Day – 4

Pre-requisites: Day 1, Day 2 & Day3

**Agenda:**

What is AWS CLI?

How to install AWS CLI on Windows?

How to verify the installation?

Configure AWS CLI from scratch

Create custom AWS profiles

Best Security practices & Environment variables

AWS CLI basics

Automate resource creation using CLI (Real time Examples & Scenarios)

Manage S3



1. **What is AWS CLI?**
2. AWS CLI is an indispensable tool for managing AWS resources efficiently from the command line.

**Command Line Interface**

**AWS CLI**

**Create**

**Manage**

**Delete**

1. Anything you do from console can be done using AWS CLI. If you know this command line, you don’t need to go to console and access various pages page-page to create a single resource.
2. It is a powerful tool which saves lots of time in managing AWS resources.
3. **How to Install AWS CLI on Windows and Linux?**

# AWS Command Line Interface: The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

The AWS CLI v2 offers several [new features](https://aws.amazon.com/blogs/developer/aws-cli-v2-is-now-generally-available/) including improved installers, new configuration options such as AWS IAM Identity Center (successor to AWS SSO), and various interactive features.

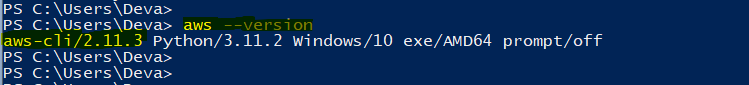
Windows: Download and run the [64-bit](https://awscli.amazonaws.com/AWSCLIV2.msi) Windows installer.

MacOS: Download and run the [MacOS PKG](https://awscli.amazonaws.com/AWSCLIV2.pkg) installer.

Linux: Download, unzip, and then run the [Linux installer](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install)

Amazon Linux: The AWS CLI comes pre-installed on [Amazon Linux AMI](https://aws.amazon.com/amazon-linux-ami/).

Release Notes: check out the [Release Notes](https://aws.amazon.com/releasenotes/CLI) for more information on the latest version.



1. **Why Installing AWS CLI?**

Because we installed this tool, but it should know where we need to run the command and both the authentication, it should get authenticated.

**Access Key**

**Where to run?**

**User**

**AWS CLI**

**Secret Key**

**Authentication**

Basically we have created a user, given access key and secret key using which you authenticate with your AWS account. Apart from that, AWS CLI will know through this user which is the AWS account where it needs to create, deploy and manage the resources.

**AWS CLI**

**API Response**

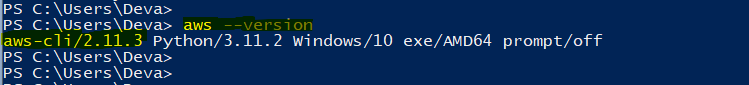
**AWS Account**

**Keys**

**API Request**

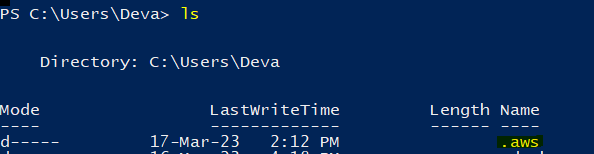
Whenever you hit the aws-cli command, it acts as an API request and using your access keys and secret keys, authenticates with particular AWS account. Once it gets authorized, it returns the API response like if the resources are created or not.

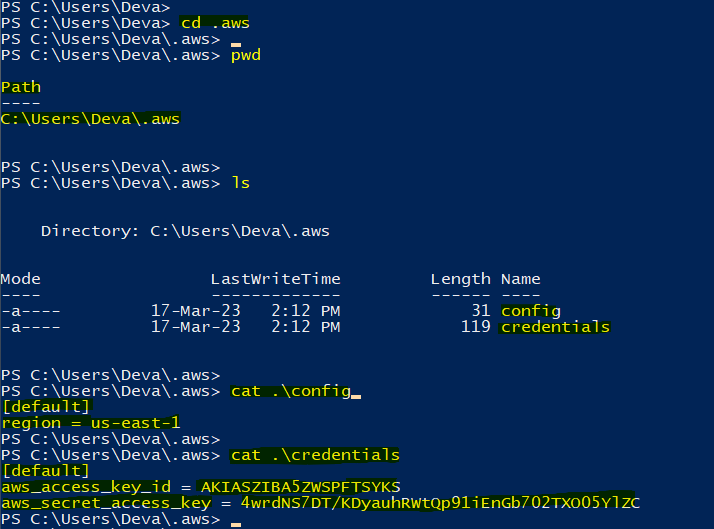
1. **How to verify installations on Windows and Linux?**



1. **Configure AWS CLI from scratch.**
   1. Go to [AWS console](http://console.aws.amazon.com/console/home)
   2. Click on Username on top right corner
   3. Select **Security Credentials** option.
   4. Under Access Keys section, click on **Create access key**.
   5. Enable the check box and click on **Create access key.**
   6. You get a window for **Retrieve access key** option. Copy the **Access key** and **Secret access key** andstore them because its only one time visible.
   7. Click on **Done.**
   8. Go back to the terminal and type **aws configure**
   9. Enter the **Access Key id** and hit enter
   10. Enter the **Secret key** and hit enter
   11. Enter **default region** as **us-east-1** (Select the region where you deploy your resources)
   12. Give **default output format** as json and hit enter.

This will create a AWS profile into our system. We see the profile details below.





1. **How to create custom AWS profiles & use?**
   1. Edit the file **config**

[DevaDevOps]

region =

output =

* 1. Edit the file  **credentials**

[DevaDevOps]

aws\_access\_key\_id =

aws\_secret\_access\_key\_id =

How we can utilize once we configured this custom profile?

We have to, whenever we run any awscli commands, we need pass the argument as below

**awscli --profile DevaDevOps**

1. **Security Best practices & Environment variables**

Securing the AWS profile & Keys

Many people don’t want to store their credentials to local, the other way to setup authentication for awscli i.e, using environment variables.

**Store Credentials**

**.aws**

**Config**

**Credentials**

**Secret Keys**

**Access Keys**

[AWS CLI](https://aws.amazon.com/cli/) 🡪 [Documentation](https://docs.aws.amazon.com/cli/index.html) 🡪 [Configure AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html) 🡪 [Environment Variables](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html)

**Linux or macOS**

$ cd .aws

***export AWS\_ACCESS\_KEY\_ID=AKIAIOSFODNN7EXAMPLE***

***export AWS\_SECRET\_ACCESS\_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY***

***export AWS\_DEFAULT\_REGION=us-west-2***

Setting the environment variable changes the value used until the end of your shell session or until you set the variable to a different value. You can make the variables persistent across future sessions by setting them in your shell's startup script.

**Windows cmd**

$ cd .aws

**To set for all sessions**

***setx AWS\_ACCESS\_KEY\_ID AKIAIOSFODNN7EXAMPLE***

***setx AWS\_SECRET\_ACCESS\_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY***

***setx AWS\_DEFAULT\_REGION us-west-2***

Using [setx](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/setx)to set an environment variable changes the value used in both the current command prompt session and all command prompt sessions that you create after running the command. It does *not* affect other command shells that are already running at the time you run the command. You may need to restart your terminal for settings to load.

**To set for current session only**

Using [set](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/set_1) to set an environment variable changes the value used until the end of the current command prompt session, or until you set the variable to a different value.

$ cd .aws

**set AWS\_ACCESS\_KEY\_ID=*AKIAIOSFODNN7EXAMPLE***

**set AWS\_SECRET\_ACCESS\_KEY=*wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY***

**set AWS\_DEFAULT\_REGION=*us-west-2***

**Windows PowerShell**

$ cd .aws

***$Env:AWS\_ACCESS\_KEY\_ID="AKIAIOSFODNN7EXAMPLE"***

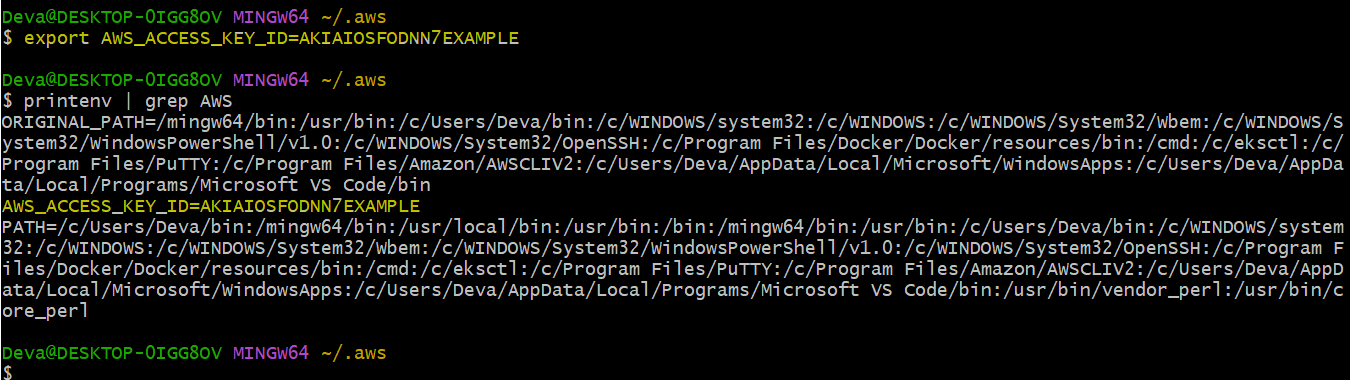
***$Env:AWS\_SECRET\_ACCESS\_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"***

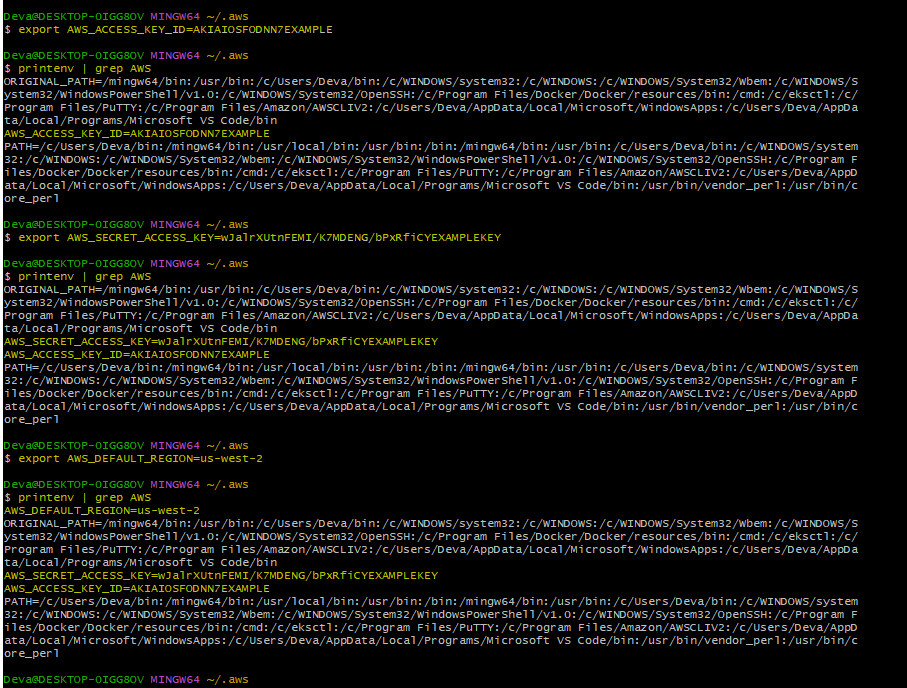
***$Env:AWS\_DEFAULT\_REGION="us-west-2"***

If you set an environment variable at the PowerShell prompt as shown in the previous examples, it saves the value for only the duration of the current session. To make the environment variable setting persistent across all PowerShell and Command Prompt sessions, store it by using the System application in Control Panel. Alternatively, you can set the variable for all future PowerShell sessions by adding it to your PowerShell profile. See the [PowerShell Documentation](https://docs.microsoft.com/powershell/module/microsoft.powershell.core/about/about_environment_variables) for more information about storing environment variables or persisting them across sessions.

Once you have exported the environment variables, to check them execute below command.

**$ printenv | grep AWS**





1. **AWS CLI Basics**

[AWS CLI basics and Command structures](https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-commandstructure.html)

**Command structure:** The AWS CLI uses a multipart structure on the command line that must be specified in this order:

1. The base call to the **aws** program.
2. The top-level **command**, which typically corresponds to an AWS service supported by the AWS CLI.
3. The **subcommand** that specifies which operation to perform.
4. General AWS CLI options or parameters required by the operation. You can specify these in any order as long as they follow the first three parts. If an exclusive parameter is specified multiple times, only the **last value** applies.

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

Parameters can take various types of input values, such as numbers, strings, lists, maps, and JSON structures. What is supported is dependent upon the command and subcommand you specify.

**Amazon S3**

The following example lists all of your Amazon S3 buckets.

$ **aws s3 ls**

2018-12-11 17:08:50 my-bucket

2018-12-14 14:55:44 my-bucket2

For more information on the Amazon S3 commands, see [aws s3](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3/index.html) in the AWS CLI Command Reference.

**AWS CloudFormation**

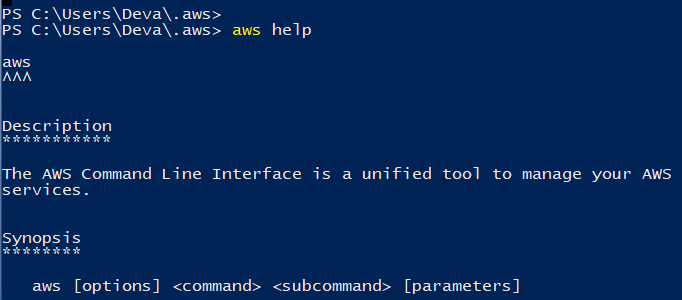
The following [create-change-set](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudformation/create-change-set.html) command example changes the CloudFormation stack name to ***my-change-set***.

***$ aws cloudformation create-change-set --stack-name my-stack --change-set-name my-change-set***

For more information on the AWS CloudFormation commands, see [aws cloudformation](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudformation/index.html) in the AWS CLI Command Reference.

# Follow this link [Use the AWS CLI to work with AWS Services](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-services.html) to know how to use awscli with various AWS services.

# If you do not know the commands using awscli and if you need help, execute aws help command



1. **Automate Resources creation using CLI**

Let’s take an example and we will try to create that infra using automation with AWSCLI

Go to AWS account 🡪 Services 🡪 EC2 🡪 Launch Instance

Try to create a server manually then we can create it with automation.

DevaDevOps 🡪 Ubuntu 🡪 t2.micro 🡪 key pair 🡪 Launch Instance

Let’s do this using AWSCLI

1. [Create AWS key-pair using AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html)

To create a key pair, use the [aws ec2 create-key-pair](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/ec2/create-key-pair.html) command with the ***--query*** option, and the ***--output text*** option to pipe your private key directly into a file.

Linux:

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

For PowerShell, the ***> file*** redirection defaults to UTF-8 encoding, which cannot be used with some SSH clients. So, you must convert the output by piping it to the ***out-file*** command and explicitly set the encoding to ***ascii***.

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

Your private key isn't stored in AWS and can be retrieved **only** when it's created. You can't recover it later. Instead, if you lose the private key, you must create a new key pair.

If you're connecting to your instance from a Linux computer, we recommend that you use the following command to set the permissions of your private key file so that only you can read it.

**$ chmod 400 MyKeyPair.pem**

1. **Display your key-pair**

**$ aws ec2 describe-key-pairs --key-name MyKeyPair**

1. **Delete key-pair**

To delete a key pair, run the [aws ec2 delete-key-pair](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/ec2/delete-key-pair.html) command, substituting *MyKeyPair* with the name of the pair to delete.

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

**Create a key pair**

**$ mkdir awsclitest**

**$ cd awsclitest**

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

**$ ls**

**devops.pem**

Once you hit enter after having above command on terminal, it gets the key pair named devops.pem created in local and aws console in parallel.

To verify, **EC2 🡪 Key Pairs 🡪 devops** you need to loin to console to verify it from awscli itself.

**List all key pairs / Describe specific key pairs**

**$ aws ec2 describe-key-pairs**

Once you hit enter, above command displays all the available ec2 key pairs in the account (the access key and secret key which you have given).

**If I want to print particular pem key, execute below command**

**$ aws ec2 describe-key-pairs –key-name devops.pem**

**Delete a key pair using AWS CLI**

**$ aws ec2 delete-key-pair –key-name devops.pem**

Validate the key pair deletion status using describe command and then you will not be able to find the key pair details. Also in AWS console it gets deleted in parallel.

**Create Update & Delete EC2 Instance Using AWS CLI**

# [Launch, list, and terminate Amazon EC2 instances](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-instances.html)

# Prerequisites

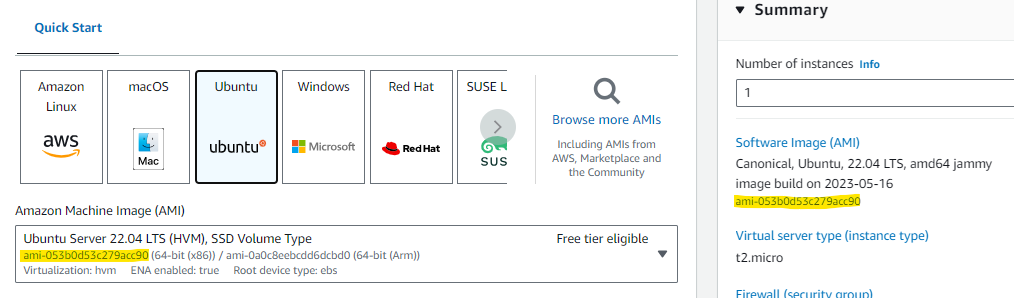
To run the ec2 commands in this topic, you need to:

* Install and configure the AWS CLI. For more information, see [Install or update the latest version of the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) and [Authentication and access credentials](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-authentication.html).
* Set your IAM permissions to allow for Amazon EC2 access. For more information about IAM permissions for Amazon EC2, see [IAM policies for Amazon EC2](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-policies-for-amazon-ec2.html) in the Amazon EC2 User Guide for Linux Instances.
* Create a [key pair](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) and a [security group](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-sg.html).
* Select an Amazon Machine Image (AMI) and note the AMI ID. For more information, see [Finding a Suitable AMI](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html) in the Amazon EC2 User Guide for Linux Instances.

**Launch Instance.**

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

If you do not define security-group-id and subnet-id the command takes the default security group and default subnet.



**$ *aws ec2 run-instances --image-id* ami-053b0d53c279acc90 *--instance-type t2.micro --key-name linux***

Once you hit enter, the API hit by the CLI and you will be able to see each and every single about the server.

Go to AWS console and verify the server if it’s created..

**Add tag to your EC2 Instance.**

**$ aws ec2 create-tags --resources < ec2 instance id > --tags Key=Name,Value=MyInstance**

**List AWS EC2 instances**

**$ aws ec2 describe-instances**

The following command filters the list to only your **t2.micro** instances and outputs only the **InstanceId** values for each match.

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

The following command lists any of your instances that have the tag Name=MyInstance.

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

The following command lists your instances that were launched using any of the following AMIs: ami-x0123456, ami-y0123456, and ami-z0123456.

**$ aws ec2 describe-instances --filters "Name=image-id,Values=ami-x0**

**Update an Existing Resource Using AWSCLI**

Update the instance with its name which was just created above

**$ aws ec2 create-tags --resources i-093efdfb43a213291--tags Key=Name,Value=SonarQube-Server**

**Terminate EC2 Instance**

Terminating an instance deletes it. You can't reconnect to an instance after you've terminated it. As soon as the state of the instance changes to shutting-down or terminated, you stop incurring charges for that instance. If you want to reconnect to an instance later, use [stop-instances](https://docs.aws.amazon.com/cli/latest/reference/ec2/stop-instances.html) instead of terminate-instances. For more information, see [Terminate Your Instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/terminating-instances.html) in the Amazon EC2 User Guide for Linux Instances.

To delete an instance, you use the command [aws ec2 terminate-instances](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/ec2/terminate-instances.html) to delete it.

***$ aws ec2 terminate-instances --instance-ids <ec2 instance id’s>***

**$ aws ec2 terminate-instances --instance-ids i-093efdfb43a213291**

Go to console verify if the instance is terminating.

Next is about handling S3 using AWS CLI, how to manage s3, how to automate, how to upload a file, create a folder, move file from local to S3 and from S3 to local, and various real time scenarios like how we can completely sync a folder to S3 bucket (whatever files put in folder will reflect into S3 bucket) and many more possible☺.

**Break**

**Revise**

1. Manage S3

All the operations we do from Console, we will do same thing using AWS CLI. So that we will come to know how time saver it is going to be. We have already saw how important and powerful AWS CLI is in above topics.

S3 is a cloud storage service. Let’s have an overview of it from AWS console and try to automate those using AWS CLI commands.

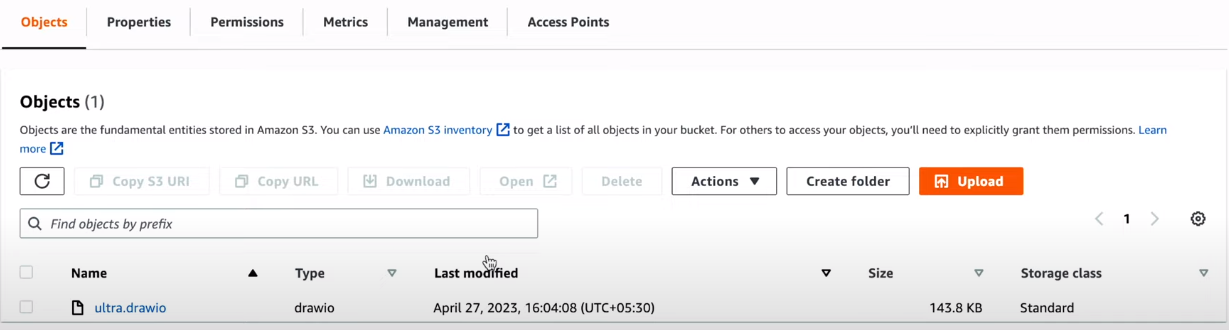
Console 🡪 Services 🡪 S3 🡪 Buckets 🡪 Create Bucket

* Bucket name should be globally unique and shouldn’t contain spaces / uppercase letters.
* Select **Block all public access** option.
* Click on **Create Bucket.**

Once the bucket is creates, click on bucket name and you will be able to see the **Upload** option.

Click on **Upload** and select **Add files.**

Select a file from local and click on **Upload**



You can see the file here. In the same way your Google drive work, like on the mobile device all the photos gets synced with particular storage account, that is also one kind of storage account of Google and in the same way you are having AWS S3 Bucket.

So, like this we can upload our files and folders to S3 bucket. Perform below tasks.

Create a folder

Move the file to the folder

Delete the file from the folder

Delete the S3 bucket

Now, you don’t have any S3 bucket, we will try to achieve the same thing using AWS CLI☺.

Go to the [Documentation of AWS CLI and choose Amazon S3](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-commands.html)

**Prerequisites**

To run the s3 commands, you need to:

* Install and configure the AWS CLI. For more information, see [Install or update the latest version of the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) and [Authentication and access credentials](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-authentication.html).
* The profile that you use must have permissions that allow the AWS operations performed by the examples.
* Understand these Amazon S3 terms:
  + **Bucket** – A top-level Amazon S3 folder.
  + **Prefix** – An Amazon S3 folder in a bucket.
  + **Object** – Any item that's hosted in an Amazon S3 bucket.

**List buckets and objects**

**$ aws s3 ls**

**Create a Bucket**

**$ aws s3 mb <target> [--options]**

The following example creates the s3://bucket-name bucket.

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

To verify if the S3 bucket is created, use **aws s3 ls** command and also you can check in console.

If you want to list the contents of particular bucket **aws s3 ls <name of the bucket>**

**$ aws s3 ls s3://devadevops**

**Copy a file from local to S3 bucket**

$ aws s3 cp <source> <target>

**$ aws s3 cp linux.pem s3://devadevops/devops**

Verify the upload status using **$ aws s3 ls s3://devadevops**

**Sync local folder to AWS S3 bucket**

Folder

Local device

Files

S3 Bucket

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

**$ aws s3 sync awsclitest s3://devadevops/awsclitest**

**$touch demo.txt**

**$ aws s3 sync awsclitest s3://devadevops/awsclitest**

**$ rm demo.txt**

**$ aws s3 sync awsclitest s3://devadevops/awsclitest**

The deleted file is not synced and it is expected behavior. To sync the deleted file (local) also to S3, you have to provide the option **--**delete 🡪 sync deleted local file to S3

**$ aws s3 sync awsclitest s3://devadevops/awsclitest** **--delete**

To recover the locally lost file, from S3 bucket 🡪 local file is deleted

**$ aws s3 sync s3://devadevops/awsclitest/ awsclitest**

To delete a file from S3

**$ aws s3 rm s3://devadevops/awsclitest/linux.pem**

To sync to local with deletion of object from S3 bucket

**$ aws s3 sync awsclitest s3://devadevops/awsclitest/ --delete**

Move a file from local to S3 bucket

**$ aws s3 mv linux.pem s3://devadevops/awsclitest**

Move a file from S3 bucket to local

**$ aws s3 mv s3://devadevops/awsclitest/linux.pem awsclitest**

Delete the S3 bucket, before deleting, move the file back to S3 bucket.

**$ aws s3 rb <target> [--options]**

The following example removes the s3://bucket-name bucket.

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

By default, the bucket must be empty for the operation to succeed. To remove a bucket that's not empty, you need to include the ***--force*** option. If you're using a versioned bucket that contains previously deleted—but retained—objects, this command does not allow you to remove the bucket. You must first remove all of the content.

The following example deletes all objects and prefixes in the bucket, and then deletes the bucket.

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

Go to console and check S3 bucket will be removed.