**Module Overview**

Welcome to the module on **‘Working with AWS’**.

**In this module**

You were introduced to the basic concepts of machine learning and cloud in the previous modules. Now, in the forthcoming video, we will see what this module covers.

Play Video

This module will help you work on one of the most used cloud platforms, **Amazon Web Services (AWS)**, and introduce you to the application of the different services offered by AWS. You will learn how to use these tools and services, and, later, you will work on a case study to build an application using these tools and services. By the end of this module, you should be able to select appropriate services to build such applications on your own. You can refer to the diagram below to get a gist of all the concepts covered in this module.



**Important note:** The previous modules mainly involved the introduction of the concepts. This module, on the other hand, will involve a lot more hands-on elements. Please keep a note of the following elements while going through the module:

* Being a tool-oriented module, it will require more time in comparison with the other modules. Make sure that you devote adequate time to understand and complete all the concepts covered in this module.
* Make sure you use your laptop or computer to complete this module. The module contains multiple demonstrations and uses different tools, which cannot be accessed from mobile phones.
* You must not wait till the last moment to start the module as issues may take a day or two to resolve. Therefore, try and use your time accordingly.
* Pay attention to each step while implementing the task or functions to avoid any errors. In case of any error, you might end up using more than the mentioned time due to troubleshooting.
* Implement all the steps at your end to get a clear understanding of the entire process. This is important as you will be tested on the same throughout the course.
* You will work with different AWS services which are not free. You will be working with a limited balance, and hence, none of them should be left idle. You must stop or terminate all the resources after they have served their purposes.

**In this session**

You will be introduced to a new tool to interact with the AWS platform: **AWS Command Line Interface (CLI)**. When dealing with a large user base, this is a much faster way to work on AWS than the Management Console. You will learn more about it in the segments to come. In this session, we will also discuss some advanced topics related to the different AWS services that were covered in the previous module. Going deeper into these services will help you better understand how an actual deployment happens in the Cloud.

This segment requires you to have basic knowledge of Linux shell commands. In case you don’t know these commands, please refer to the document provided below before proceeding.

**[Basic Linux Commands](https://cdn.upgrad.com/UpGrad/temp/f1df8667-71b9-4cd6-9d8a-31217ea504c7/Basic_Linux_Commands.pdf" \o "Basic_Linux_Commands.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/f1df8667-71b9-4cd6-9d8a-31217ea504c7/Basic_Linux_Commands.pdf" \o "Basic_Linux_Commands.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/f1df8667-71b9-4cd6-9d8a-31217ea504c7/Basic_Linux_Commands.pdf" \o "Basic_Linux_Commands.pdf" \t "_blank)**

Now, try to answer the questions below to test your understanding. These questions are based on Linux shell commands. If you are unable to answer even a single question, then please refer to the documentation provided above.

keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:4/4

Mandatory

**Linux Commands**

Match the entries in Column A with the correct options in Column B.

| **Column A** | **Column B** |
| --- | --- |
| 1. Exiting the cat command without saving the changes made to the file | 1. (Ctrl+D)\*2 |
| 1. Exiting the cat command after saving the changes made to the file | 1. Esc + :q! |
| 1. Entering the edit mode inside the VI editor | 1. Esc+:wq! |
| 1. Exiting the VI editor without saving the changes made to the file | 1. i |
| 1. Exiting the VI editor after saving the changes made to the file | 1. Ctrl + C |

Top of Form



1 - i, 2 - ii, 3 - iii, 4 - iv, 5 - v



1 - i, 2 - v, 3 - iv, 4 - iii, 5 - ii



1 - v, 2 - i, 3 - iv, 4 - iii, 5 - ii



**1 - v, 2 - i, 3 - iv, 4 - ii, 5 - iii**

**Feedback :**

*These commands will be helpful in the coming segments. Make sure you are well versed with them.*

Bottom of Form

**AWS CLI: Introduction**

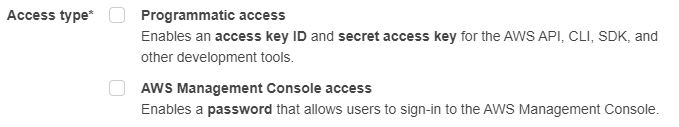
In the previous module, you learnt how to access AWS services from the AWS web platform or the Management Console. Now, we will begin this segment with the forthcoming video where Vinod will talk about another way to work with all of these services.

Play Video

So, as mentioned in the video, there are two ways to access AWS services:

1. AWS Management Console Access
2. Programmatic Access

These are discussed briefly in the image given below.



You have already accessed different services through the Management Console. Now, you will use the programmatic access through the **AWS Command Line Interface (AWS CLI)** for the same purpose. As per the documentation,

“*The AWS Command Line Interface is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.*”

In the AWS CLI, instead of relying on the web platform, you work with the AWS services through the local command-line environment on the machine. It comes with its own syntax and commands, which you will learn in the following lectures.

The video also discussed some of the merits that the CLI has over the Management Console. One of the key features of the AWS CLI that was highlighted is its ability to save time. To reiterate:

* The AWS CLI gives you control over all the services that AWS offers through the Management Console. Any service that is launched on the Management Console is made available through the CLI at the time of launch or within **180 days**of launch. To see this in action, you will work with all the services that were covered in the previous module through the AWS CLI as part of this module.
* The AWS CLI gives you control of all the services on one platform, thus saving you from visiting the individual pages of each service to execute any task. For example, to access the S3 bucket from an EC2 instance, you will have to visit web pages for two different services: First, the IAM page to create a new role that allows S3 access, and, second, the EC2 page to attach the created role to the required EC2 instance. All this can be done through a few lines of code using the AWS CLI. This is depicted in the image given below. However, in some cases, the management console could be the better option.



* Since the AWS CLI is a programming console, it also comes with the ability to automate tasks through scripts. You can easily reduce the time spent on writing code for each task that you were expected to perform manually on the management console. This also reduces the scope of errors that can be committed while executing the operations manually.  
    
  To understand this better, think of a data analytics firm that collects consumer data daily and uses certain parameters from the collected data to generate reports. Now, with the help of scripts, the firm can automate the task to collect the required data from S3, instead of manually fetching the data daily.

Hence, you can easily say that the AWS CLI is a better tool to work with the cloud services offered by AWS in certain conditions. As part of this session, you are expected to work with the AWS CLI interface wherever possible. Now, try to solve a few problems based on your learning.

keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:1/2

Mandatory

**AWS CLI**

The AWS CLI serves as a better tool than the AWS Management Console when you are working in the Cloud environment. Which of the following options is true and does not support this statement?

Top of Form



The Management Console saves time, as you can run all the operations through a series of clicks.



The Management Console is better as it gives access to all the features, which may not be provided in the CLI.



The Management Console lists the operations associated with a service in a single space.

Bottom of Form

1. *While working with the CLI, you can easily access all the resources without the hassle of going to the AWS website and entering credentials to log on to the platform. You can list all the objects by simply running a single command over the terminal. The online session for AWS expires after a period of inactivity. However, your CLI credentials stay in the configure file until you remove them.*
2. *With a new feature, you must first try to follow a guided approach. It is done under the Management Console.*
3. *While working with the CLI, you can easily work with resources without going to the web page and entering all the details. You can simply run one command in the command line and you will have the objects listed for you.*
4. *As you have learnt above, attaching a role involves visiting multiple pages, and, hence, you must prefer working on the AWS CLI instead of the AWS Management Console.*

**AWS CLI: Installation**

The AWS CLI serves as a powerful tool when working with AWS resources. It allows control over all the AWS services from the local command-line shell on your machine. Let us now proceed with the installation process for the AWS CLI.

**Note:** The video below covers the installation process for macOS. You can refer to the document provided after the video to learn how to download the AWS CLI on a Windows-based machine after you have watched the video.

Play Video

The installation guide for Windows is provided below.

**[AWS CLI Installation - Windows](https://cdn.upgrad.com/uploads/production/83e4cb08-293b-4819-9b40-e58a17426d78/AWS+CLI+installation+on+Windows+-+MSI.pdf" \o "AWS CLI installation on Windows - MSI.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/uploads/production/83e4cb08-293b-4819-9b40-e58a17426d78/AWS+CLI+installation+on+Windows+-+MSI.pdf" \o "AWS CLI installation on Windows - MSI.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/uploads/production/83e4cb08-293b-4819-9b40-e58a17426d78/AWS+CLI+installation+on+Windows+-+MSI.pdf" \o "AWS CLI installation on Windows - MSI.pdf" \t "_blank)**

You can refer to the instructions provided below before proceeding with the documentation.

AWS CLI comes in two different versions:

* **Version 1.x**(Used in the module)
* **Version 2.x**(Latest)

To learn more about the different versions, you can refer to the official [user guide](https://docs.aws.amazon.com/cli/latest/userguide/aws-cli.pdf) provided by AWS. Now, moving on to the installation, there are two prerequisites for Version 1.x:

* Python package (version 2 above 2.7 and version 3 above 3.4)
* pip package

PIP is useful in installing and managing the Python packages. Both the packages come preloaded with Anaconda, which you installed as part of the preparatory content. However, it is advised that you check for the PIP package before proceeding.

To do so, you can run the following command: **pip --version**. This will return with the version of the PIP package if it exists on your machine. You can also check for pip3, as you were working with Python3 prior to this. In case the package has to be installed, you can refer to the documentation provided below.

**[pip Installation Guide](https://cdn.upgrad.com/UpGrad/temp/6360e287-7c51-44b2-a12a-0c862eed045b/PIP+Installation+Guide.pdf" \o "PIP Installation Guide.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/6360e287-7c51-44b2-a12a-0c862eed045b/PIP+Installation+Guide.pdf" \o "PIP Installation Guide.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/6360e287-7c51-44b2-a12a-0c862eed045b/PIP+Installation+Guide.pdf" \o "PIP Installation Guide.pdf" \t "_blank)**

Once all the required packages are present, you can install the AWS CLI on your machine using the instructions provided above. Note that you must update the CLI package regularly in order to support the latest features added to it. To learn how to do this, you can refer to the user guide provided above.

So, now that you have installed the AWS CLI, in the forthcoming video, Vinod will explain how to access the CLI on the local machine.

Play Video

To access AWS CLI from your local terminal, you need the access key pair. The access key pair acts as the credentials for you to log in to the AWS CLI through the local terminal. There are two components associated with it:

* Access key ID (user ID)
* Security access key (password)

Therefore, first, you must create the access key pair to access AWS through the CLI. These credentials must be safely stored and **must not be shared**, as they give complete control of your AWS account through the CLI. Now, in the next video, you will learn how to use them to run AWS CLI on your local machine.

**Important note:** In the demonstrations, the region has been selected as ‘ap-south-1’. But you are advised to work with the **North Virginia** region only. The region code for that region is ‘**us-east-1**’. Make sure you use the same region throughout the program.

Play Video

Once you have provided all the required details, you are ready to access all the AWS services from the command line.

To access the ‘*credentials*’ file, Windows users need to type the following command in the command prompt:

notepad "./.aws/credentials"

If you are a Windows user, then you must provide the application name (notepad) and the path of the file (“./.aws/credentials”) to open a file through the command prompt.

The steps mentioned in the video have been summarised in the document provided below.

**[Accessing the AWS CLI](https://cdn.upgrad.com/uploads/production/75ce2418-f15a-4c4c-b3db-c7c6899a66ea/Accessing+AWS+CLI.pdf" \o "Accessing AWS CLI.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/uploads/production/75ce2418-f15a-4c4c-b3db-c7c6899a66ea/Accessing+AWS+CLI.pdf" \o "Accessing AWS CLI.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/uploads/production/75ce2418-f15a-4c4c-b3db-c7c6899a66ea/Accessing+AWS+CLI.pdf" \o "Accessing AWS CLI.pdf" \t "_blank)**

So, to log in to AWS through the AWS CLI, you can simply run the following command:

**aws configure**

Questions:1/1

Mandatory

**AWS CLI**

What are the different output formats available when working with the AWS CLI?

Top of Form



Table



Text



Image



JSON

Bottom of Form

**AWS CLI: IAM**

In the previous segment, you learnt how to create an access key pair and use it to log in to the AWS CLI. Now, the next step is to start using the AWS CLI to access different services through the local terminal. Since you will be working with a new tool, let us watch the forthcoming video and try to understand its syntax first.

Play

Mute

Current Time 0:00

/

Duration 5:45

Loaded: 11.49%

1.5x

Playback Rate

Quality Levels

Picture-in-PictureFullscreen

So, as mentioned in the video, you can easily access different AWS services using the following code structure:

**aws <service> <operation>**

You can replace the elements based on the service and the operation that you wish to execute. To make things easier, AWS provides a list of functions along with their parameters under the same environment. To access them, you can simply use the help attribute along with the code. AWS provides the same documentation online as well. It is easier to use the online version because of the user-friendly interface. You can access it using this [link](https://docs.aws.amazon.com/cli/latest/index.html).

The code used in the video above is provided below:

**#** Help command

aws iam create-user help

**#** IAM Commands

**#** Creating an IAM user

aws iam create-user --user-name user1

keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:2/2

Mandatory

attach-user-policy

**Feedback:**

To attach a policy to a user, you can use the following command:

aws iam attach-user-policy --user-name user1 --policy-arn arn:aws:iam::aws:policy/Policy

**AWS CLI: IAM**

Which of the following is the correct command to assign the user ‘user1’ the following permission: [AdministratorAccess](https://console.aws.amazon.com/iam/home?region=ap-south-1" \l "/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAdministratorAccess" \t "_blank)?

**Note:** For your aid, the policy ARN attached in all the options is correct. However, you must check the ARN format when working with a new policy.

Top of Form



aws attach-user-policy --user-name user1 --policy-arn arn:aws:iam::aws:policy/AdministratorAccess



aws iam attach-policy --user-name user1 --policy-arn arn:aws:iam::aws:policy/AdministratorAccess



aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/AdministratorAccess



**aws iam attach-user-policy --user-name user1 --policy-arn arn:aws:iam::aws:policy/AdministratorAccess**

**Feedback :**

*All the details to attach the policy are specified correctly in this option.*

**Correct**

Bottom of Form

**done**Your answer is**Correct.**

Continue

Attempt 1 of 2

**Note:**  
If you want the user ‘user1’ to access all the services through CLI, then you need to first run the correct command mentioned in Question 2 above. You can change the policy ARN in case you want to restrict the permissions to a specific service. You can find the ARNs for different policies under the **Policies**section of the Amazon IAM service.

Now, let’s see how to perform different IAM operations covered in the last module using AWS CLI. You will have to extract the following files in the directory used in the shell environment in order to execute the commands before proceeding with the demonstration. To find the path in the command shell, use dir for Windows and pwd for Linux/macOS. Open the obtained path in your file browser and extract the files there.

**[JSON Files](https://cdn.upgrad.com/UpGrad/temp/ecd1b2fd-cce9-43a6-a0b0-26234d49be70/JSON+Files.rar" \o "JSON Files.rar" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/ecd1b2fd-cce9-43a6-a0b0-26234d49be70/JSON+Files.rar" \o "JSON Files.rar" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/ecd1b2fd-cce9-43a6-a0b0-26234d49be70/JSON+Files.rar" \o "JSON Files.rar" \t "_blank)**

You can also download the individual JSON files directly from the links provided below.

**[test.json](https://cdn.upgrad.com/uploads/production/02eb8f31-8d49-4703-8757-6bb837c04860/test.json" \o "test.json" \t "_blank)**

[file\_download](https://cdn.upgrad.com/uploads/production/02eb8f31-8d49-4703-8757-6bb837c04860/test.json" \o "test.json" \t "_blank)**[Download](https://cdn.upgrad.com/uploads/production/02eb8f31-8d49-4703-8757-6bb837c04860/test.json" \o "test.json" \t "_blank)**

**[test\_role.json](https://cdn.upgrad.com/uploads/production/731e2aca-d792-4e5b-bd49-d8b3f2248a19/test_role.json" \o "test_role.json" \t "_blank)**

[file\_download](https://cdn.upgrad.com/uploads/production/731e2aca-d792-4e5b-bd49-d8b3f2248a19/test_role.json" \o "test_role.json" \t "_blank)**[Download](https://cdn.upgrad.com/uploads/production/731e2aca-d792-4e5b-bd49-d8b3f2248a19/test_role.json" \o "test_role.json" \t "_blank)**

Play Video

So, the key learning from the videos above is that you can perform all the actions of the Management Console through the AWS CLI. To summarise the tasks performed in the CLI, you learnt how to:

* Create a new user,
* Create a new group,
* Add a user to a group,
* Create a policy and a role from a JSON file, and
* Attach a policy to the role using the JSON file and the policy ARNs.

The code used in this above video is provided below.

**#** IAM Commands

**#** Creating a group

aws iam create-group --group-name group1

**#** Adding user to the group

aws iam add-user-to-group --user-name user1 --group-name group1

**#** Fetching the group details

aws iam get-group --group-name group1

**#** Opening the json file that stores the policy to allow S3 access

vi test.json

**#** Creating a policy using the json file

aws iam create-policy --policy-name policy1 --policy-document file://test.json

**#** Make sure that you copy the policy ARN here as it will be used later

**#** Opening the json file that stores the role that allows EC2 to assume the Role

vi test\_role.json

**#** Creating a role using the json file

aws iam create-role --role-name role1 --assume-role-policy-document file://test\_role.json

**#** Attaching the poicy to the created role

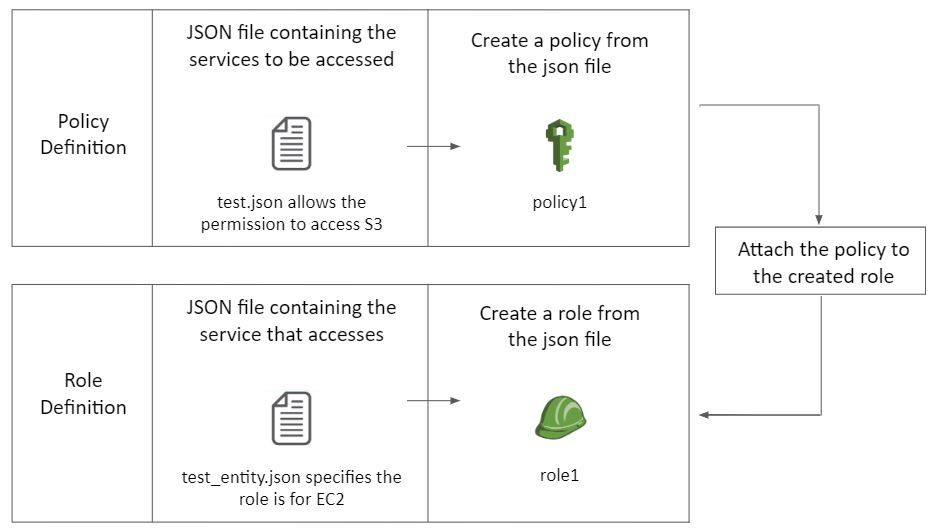
aws iam attach-role-policy --role role1 --policy-arn paste\_the\_arn\_copied\_above

**#** Checking the created role

aws iam list-attached-role-policies --role-name role1

You can also create your own policies by saving the files as JSON on Notepad or Notepad++. However, you must run your file through a JSON validator before using it in the production. One such validator is attached [here](https://jsonlint.com/).

Here is a diagram explaining the steps to attach a policy to a role.



keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:1/2

Mandatory

Top of Form

**AWS CLI**

How does Amazon identify the resources that are available on the AWS Cloud network?

**!Note:**Once submitted, answer is not editable.



Bottom of Form

Submit

Word Limit**10-100**

Word Count**0**

So, in this segment, you learnt how to work with the AWS IAM service using the CLI. The next service that you will work with is Amazon S3

**Suggested Answer**

AWS tracks all the resources on its platform through Amazon Resource Names, or ARNs. ARNs help with uniquely identifying all the resources across all the users and services available on the platform through their unique structure, **arn:partition:service:region:account-id:resource-type/resource-id**

**Suggested Answer**

Follow the steps below to create a role for an EC2 instance:

1. Create a JSON file that stores the policy to allow the EC2 instance to assume a role
2. Using the JSON file created in Step 1, run the following command to create a role:  
   aws iam create-role --role-name role1 --assume-role-policy-document file://file.json
3. Now, you need to attach a policy to the created role. In the previous case, we created the JSON file ‘test.json’ to define the policy. Since AWS has the inbuilt policy “AWSRDSFullAccess” to provide complete RDS access, you can directly attach it to the created role. Here, you can use the ARN of the policy to attach it to the role using the following command:  
   aws iam attach-role-policy --role-name role1 --policy-arn arn:aws:iam::aws:policy/AWSRDSFullAccess
4. The role is now ready to be attached to the EC2 instance. We will see how to attach it to an EC2 instance when we work with EC2 through the CLI.

**AWS CLI: S3**

Having worked with the IAM service, you will now use the AWS CLI to perform different functions associated with S3, the storage service of AWS.

Play

Mute

Current Time 2:51

/

Duration 6:29

Loaded: 61.12%

1.25x

Playback Rate

Quality Levels

Picture-in-PictureFullscreen

**Note:** In case there are any spaces in the file path, make sure that you put it in double-quotes.

The code used in the video is provided below.

**#** S3 Commands

**#** Creating a bucket - mb command

aws s3 mb s3://unique\_bucket\_name

**#** Listing the buckets in S3 - ls command

aws s3 ls

**#** Copying a file from local machine to S3 bucket - cp command

aws s3 cp ./test.json s3://unique\_bucket\_name/file\_name.json

**#** Listing the objects in the bucket - ls command

aws s3 ls s3://unique\_bucket\_name

**#** Removing a file from the bucket - rm command

aws s3 rm s3://unique\_bucket\_name/file\_name.json

**#** Removing all the files from the bucket - rm command

aws s3 rm --recursive s3://unique\_bucket\_name

**#** Removing the bucket - rb command

aws s3 rb s3://unique\_bucket\_name

You can explore all the commands available for Amazon S3 using the help attribute. You will find the following commands under the service:

* **cp:** To copy objects. It can be used to upload and download files from S3 to the local machine
* **ls:** To list the buckets/objects in S3
* **mb:** To create a new S3 bucket
* **mv:** To move the objects present on the local disk or in S3
* **presign:** To provide a pre-signed URL for an Amazon S3 object
* **rb:** To remove an empty bucket
* **rm:** To remove an object
* **sync:** To sync the directories present in S3
* **website:** To set the website configuration for a bucket

There are multiple attributes associated with each command that you can explore in the help section. Let’s take a look at a few of them using the questions provided below.

keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:1/3

Mandatory

**AWS CLI: S3**

The ‘**rb**’ command helps you to delete an empty bucket. How will you delete a bucket that contains multiple files and folders?

Top of Form



aws s3 rb --recursive s3://bucket\_name



aws s3 rb --force s3://bucket\_name



aws s3 rb --f s3://bucket\_name



You have to empty the bucket first in order to delete it.

Bottom of Form

Submit

Attempt 1 of 2

As you can see in the examples above, the AWS CLI is quite a useful tool to work with Cloud services. You can code your requirements using the features provided and execute the tasks, which would, otherwise, have been very difficult.

In the next segment, we will take a break from CLI and you will learn more about the different storage classes associated with Amazon S3.

aws s3 rb --force s3://bucket\_name

**Feedback :**

*Using the --force attribute, you can remove a bucket that contains multiple objects.*

**Suggested Answer**

You can use the Exclude and Include filters to obtain the required objects. To download all the files associated with a particular category, you must use the --exclude and --include attribute together with the cp command under S3. Also, to check each file in the bucket, you have to use the recursive function:

aws s3 cp s3://records “local\_path” --recursive --exclude “\*” --include “electronics\*”

Through this task, you must have realised the power of the AWS CLI and also learnt how it is better than the Management Console in performing challenging tasks.

**S3: Storage Classes - I**

By now, you are aware of the commands that can help you access Amazon S3 through the AWS CLI. You can perform all the basic functions, such as create, delete, move and copy. Now, in this segment, you will see the implementation of this service in a real scenario.

In the production environment, you must remember that all the AWS services that you use are chargeable after a free tier. The applications or the resources of any company cannot be sustained only on the free tier. In the case of Amazon S3, the cost is calculated based on the following parameters:

1. The region where the S3 is located,
2. The size of the data stored,
3. The duration of the data in S3 and
4. The object access request.

Now, to make the service cost-effective, AWS offers the feature of **Storage Classes**, which you were introduced to in the previous module on ‘Introduction to Cloud’. We will try to understand this feature in detail in the forthcoming video.

Pause

Mute

Current Time 0:18

/

Duration 2:33

Loaded: 45.10%

1.25x

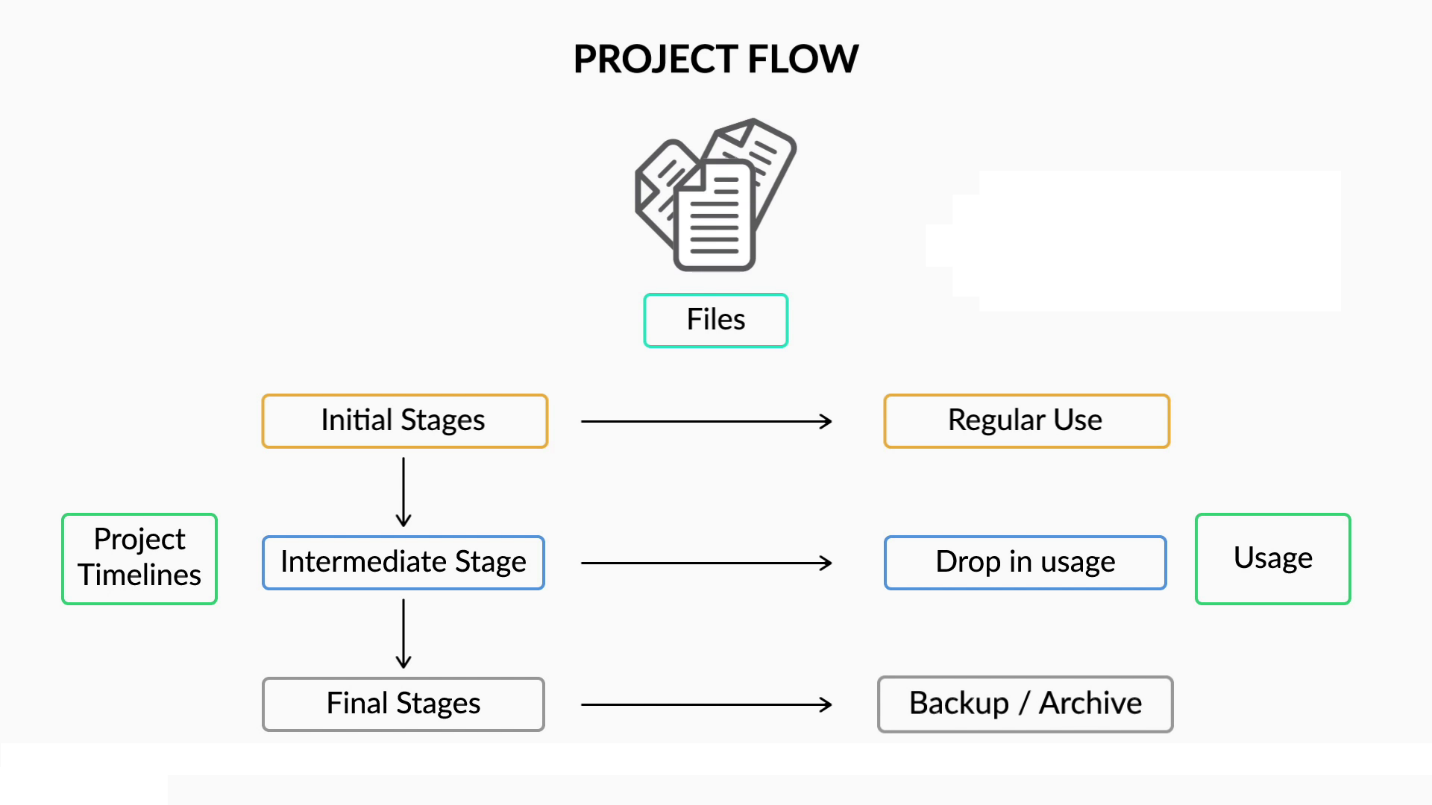
Playback Rate

Quality Levels

Picture-in-PictureFullscreen

2041292

So, as you saw in the video, the usage or the frequency of accessing a file, either on a local machine or in the Cloud environment, varies from one file to another. Some files are accessed daily, whereas others are opened only when required. The usage pattern of a file can also change over time, as mentioned.



Since Amazon S3 is a paid service, you would want to optimise the storage as much as possible. In the next video, you will learn how to do this in the Cloud.

Play Video

**Note:** At [01:23], Vinod mentions that Object Availability is the percentage of a 1-year period for which the file is inaccessible. That is not correct. It tells the percentage of a 1-year period for which the file will be **accessible**.

AWS provides you the service to store data based on the usage pattern and charges you accordingly. Depending upon the budget and the access frequency, you can select from the provided storage classes. All the storage classes that you saw above deal with files that are accessed repeatedly.

However, you may also require the storage service just for backup or to archive files. In that case, you would not be interested in paying the same amount as you pay for the accessed data. Storing the data on external hard drives or on spare machines would be cheaper than paying for the aforementioned storage classes. However, you will then face the hassle of maintaining and tracking all of these external resources.

In the forthcoming video, we will see how AWS provides a solution to overcome this issue.

Play Video

AWS provides a very useful and cost-effective solution to back up or archive data. The table below summarises all the storage classes discussed above.



**Note:** Open the image in a new tab and zoom in for clarity.

Note that the objects stored in S3 Glacier and S3 Deep Archive do not allow real-time access and have to be restored first before accessing data. The restored object is a temporary copy, which expires after a specified time. Hence, from the image above, you can see that object availability for these classes is 99.99% after the files have been restored. To get more clarity on this, you can refer to [this](http://docs.aws.amazon.com/AmazonS3/latest/dev/restoring-objects.html) link.

So, you can now choose the most suitable storage class based on your usage and the allocated budget. You can check the pricing of the different storage classes [here](https://aws.amazon.com/s3/pricing/). Now, based on the concepts that you have learnt above, try to answer the questions provided below.

keyboard\_arrow\_leftkeyboard\_arrow\_rightQuestions:1/2

Mandatory

**Amazon S3: Storage Classes**

Which is the most suitable storage class when you are working on a very important file that is accessed only once or twice a year?

Top of Form



S3 Standard



S3 Standard IA



S3 Glacier



S3 Intelligent Tiering

Bottom of Form

Submit

Attempt 1 of 2

So, now that you have a basic understanding of storage classes, in the next segment, you will learn how to set the storage class for any bucket or for an object.

**1 out of 200 files stored in S3 may be lost.**

**Feedback :**

*99.5% means 995 out of 1,000 files will not be lost.*

# S3: Storage Classes - II

In this segment, you will learn how to change the storage class of an object stored in Amazon S3. You must know that the default storage class of any object, when uploaded to S3, is **S3 Standard**. So, the object is placed under the most expensive option by default. Therefore, to prevent unnecessary costs, you must change the storage class of the object accordingly. In the forthcoming video, you will learn how to do that.

Play Video

**Note:** In the newer versions of AWS, the Change storage class option has been renamed to edit storage class.

Now you can easily move an object from one storage class to another manually. Apart from this, you can also automate the entire process with the help of **Object Life Cycle**. This is a very useful feature, and it helps reduce the risk of unnecessary costs in case someone forgets to move the file to an appropriate storage class.

Under life-cycle configuration, you can define a set of rules that automatically delete or move an object from a storage class after the defined time. So, there are two types of actions:

* **Transition actions:** To define the transition from one storage class to another
* **Expiration actions:** To define the expiry (auto-deletion) of an object

It is always advised that you develop the habit of defining an object life cycle at the time of upload. You should always keep the following points in mind when defining an object life cycle:

* You must have a good understanding of the usage pattern of the file for which the object life cycle is being defined. For example, if a file is deleted based on the expiration action, then you will not be able to retrieve it.
* You must be careful about the costs associated with both transition and expiration actions. These are paid services, and the cost varies based on the size of the object, the storage class and the duration specified in the rules.

Questions:1/1

Mandatory

**Amazon S3: Storage Classes**

Which of the following is not a suitable storage class when you want to store data in the region **ap-northeast-1**?

Top of Form



S3 Standard



S3 Intelligent Tiering



S3 One Zone-IA



S3 Standard-IA

Bottom of Form

Submit

Attempt 1 of 2

In the next segment, you will resume performing tasks on the AWS CLI and learn how to work with Amazon EC2 through it.

**S3 One Zone-IA**

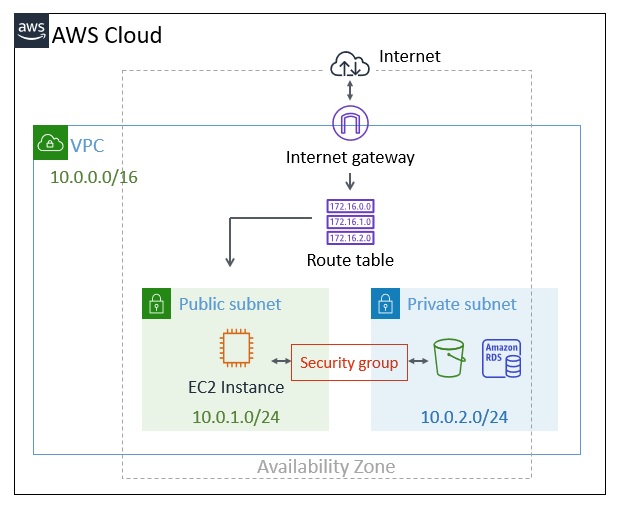
**Feedback :**

*The region code ap-northeast-1 refers to Tokyo. Since this is an earthquake-prone region, you must never rely on only one Availability Zone to store your data.*

# AWS CLI: EC2

In this segment, you will explore the computation service of AWS, that is, Amazon EC2, through the AWS CLI. To make things interesting, you will learn how to create a public network where the instance can interact with components external to a private Cloud network.

You can refer to the diagram below to get a clear understanding of the architecture.



Let us listen to Vinod as he talks about this in the forthcoming video.

Play Video

As mentioned in the image above, the following components must be created in order to complete the expected task:

* VPC
* Subnets
* Internet gateway
* Route table
* Security group
* Key pair
* EC2 instance

The code covered in the video is provided below.

**#** EC2 Commands

**#** Creating a VPC

aws ec2 create-vpc --cidr-block 10.0.0.0/16

**#** VPC Details

aws ec2 describe-vpcs --vpc-ids vpc\_ID\_from\_above\_command

**#** Creating subnet 1

aws ec2 create-subnet --vpc-id vpc\_ID\_from\_above\_command --cidr-block 10.0.1.0/24

**#** Copy the subnet ID for subnet 1 from here

**#** Creating subnet 2

aws ec2 create-subnet --vpc-id vpc\_ID\_from\_above\_command --cidr-block 10.0.2.0/24

**#** Copy the subnet ID for subnet 2 from here

**#** Subnet 1 Details

aws ec2 describe-subnets --subnet-ids subnet\_1\_ID\_copied\_from\_above

You learnt how to create two components: VPC and subnets. In doing so, you came across the networking concept of **CIDR Block**. CIDR stands for Classless Inter-Domain Routing, and it is a method for allocating IP addresses for the VPC and the subnets in AWS. In case you are not aware of the notation, you can refer to the document below to understand it better.

**[CIDR Blocks](https://cdn.upgrad.com/UpGrad/temp/6ca5ce30-b0b1-46ce-a58e-dfb510333358/CIDR+Blocks.pdf" \o "CIDR Blocks.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/UpGrad/temp/6ca5ce30-b0b1-46ce-a58e-dfb510333358/CIDR+Blocks.pdf" \o "CIDR Blocks.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/UpGrad/temp/6ca5ce30-b0b1-46ce-a58e-dfb510333358/CIDR+Blocks.pdf" \o "CIDR Blocks.pdf" \t "_blank)**

Questions:1/1

Mandatory

**CIDR Block**

Which of the following ranges covers all the IP addresses that fall under the CIDR Block **10.0.1.0/26**?

Top of Form



10.0.1.0 to 10.0.1.45



**10.0.1.0 to 10.0.1.63**

**Feedback :**

*Since the last number is 26, you can alter 32 - 26 = 6 bits. Therefore, you will have 2^6 = 64 IP addresses under the block.*

**Correct**



10.0.1.0 to 10.0.1.99



10.0.1.0 to 10.0.1.127

Bottom of Form

**done**Your answer is**Correct.**

Attempt 1 of 2

Let’s now see how to execute the remaining tasks for EC2 instance creation through AWS CLI.

Play Video

The code in the video has been provided below.

**#** Creating internet gateway

aws ec2 create-internet-gateway

**#** Copy the Internet Gateway ID from here

**#** Attaching the internet gateway with the VPC created before

aws ec2 attach-internet-gateway --vpc-id vpc\_ID --internet-gateway-id internet\_gateway\_ID

**#** Create Route table

aws ec2 create-route-table --vpc-id vpc\_ID

**#** Copy the Route table ID from here

**#** Attaching the route table to the internet gateway

aws ec2 create-route --route-table-id route\_table\_ID --destination-cidr-block 0.0.0.0/0 --gateway-id internet\_gateway\_ID

**#** Attaching the route table with the subnet 1

aws ec2 associate-route-table --subnet-id subnet\_1\_ID --route-table-id route\_table\_ID

**#** Code to map a public IP to the instance on launch

aws ec2 modify-subnet-attribute --subnet-id subnet\_1\_ID --map-public-ip-on-launch

So, using the code above, you have attached an Internet gateway to Subnet 1 inside the VPC so that it can be accessed from the external environment. To access it, you have also modified the subnet attribute to attach a public IP that will be assigned to an instance as soon as it is launched under Subnet 1.

After you have completed the above tasks, you can proceed with the next video.

**Important Notes**

You will require the ami-ID and the public IP of your system for the next video.

* The ami-id AMZNLINUXHVM described in the video below at [05:55] is for ap-south-1 as Vinod is working in the Mumbai region. However, your account is restricted for the **North Virginia** region. Therefore, you must update the command before implementation. You can refer to the document 'AWS CLI - EC2' provided after the video to check the correct ami-id.
* The document will also help you determine the public IP address of your system. Please **do not**type the same IP address, because in that case, the instance will not open on your system.

Play Video

**Note:** Students working with the Windows OS must use the [documentation](https://cdn.upgrad.com/uploads/production/e7598f7d-6bea-46e2-a35c-dfaac6c57392/Connecting+to+EC2+instance+-+Windows.pdf) provided in the previous module to access the EC2 instance through PuTTY.

The document for finding the correct ami-ID and the public IP address of your local machine is provided below.

**[AWS CLI - EC2](https://cdn.upgrad.com/uploads/production/7d72e077-55f5-4a4f-b034-e1f8f5487f4c/AWS+CLI+-+EC2.pdf" \o "AWS CLI - EC2.pdf" \t "_blank)**

[file\_download](https://cdn.upgrad.com/uploads/production/7d72e077-55f5-4a4f-b034-e1f8f5487f4c/AWS+CLI+-+EC2.pdf" \o "AWS CLI - EC2.pdf" \t "_blank)**[Download](https://cdn.upgrad.com/uploads/production/7d72e077-55f5-4a4f-b034-e1f8f5487f4c/AWS+CLI+-+EC2.pdf" \o "AWS CLI - EC2.pdf" \t "_blank)**

You finally learnt how to create an instance in the above video. The following code was used to do so:

**#** EC2 Commands

**#** Creating key pair

aws ec2 create-key-pair --key-name keypair1 --output text --query "KeyMaterial"> keypair1.pem

**#** Opening the .pem file (Windows)

**#** Open using the 'Notepad' application

**#** Opening the .pem file (Linux/Mac)

vi keypair1.pem

**#** Changing the permission to read only (Linux/Mac)

chmod 400 keypair1.pem

**#** Creating Security Group

aws ec2 create-security-group --group-name test\_group --description "Security group for demo" --vpc-id vpc\_ID

**#** Copy Security Group ID as it will be helpful later

**#** Adding rule to your security group to permit access from your laptop - Ports 22 and 8888

aws ec2 authorize-security-group-ingress --group-id security\_group\_id\_from\_above --protocol tcp --port 22 --cidr your\_public\_ip

aws ec2 authorize-security-group-ingress --group-id security\_group\_id\_from\_above --protocol tcp --port 8888 --cidr your\_public\_ip

**#** Initiating an ec2 instance

aws ec2 run-instances --image-id correct\_AMI\_ID\_from\_document --count 1 --instance-type t2.micro --key-name keypair1 --security-group-ids security\_group\_id\_from\_above --subnet-id subnet\_1\_ID

**#** Copy ths instance ID from above

**#** Describing the details of the instance

aws ec2 describe-instances --instance-id Instance\_ID\_from\_above

**#** Logging into the instance in a LINUX/Mac environment

ssh -i keypair1.pem ec2-user@EC2\_instance\_public\_IP

**#** Starting an existing instance

aws ec2 start-instances --instance-ids Instance\_ID

To access an AWS service from the created EC2 instance, you must also attach a role to it. The code for this is provided below. It helps you attach the role created in the previous segments to the created instance.

**#** EC2 Instance

**#** Creating an instance profile to attach the role to the instance

aws iam create-instance-profile --instance-profile-name instanceprofile1

**#** Attaching the created role to the instance profile

aws iam add-role-to-instance-profile --role-name role1 --instance-profile-name instanceprofile1

**#** Attaching the instance profile to the instance

aws ec2 associate-iam-instance-profile --instance-id Instance\_ID --iam-instance-profile Name=instanceprofile1

**#** Stopping the instance

aws ec2 stop-instances --instance-ids Instance\_ID

Now you must have a clear understanding of how to create an EC2 instance through the CLI.  Note that you must stop the instance once you have implemented all of the steps if you are taking a break. You can restart the instance using the code provided above whenever required. However, you must also keep a check on the public IP in the security groups. It must always be updated to ‘My IP’ under the inbound rules on the EC2 dashboard at the start of the section.

Questions:1/1

Mandatory

**AWS CLI: EC2**

Which of the following components is responsible for establishing permissions to access resources from a public environment?

Top of Form



VPC



Internet gateway



**Security group**

**Feedback :**

*A security group defines the permissions for AWS services. It does not control the permissions for external resources.*

**Incorrect**



Route table

**Feedback :**

*You can define the rules to interact with external resources through a route table.*

**Correct**

Bottom of Form

**close**Your answer is**Incorrect.**

Attempt 2 of 2

Now, you can easily build an EC2 instance using the AWS CLI commands. In the next segment, you will learn about a few advanced concepts associated with Amazon EC2.

#### EC2: Autoscaling

Suppose you want to introduce a new feature in your application that requires very high processing and increases the load on the existing infrastructure by a huge amount. Which of the following types of scaling must you use in this scenario?

Vertical scaling

Horizontal scaling

**Feedback:**

In case the entire infrastructure is affected by a huge amount of load, it is advised to use horizontal scaling. In vertical scaling, there is a limit to which you can increase the efficiency of resources. Hence, you may not be able to accommodate the entire load.