interface

1.1. Introduction:

Every Cloud Provider offers a Command Line Utility to manage their resources. Without a good CLI utility, it's impossible to automate the administration of cloud resources from Administrators' Laptops or desktops.

The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. With minimal configuration, the AWS CLI enables you to start running commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from the command prompt in your terminal program:

Linux shells: Use common shell programs such as bash, zsh, and tcsh to run commands in Linux or macOS.

Windows command line: On Windows, run commands at the Windows command prompt or in PowerShell or in Git bash or WSL.

Remotely: Run commands on Amazon Elastic Compute Cloud (Amazon EC2) instances through a remote terminal program such as PuTTY or SSH, or with AWS Systems Manager.

All IaaS (infrastructure as a service) AWS administration, management, and access functions in the AWS Management Console are available in the AWS API and CLI. New AWS IaaS features and services provide full AWS Management Console functionality through the API and CLI at launch or within 180 days of launch.

The AWS CLI provides direct access to the public APIs of AWS services. You can explore a service's capabilities with the AWS CLI, and develop shell scripts to manage your resources. In addition to the low-level, API-equivalent commands, several AWS services provide customizations for the AWS CLI. Customizations can include higher-level commands that simplify using a service with a complex API.

AWS's CLI is one of the feature rich and powerful utility to create, read, update and delete the resources in aws accounts. aws cli works in uniform manner across Operating Systems, meaning, no matter which operating system you are using, you run the same commands.

1.2. Versions:

aws cli is currently in Version 2. It's recommended to use the awscli v2. The commands are almost same in both the version 1 and Version 2.

1.3. Installation:

Follow the instructions in the [Installation page](#) to install aws Cli V2

To confirm succesful installation, run the below command in your Command prompt, Git bash or powershell

```
aws --version
```

You should see output similar to the below one:

```
aws-cli/2.1.39 Python/3.8.8 Linux/4.19.0-16-amd64 exe/x86_64.debian.10  
prompt/off
```


1.4. Configuration:

Once the awscli is installed on your laptop or desktop, you need to ensure the awscli has the credentials to authenticate to your aws account and perform the required actions. To be able to use awscli, you should have **Programmatic Access** enabled in IAM for your **user**. You will need the **Access Key ID** and **Secret Access Key** for your user account.

Follow the below steps to enable **Programmatic Access**



1. Login to your aws Account.
2. Go to **IAM** from the Services
3. Click on your user account.
4. Click on Security Credentials

Summary

[Delete user](#)**User ARN** `arn:aws:iam::119077514921:user/devA` **Path** `/`**Creation time** 2021-04-20 22:22 UTC+0530**Permissions****Groups****Tags (1)****Security credentials****Access Advisor**

▼ Permissions policies (1 policy applied)

[Add permissions](#)[+ Add inline policy](#)

Policy name ▼	Policy type ▼	
Attached directly		
 AmazonS3ReadOnlyAccess	AWS managed policy	

▶ Permissions boundary (not set)

▶ Generate policy based on CloudTrail events

1. Click on Create Secret Keys


Access keys

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.

If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. [Learn more](#)

[Create access key](#)

Access key ID	Created	Last used	Status	
AKIARXOMSV2USNN5YHN6	2021-04-28 06:10 UTC+0530	2021-0...	Active	Make inactive 

SSH keys for AWS CodeCommit

Use SSH public keys to authenticate access to AWS CodeCommit repositories. [Learn more](#)

1. Copy the Access Key and Secret key

Create access key

✓ Success

This is the **only** time that the secret access keys can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time.

Download .csv file

Access key ID	Secret access key
AKIAEMCMVWUOBBOWXNA	<div>Show</div>

Close

Once you have the secret and access key, go to command prompt or Git bash and run the following command

```
aws configure
```

You will be prompted to enter **AWS Access Key ID** and **AWS Secret Access Key** as shown below

```
aws configure
AWS Access Key ID [None]: AIzaSyD...vI
AWS Secret Access Key [None]: 3sZNE9GnBIULm0G1QmkxoD74Kf6GzTI4AaP1D7y9
Default region name [None]: us-east-1
Default output format [None]: json
```

1.5. Syntax:

awscli has a very syntax to understand.

aws <command> <subcommand> [-argument1] [value1] [-argument2] [value2]

1.5.1. Examples:

```
aws ec2 describe-instances --region us-east-1
```

In the above command `ec2` is the main command `describe-instances` is the sub command. The `region` is argument and the `us-east-1` is the value.

- Lets' see an example to create a VPC

- Create VPC with a CIDR block on 10.0.0.0/16 in us-east-1 region

```
aws ec2 create-vpc --region us-east-1 --cidr-block "10.0.0.0/16"
```

- Create a name tag for VPC

```
aws ec2 create-tags --resource <VPC_ID> --region $REGION --tags
"Key=Name,Value=$PREFIX-vpc" "Key=Environment,Value=Testing"
```

1.5.2. Query Examples

```
aws ec2 describe-vpcs --region us-east-1 --query 'Vpcs[*].OwnerId' --output
text
```

1.5.3. Filters

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-078343a5084ef8951" \
    --region us-east-1 --query "Subnets[*].SubnetId" \
    --output text
aws ec2 describe-instances --filter Name=tag:Name,Values=dev-server
```

Find VPC with a certain tag

```
aws ec2 describe-vpcs --filters Name=tag:<tagKey>,Values=<tagValue>
```

Example:

Let's filter all the VPCs matching the Tags Environment="Testing"

```
aws ec2 describe-vpcs --filters Name=tag:Environment,Values=Testing
```