upGrad

Summary

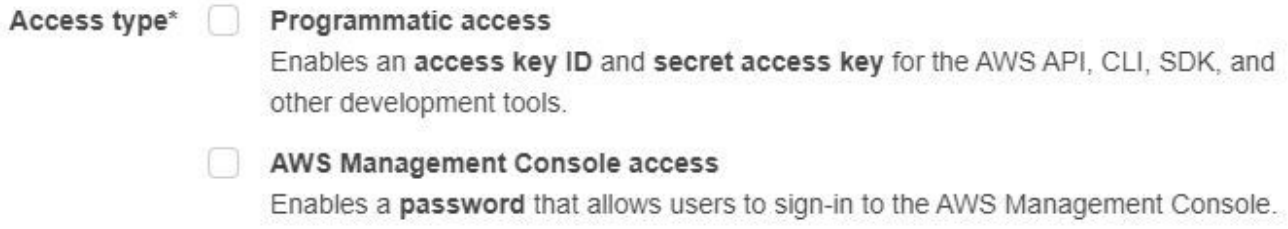
AWS CLI

In this session, you were introduced to the AWS Command Line Interface (CLI). You also gained a better understanding of how deployment happens in a cloud.

AWS CLI: Introduction

There are two ways to access AWS services:

1. **AWS Management Console Access**
2. **Programmatic Access**



Programmatic access means that you can manage the AWS resources by writing a code. This is where the AWS Command Line Interface, or CLI, comes into the picture.

According to the AWS 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***".

Thus, you would be able to perform all the tasks that you saw on the AWS Management Console using the terminal program through the CLI. It comes with its own syntax and commands, which you will learn about in the following lectures.

Advantages of the CLI has over the Management Console:

1. 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 the launch.
2. The AWS CLI gives you control of all the services on **one platform**, which saves 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 webpages for two different services: first, the IAM page to create a new role that allows access to S3, 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.

3. Since AWS CLI is a programming console, it also comes with the ability to **automate tasks through scripts**. You can easily reduce the time spent by writing code for every task that you were expected to perform manually on the Console. This also reduces the scope of errors that can be committed while executing the operations manually. For example, if you are expected to extract reports from your S3 bucket on a daily basis, instead of manually performing the task every day, you can automate the process via the CLI.

AWS CLI: Installation

AWS CLI comes in two different versions:

1. Version 1.x: Suitable for use in a production environment
2. Version 2.x: Work-in-progress and intended for testing and evaluation

Important points to note:

1. Version 1 is recommended. Version 2, being the latest, is still a work-in-progress and, hence, it is not advised for use in production environments.

2. There are two prerequisites for Version 1.x:
    a. Python package (versions above 2.7 and 3.4)
    b. PIP package

3. You can install the CLI package on your local machine using the command '*pip install awscli*'. You can use '*Aws –version*' to verify that your installation is complete.

4. The user will be able to log in through the console only if they have programmatic access. In case you have not provided your account with programmatic access, you can go back to the Management Console and change the settings.

5. Access keys are the credentials that allow you to access the AWS services through the CLI. Each access key consists of two parts: the access key ID and the security access key. You must always keep these secure and never share them with anybody.

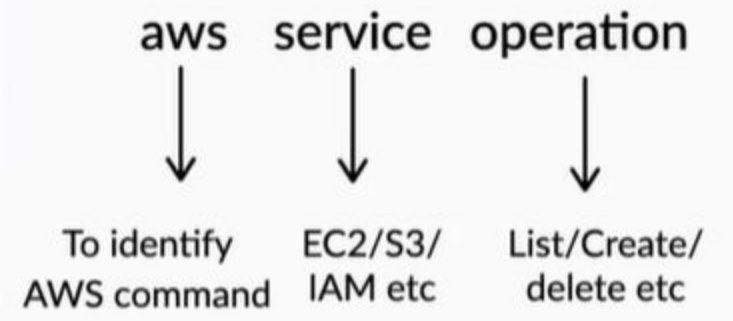
6. Note you can only have a maximum of two active access key pairs. Once a new access key is created, you will see it on the screen. Download the key by clicking on the Download.csv file. Note that from then on, the secret key will not be accessible; hence, it is important to download a local copy. Since the secret key gives complete access to your account, you must store it safely and not share it with anybody.

7. Use '*aws configure*' to log into the CLI. This command helps you configure the AWS CLI as per your requirements. It prompts you to provide your access credentials, the region of operation and the default output mode. It is advisable to specify a region or output format, since you will have to define it every time along with the code. Hence, do not leave these options blank.

8. You can access AWS services directly from the console environment using the access key pair of an IAM user.

AWS CLI: IAM

The structure of the CLI commands are as follows:



1. They start with aws, as it helps the console environment identify that you want to work with an AWS service.
2. Next, you mention the service that you want to work with, for example, s3, ec2, etc.
3. Finally, you can choose the operation, for example, listing the available services, creating a new one or starting an existing one.

This structure is followed by all the services that you want to use in AWS.

aws iam help	The 'Help' command gives the complete command usage along with a lot of information about each parameter.
aws iam create-user --user-name user1	This command is used to create the user (user1). It returns the details of the created user along with their name, userid and ARN.

ARN or Amazon Resource Names is a unique ID to identify Amazon resources across all accounts and resources. It helps you to differentiate between every resource that is available on your cloud. Every customer has a unique customer ID and no two components can have the same name under a service.

The usual format of an ARN is:

Arn : aws : <name of the service> : : <account-id> : <resource-type> / <resourceid>
Arn : aws : iam : : 688716701626 : user / user1

aws iam create-group --group-name group1	1. Creating a group (group1) for the IAM user 2. This command returns the details of the group created along with its name, ID and ARN, along with the date of creation, in JSON format.
aws iam add-user-to-group --user-name user1 --group-name group1	1. This command is used to add users to the group. 2. Using this command, we have added the user user1 to the group group1.
aws iam get-group --group-name group1	1. You can use this command to verify the details of your group. 2. If you execute the previous command, you will find user1 in the list.
aws iam create-policy --policy-name policy1 --policy-document file://test.json	1. Create a policy file(policy1) that provides access to S3.

	2. Once the policy is ready, you can see all the details, including its ARN, name and ID, in JSON format. 3. We use the ARN of this policy to attach it to the role and, hence, store it for further use.
aws iam create-role --role-name role1 --assume-role-policy-document file://test_entity.json	1. We require a role so that the policy can be attached to a service. Thus, we create a role(role1) that allows S3 access to an EC2 instance. 2. We use this command to attach the policy document *test_entity.json* to the role.
aws iam attach-role-policy --role-name role1 --policy-arn arn:aws:iam::688716701626:policy/policy1	1. Use this command for attaching the policy to the role.
aws iam list-attached-role-policies --role-name role1	1. Use this command to check whether your policy has been attached to the role. 2. Your policy will be listed under attached policies.

AWS CLI: S3

1. S3 is AWS's simple storage service, where you can **store files of any size or format**. Many companies use this service to store their data on the cloud.
2. It brings **scalability** in terms of storage to your AWS account. You can upload gigabytes, terabytes or even petabytes of data here and it will still have enough space for more files.
3. Once uploaded to the cloud, the **file stays secure and can be accessed from anywhere by authorised users.** For this, both your IAM and bucket policies need to be defined correctly.

aws s3 mb s3://testbuck-2	1. This command is used to create an S3 bucket(testbuck-2) using the CLI. 2. You can use the mb attribute. 3. Along with the attribute, you need to specify the name of the bucket and the path where you want to store it.
aws s3 ls	This command lists all the buckets under your account.
aws s3 cp ./test.json s3://testbuck-2/test.json	This command is used to upload a file (test.json) to that bucket.
Aws s3 ls s3://testbuck-2	This command is used to verify that the bucket has been created.
Aws s3 rm s3://testbuck-2/test.json	This command is used for deleting an object from the bucket.
Aws s3 rm s3://testbuck-3 --recursive	This command is used for deleting all objects inside the bucket.
Aws s3 rb s3://testbuck-2	This command is used for deleting the entire bucket.

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

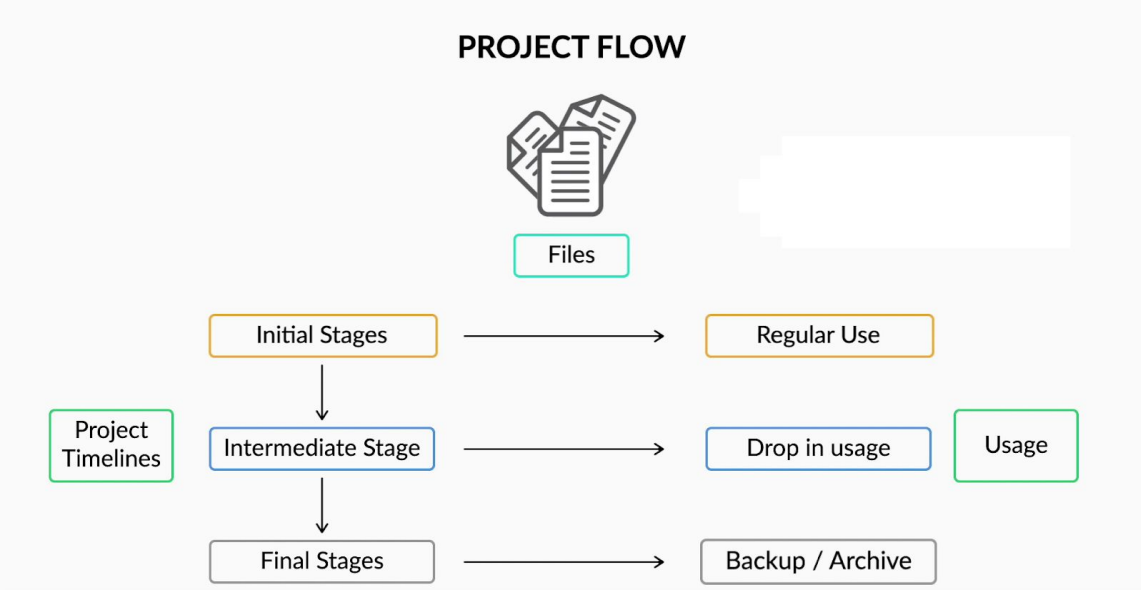
cp	To copy objects; it can be used to upload and download files from S3 to the local machine
ls	To list buckets/objects in S3
mb	To create a new S3 bucket
mv	To move objects present on a local disk or in S3
presign	To provide a pre-signed URL for an Amazon S3 object
rb	To remove an empty bucket
rm	To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

sync	To sync directories present in S3
website	To set the website configuration for a bucket

S3: Storage Classes - I

The usage or 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.

Since Amazon S3 is a paid service, you would want to optimise the storage as much as possible. The usage pattern of a file can also change over time, as mentioned. This cost will be influenced by both the capacity and the duration of usage of the service. Hence, you have to optimise your processes to reduce these costs as much as possible. This is where storage classes are helpful.



AWS S3 Storage Classes

1. **Amazon S3 Standard Storage Class**
    a. It is used for the **most frequently accessed data**. It is the costliest per-unit storage class, as all the data stored under this class is readily accessible.
    b. This storage class offers **low latency and high throughput**, which provides a fast network coupled with fast I/O performance, and guarantees 99.999999999% or eleven 9's durability for objects stored across all availability zones and 99.99% availability throughout the year.

- In the AWS context, object durability reflects the percentage of a 1-year period for which the data stored in S3 will not be lost.
- Object availability reflects the percentage of a 1-year period for which the data stored in S3 might be inaccessible.

c. Due to all of its features, this class is best suited for **running cloud applications**, performing data analytics, hosting dynamic websites, and mobile and gaming applications.

2. **Amazon S3 Standard-Infrequent Access Class**
   a. It is useful for data that is **accessed less frequently** but requires quick retrieval when needed.
   b. Here, the **storage cost is reduced, as the data is stored under infrequent access**, although a per-GB retrieval fee is later added for the quick-access feature.
   c. This type of storage is ideal for important data that you want to **store for long durations**, as you may want to retrieve files quickly whenever they are needed. It can also be helpful for storing disaster recovery files to retrieve as quickly as possible in case of failures.
   d. Like the previous class, it also offers **low latency and high throughput** with eleven 9's durability and 99.9% availability.

3. **Amazon S3 One Zone-Infrequent Access Class**
   a. It is quite similar to the S3 Standard-Infrequent Access class, as it also stores infrequently accessed data, although it provides **quick access for a small retrieval fee**. The difference is that the data is available over a single zone in this class.
   b. In this storage class, the entire data is stored **only in one availability zone**, thus making it **less reliable**. For all the other storage classes, S3 maintains a minimum count of three availability zones, so that if one availability zone fails or becomes unavailable, the data is not lost and S3 can provide the file from another availability zone. Hence, this **storage class is cheaper, but at the cost of reliability**.
   c. It is generally used as a **low-cost option**, as it is 20% cheaper when compared with the S3 Standard Storage class. Also, it finds **utility when you need to maintain secondary backups** of data with quick access.
   d. You would always be advised to keep only **easily recreatable data** under this class, so that you do not face major issues in case of any failure.
   e. This storage class offers **eleven 9's object durabilit**y over one availability zone, although due to lower reliability, object availability is reduced to 99.5%.

4. **Amazon S3 Intelligent-Tiering Class**
   a. This storage class is used for **data with unknown or varying access patterns**. Under this class, files are automatically **moved to the most cost-effective access tier**, without any performance impact or operational overhead.
   b. Based on the usage pattern, this storage class automatically **sorts data** into frequently and infrequently accessed tiers. If a file has not been accessed over the past **30 days**, then it is transferred to the infrequently accessible tier. Hence, this class **optimises the storage** cost incurred by the user.

c. However, AWS charges a **small monitoring and automation fee** for each object under this tier.
d. With **low latency and high throughput**, this class offers the same eleven 9's durability across all availability zones, although availability drops from 99.99% to 99.90% over a given year.

Apart from the above-mentioned storage classes, Amazon also offers two storage classes dedicated to the purpose of archiving. These are secure, durable and low-cost options to store your data solely for the reason of backup over the cloud.

1. **Amazon S3 Glacier**
   a. This storage class **offers large amounts of data storage at competitive prices**, which may be equal to or even cheaper than keeping them on local premises.
   b. Under this class, AWS also provides the option to **choose the retrieval speed** based on the requirement. Based on your choice, the speed may vary from a few minutes to a couple of hours. An extended version of this is the Amazon S3 Glacier Deep Archive.

2. **Amazon S3 Glacier Deep Archive**
   a. This is **S3's cheapest storage class** and preserves data for very long durations, for instance, data that may be accessed only once or twice in a year.
   b. As the files are deep-archived, **complete retrieval can take up to 12 hours.**

Both the services come with eleven 9's durability and 99.99% availability. Here also, S3 maintains replicated copies of data in at least three availability zones, which results in the same level of availability as the Standard S3 class. These storage classes find great utility in industries such as Financial Services, Healthcare and Public Sector, as they are legally bound to retain data for around 7-10 years or more due to regulatory compliance requirements. Hence, S3 offers a very cheap yet secure solution for storing their data.

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	40 KB	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	40 KB	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.

Source: AWS S3 Documentation

Note that the objects stored in **S3 Glacier and S3 Deep Archive cannot be accessed in real time** and have to be restored first before accessing data. The restored object is a temporary copy, which expires after a specified time period. Hence,the object availability for these classes is 99.99% after the files have been restored, as you can see from the table above. To get more clarity on this, you can refer to this link. You can also check the pricing of different storage classes here.

S3: Storage Classes - II

In this segment, you have learnt how to shift data from one storage class to another and how you can automate this process using the object life cycle. Files that are used frequently in the beginning may no longer be required and, hence, it would be better if we started with the standard storage class in the beginning and later shift to infrequent access tiers or archive tiers when the work is complete.

1. **Manually changing the storage class of a bucket with the Management Console: Key demo points**
    a. Click on the bucket 'testbuck-2' and select the object

    b. Go to 'actions' and click on change storage class

    c. Select any other storage class and click on Save

    AWS has come up with the concept of object life cycle. It provides the option to automate the transition of an object from one storage class to another by defining its life cycle.

2. **Automating the changes in the storage class of a bucket using the object life cycle using the Management Console: Key demo points**

    Suppose you are working on a project where a file has the following trend: you will be using the file actively for the first 30 days, and then the usage would drop and become infrequent. It would remain in that phase for the next 60 days and, finally, after 90 days, when the project ends, you will no longer need the file. However, you are expected to keep a backup of the file in case there are any future requirements. In such a case, follow the steps given below:

    a. Find the option to add a life-cycle rule

    b. Provide a name to the life-cycle rule. In case you want to limit the rule to be set only for files within a specific folder, then you can use the prefix or tags by typing them below.

    c. Set the version of the file for which you want to apply the rule. For you, it would be the current version only

d. To define the transition, click on the ‘add transition’ option.

e. To define the life-cycle rule discussed before, you will be using the file actively for the first 30 days. Hence, it should remain in the Standard Storage class. But after 30 days, the usage is expected to be infrequent. Hence, you can transfer the document to Standard-IA after 30 days, as it will help reduce the cost. Finally, the file has to be archived after a period of 90 days. Hence, you can move it to S3 Glacier after keeping it in Standard-IA for 60 days.

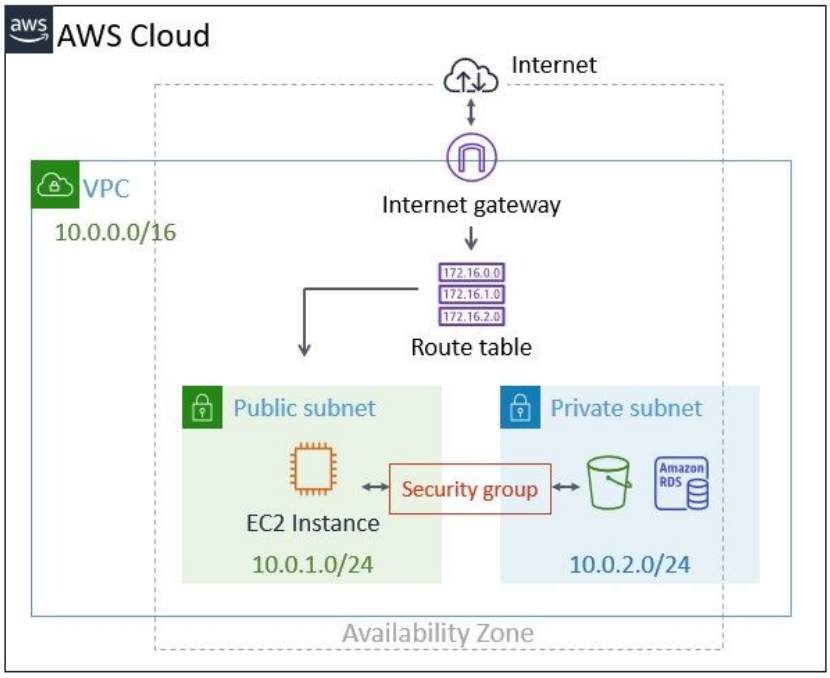
It is always advisable to define the object life cycle at the time of uploading. However, since it is expected that you will work with the free tier as much as possible throughout the course, this will not matter for now. However, in real-life scenarios, you should try to optimise costs as much as possible. Also, you should review the configurations at regular intervals, so that if there are any changes in the workflow, your life cycle would reflect the same.

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

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

AWS CLI: EC2

In this segment, you have explored the computation service of AWS, that is, Amazon EC2, through AWS CLI.



EC2 instances are those components that replace the extra hardware. These instances act as those hardware machines and help in hosting any service that runs on the AWS cloud. With the ability of elastic computation, EC2 helps you to scale up or down based on the requirements within minutes, saving both time and money. Hence, when you want to perform any computation task on the AWS cloud network, EC2 should be the first thing that you should consider.

Working with an EC2 instance using AWS CLI: Key demo points

aws ec2 create-vpc --cidr-block 10.0.0.0/xx	1. We are asking AWS to create a VPC, which will have an internal IP address block: 10.0.0.0/xx. 2. This means that the resources within the VPC can have a private IP address, between 10.0.0.0 and 10.0.255.255, to connect among themselves. 3. We need to note down the VPC id for future reference.
aws ec2 describe-vpcs --vpc-ids vpc-xxxxxxxxxxxxxxxx	Use this command to get the current status of the VPC.

aws ec2 create-subnet --vpc-id vpc-xxxxxxxxxxxxxxxx --cidr-block xx.x.x.x/24	1. Under a VPC, you can create multiple multiple subnets. 2. Subnets can be thought of as sub-networks within the VPC. They allow you to divide your VPC into different sections, which can be defined differently based on the requirement. 3. This command is used to create a subnet.
Aws ec2 describe-subnets --subnet-ids subnet-xxxxxxxxxxxxxx	Use this command to verify that the subnet has been created.
aws ec2 create-internet-gateway	1. You will need an internet gateway to make a subnet public or private. 2. An internet gateway acts as a channel through which you can interact with the outside entities. 3. Use this command to do so.
aws ec2 attach-internet-gateway --vpc-id vpc-xxxxx --internet-gateway-id igw-xxxxx	1. Use this command to open your network for public access. 2. You will have to provide your VPC id and your internet gateway ID.
aws ec2 create-route-table --vpc-id vpc-xxxx	1. Since there will be certain resources like files in RDS, which you would not want the people outside the network to access, you will require a route table 2. Use this command to add a custom route table, which determines which port and IP address can be allowed to access which resource. 3. You can use the route table ID to add rules to the route table.
aws ec2 create-route --route-table-id rtb-xxxx --destination-cidr-block 0.0.0.0/0 --gateway-id xxxxxxx	1. We use this command to create a rule that makes the resources in this VPC accessible from outside.

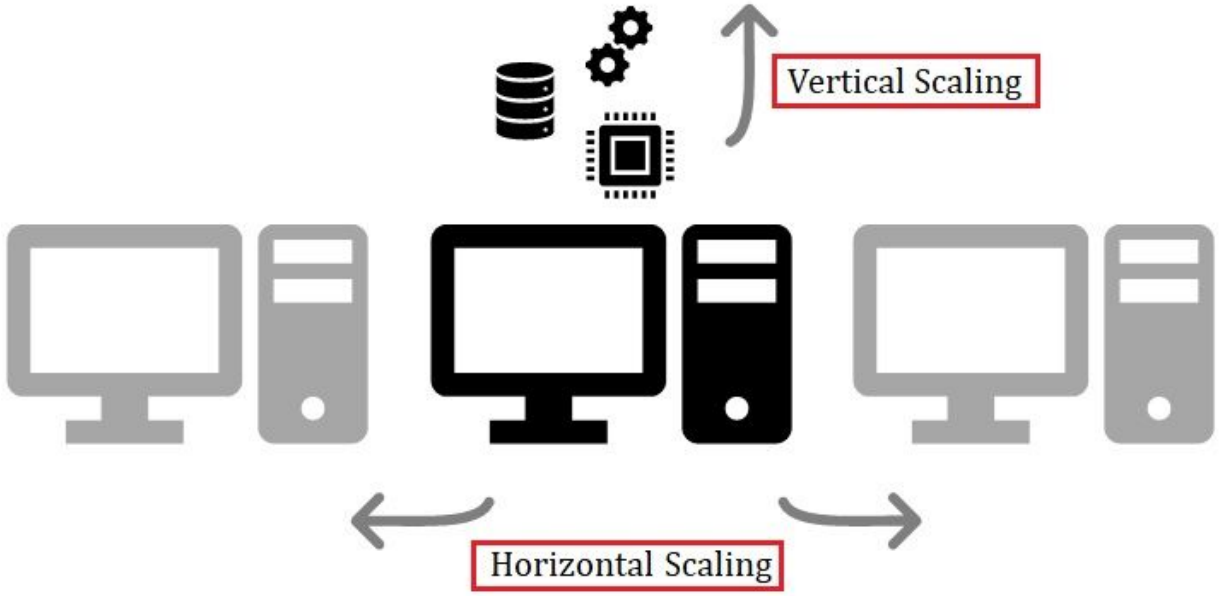
	2. Thus, we will add an entry in the route table to redirect all traffic to the Internet gateway.
aws ec2 associate-route-table --subnet-id subnet-xxxxx --route-table-id rtb-xxxxx	Use this command to define the rule that makes subnet 1 public.
aws ec2 modify-subnet-attribute --subnet-id subnet-xxxxx --map-public-ip-on-launch	Use this command to enable the instance to acquire a public IP address.
aws ec2 create-key-pair --key-name keypair1 --output text --query "KeyMaterial">keypair.pem	This command is used to create a key pair and store the pem file locally.
aws ec2 create-security-group --group-name sshaccess --description "Security group for ssh access" —vpc-id	1. This command is used to create a security group for the instance. The security group will hold the required rules to authorise the connection with other resources. 2. Ensure that you associate it with the VPC ID that has been created.
aws ec2 authorize-security-group-ingress --group-id xxxxxxxxxxxxxxxxxx --protocol tcp --port 22 --cidr xx.xx.xx.xx/xx	1. Use this command to create security group rules to ensure port 22 is allowed to this instance from the local IP address. 2. Note that this IP address must be your public IP address, visible to AWS.
aws ec2 describe-instances --instance-id i-xxxxxxx	You can get the details of your EC2 instance by running this command.
aws ec2 run-instances --image-id ami-02913db388613c3e1 --count 1 --instance-type t2.micro --key-name csd_pair --security-group-ids xxxxxxxxxxxxxxxx —subnet-id subnet-xxxxx	Use this command to create an EC2 instance with the AMI ID (here, it is Amazon Linux) within the subnet and VPC that you have created along with the key pair.
aws iam create-instance-profile --instance-profile-name instanceprofile1	1. Use this command to add your role to the created instance; you will need to create an instance profile.

	2. An instance profile is a container for an IAM role that you can use in order to pass the role information to an EC2 instance when the instance starts.
aws iam add-role-to-instance-profile --role-name role1 --instance-profile-name instanceprofile1	Use this command to add your role to the instance profile mentioned before.
aws ec2 associate-iam-instance-profile --instance-id YourInstanceId --iam-instance-profile Name=instanceprofile1	Use this command to use this newly created instance profile to attach it to the EC2 instance.

EC2: Scaling

Scaling can be achieved in two ways:

1. **Horizontal scaling**:
    a. It increases the number of instances, i.e., increasing the number of virtual machines on which the application is running.
2. **Vertical scaling:**
    a. It increases the power of the existing machines, i.e., ramping up the parameters on the existing instances.
    b. In vertical scaling, you increase the processing memory or storage, or select a more powerful instance type on which to run your application.



It would be very difficult to predict scenarios where the load would increase and when it would decrease. Even if you could predict, the next problem you would face is to calculate the required number and configuration of new instances to be launched. AWS provides a service that could automate the task of scaling as and when required. This feature is termed autoscaling.

Autoscaling:

1. Autoscaling enables you to **run your applications with the correct number of EC2 instances based on the load**. As the load increases, it performs horizontal scaling over the existing architecture. You can specify the conditions and mention the specifications of instances that must be initiated, and as soon as those conditions are met, AWS will start a new instance automatically.

2. In case the instance is no longer required, it will stop it automatically as well. In this way, autoscaling helps to **optimise the costs**, since instances are launched as and when required.

3. You can use autoscaling to maintain the health of your cloud environment. There is always a possibility of instance failure and autoscaling helps you prepare for that. Here, instead of monitoring the load, you can **configure autoscaling to monitor the health of your instances**, and in case there is an issue, it would replace the unhealthy instance with a healthy one. This is a very useful feature, as the failure of even one instance can cause the failure of an entire application.

Specifications to execute autoscaling

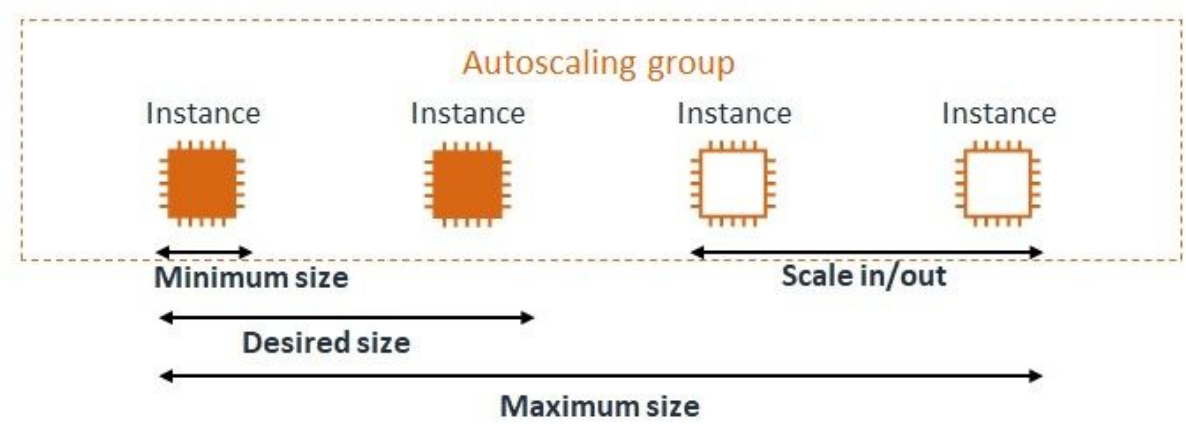
1. **Autoscaling group:**
    a. These are the conditions under which a new instance will be launched.
    b. The autoscaling group contains a collection of EC2 instances based on the execution plan for any application or project.

c. You specify the minimum number of instances that have to be active at all times to support the application at regular load. That is the minimum capacity.
d. Similarly, you are also expected to specify the maximum number of instances in case the load reaches its peak.

2. **Launch configuration:**
   a. This is the configuration of the newly launched instance.
   b. In the launch configuration, you provide the template of the new instances that will be launched under autoscaling.
   c. The configuration could be the same as that of the existing instances, or it could be modeled based on the requirements under which you want a new instance.

The maximum limit is defined to prevent autoscaling from creating new instances after a point, as the cost incurred may exceed your budget. The autoscaling group monitors the desired parameters like health, load, etc., through CloudWatch, and based on your requirements, it launches the desired number of instances at a given point of time to ensure smooth functioning of the environment.

Another way in which you can perform autoscaling is schedule-based. In this case, you will launch a specific number of instances at specific hours or days. This is only possible when you are very sure of the load that you will receive each time.



Performing autoscaling in the AWS environment: Key demo points

1. To perform autoscaling over EC2 instances, you must go to the EC2 service on the Management Console. Both of its components are provided separately under the Autoscaling option.

2. To specify the launch configurations:

   a. Click on the Create Launch Configuration tab.

   b. Here, AWS will first ask you to choose the AMI that you want to work on. Next is the instance type. It also asks you for a role. You are expected to provide a name for the launch configuration.

c. You can also attach an IAM role if required.

d. Next, add the storage.

e. Finally, you can modify the security groups.

f. Once everything is ready, you can review the launch configuration and click on Create.

g. You will be asked to provide a key pair for the instance. You can either create a new key pair or select an existing one from the list.

3. Defining the autoscaling groups:

a. Here, AWS will prompt you to create a new launch configuration if you haven't specified any.

b. Once you click on Next, you will be prompted to specify the group name and select the network parameters, VPC and subnet, under which you want to launch the new instance. These should be selected carefully as the newly launched instances will not serve any purpose if they are inaccessible.

c. Next, you will configure the scaling policies. This is the tab where you specify the parameter based on which a new instance will be launched. You can specify the minimum and maximum number of instances that you want within your group.

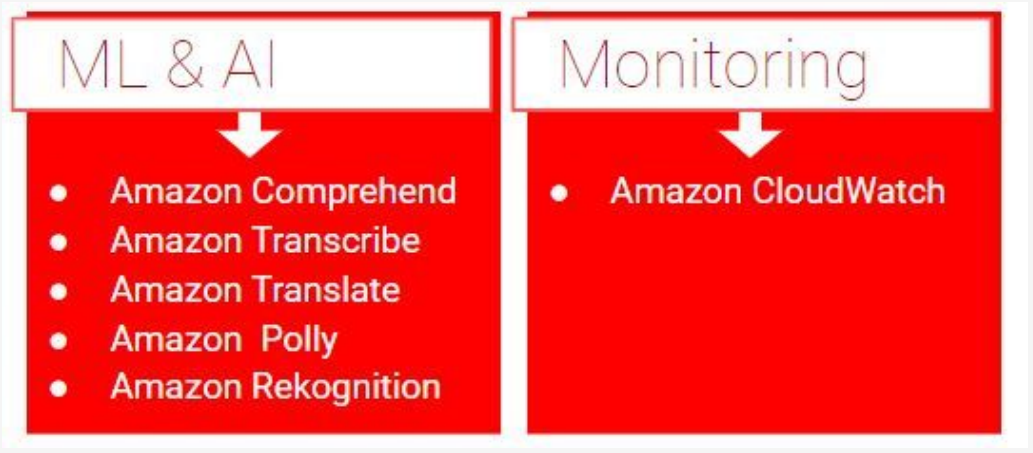
d. As we are working with EC2, to scale between the specified numbers, you are provided with the option of associating with load and computational features. Using the target value, you can specify the threshold load after which a new instance will be launched.

e. AWS also offers the feature of notifying in case any action takes place within the group.

Summary

AWS Services

In this session, you learnt about the following ML and AI and Monitoring services:



Amazon Comprehend

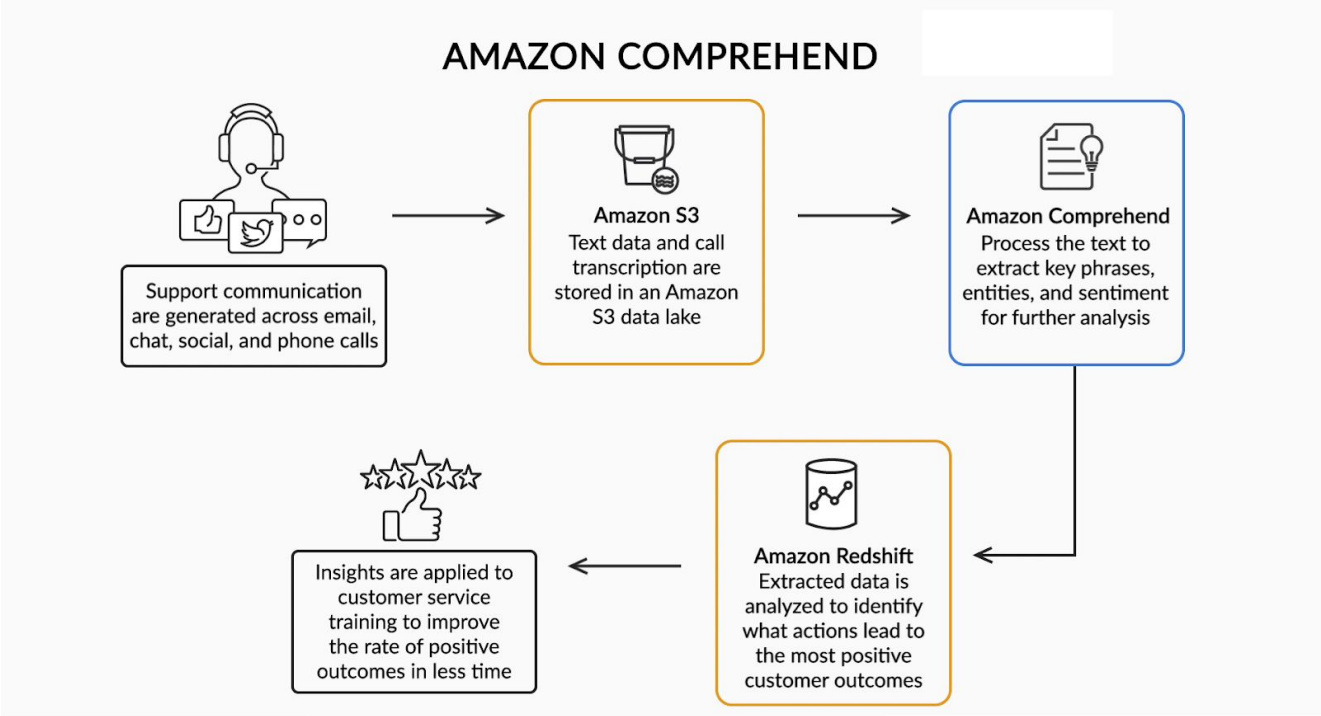
Amazon Comprehend is a **text analytics-based service** from AWS. It uses advanced text analytics algorithms to extract and find insights from the text.

Organisations generate a significantly huge amount of unstructured data, including medical transcripts, product reviews, customer emails and support texts. Deriving meaningful insights from such unstructured data is a difficult problem. However, if the data is generic and not specific to a particular domain, Amazon Comprehend can help you gain insights from it.

It uses the Natural Language Processing (NLP) function in the background to accomplish the task. The service offers the following features:

1. **Keyphrase Extraction**: It provides key phrases from the text.
2. **Sentiment Analysis**: if text data is provided, the service can provide the mood or the sentiment behind the text. This could be mixed, positive, negative or neutral.
3. **Syntax Analysis**: Amazon Comprehend identifies the word boundaries and the part of speech for each word, such as pronoun, noun and so on.
4. **Entity Recognition:** Amazon Comprehend automatically identifies common entities such as location, people, places, etc.
5. **Topic Modeling**: Based on a group of texts provided as input , Amazon Comprehend automatically assigns these text items to some topics.
6. Language Detection: Amazon Comprehend also detects the language of the text and also performs text classification.

Due to the long list of features that it offers, Amazon Comprehend finds great use in multiple industries. You can learn about how companies use this tool for different purposes in the following link. The image below shows one such use:



In the above use case, the data from the customer support centre's emails, Twitter and other social media websites is fed into S3 in the form of collated data. From S3, Amazon Comprehend extracts the sentiment and key phrases that are then stored in the data warehouse. Based on the sentiment and topics, companies can improve their backend support by identifying customer pain points efficiently.

While using Amazon Comprehend, you must keep a check on the cost for each service. You are charged only for the feature that you use. You can learn about the pricing by visiting this page.

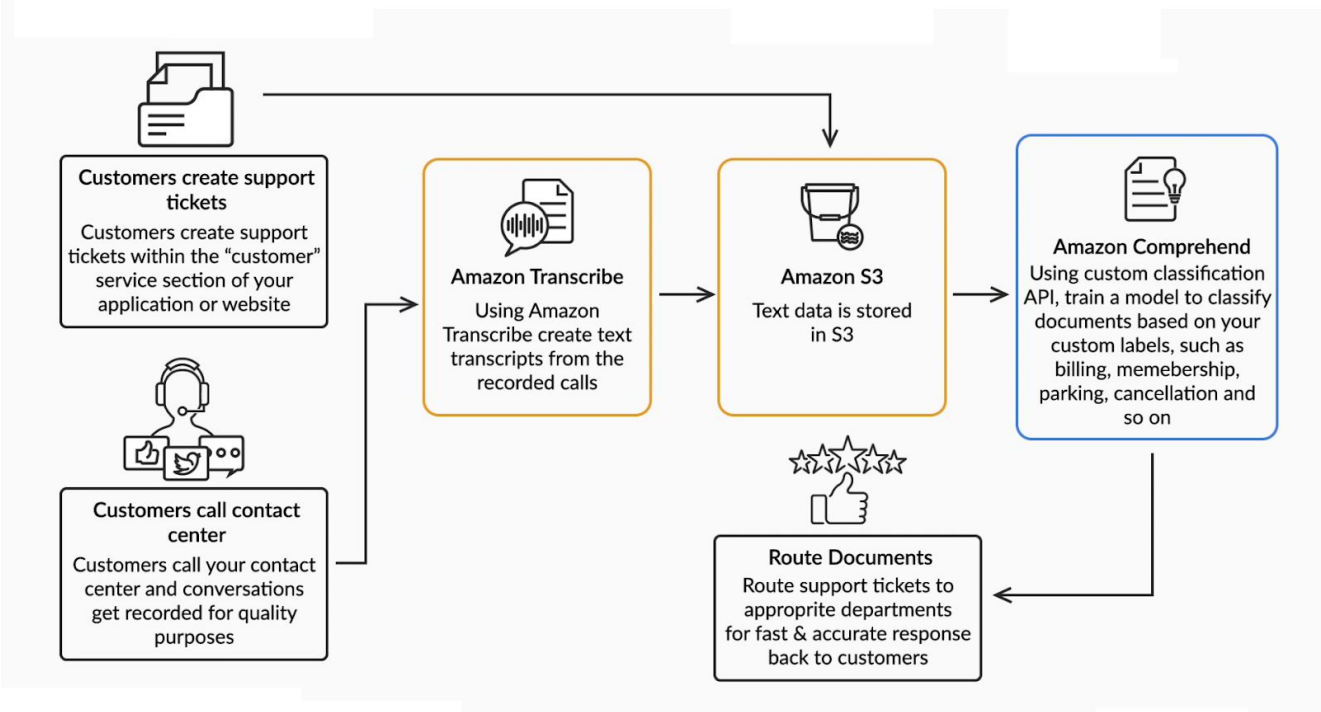
Key Demo Points:

1. You can go to the Amazon Comprehend page, where you can paste a sample customer review in the stub.
2. You can notice that in the entity section Amazon Comprehend will be able to identify the important entities in the conversation, such as locations, organisation names, etc.
3. If you move to the next tab, you can see the key phrases from the text highlighted, which is sufficient to get a gist of the entire text. If you have a huge number of reviews, the key phrases could be helpful in segregating and analysing similar reviews.
4. Under the language tab, it indicates that the text has been written in English.
5. The sentiment tab shows whether the person had a good or a bad experience.

6. This service, when coupled with other analytics services, can give an overall picture of the areas where customers are happy and where they are not.

Amazon Transcribe

Amazon Transcribe, which is a **speech-to-text service,** transcribes audio in real-time. It supports multiple speakers with a custom vocabulary if needed. It also automatically adds relevant punctuation marks while converting audio files into text.



In this case, Amazon Transcribe would enable even customer support calls to be used in improving customer satisfaction as shown in the image above. Before passing the data to Amazon Comprehend, Amazon Transcribe would help convert the calls into text, and then the data from the text can be passed on to Amazon Comprehend, which would generate the sentiment and the keys.

Key Demo Points:

1. Go to the Amazon Transcribe page and click on Real-Time Transcription
2. Click on 'Start Streaming', where you can record your speech and see the transcription output

The service can also be used for **generating subtitles or captions** that can be displayed with videos. The tool further makes this task easier by providing time stamps to map the audio with the text. You can explore different business cases associated with this tool in the following link.

You must be careful while using this service, as you are **charged based on the seconds of audio transcribed per month.** The price varies from one region to another. To check the pricing in different regions, you can visit the following page.

Amazon Polly

Amazon Polly is a **text-to-audio service**, which can help you enable features like dictation in your application.
It uses deep learning technology to mimic human speech. Hence, it has a voice that sounds natural. Amazon Polly also supports real-time conversion in a variety of languages.

Key Demo Points:

1. This service can be found under the Machine learning tab. To start the service, open Amazon Polly
2. It has a dedicated space where you can enter the text that you wish to convert to speech. Amazon Polly also provides the feature of selecting the region and the language that you want the audio output to be in
3. You even have the option to among different voices
4. Another feature that makes Amazon Polly special is that you can add elements to the voice through SSML, which stands for Speech Synthesis Markup Language. A list of tags, for example, pause, whisper, etc. can be used to further enhance the audio. To try this feature, open the SSML tab

The service has multiple use cases across different industries. Some of them are mentioned below:

1. It can be used to generate audio for language-learning apps such as Duolingo, as it provides the feature to generate the output in different languages.
2. It can also be integrated with navigation services. You can use the tool to generate audio directions based on the path to your destination.
3. You can also use this tool to create audiobooks.

You can explore other use cases of this service here.

Now, considering the pricing, you are **charged based on the number of characters of text that you convert to speech.** You can check the charges of using this service here.

Amazon Translate

As the name suggests, Amazon Translate helps **translate content from one language to another.** The service supports approximately **54 different languages**.

The key advantage of using this service is that entities like popular brand names, industry terms, etc. are kept intact in the translated text. You can further define **custom terminologies** to do the same for items specific to your task.

This service can be used for developing websites, documentations, etc. in different languages. It overcomes all the challenges of the traditional practices and provides a quick, accurate and customisable output. By using these features, companies can serve to a wider customer base from different regions and demographics.

Since Amazon Translate is a neural network-based service, the context of the text remains intact during translation. It also provides options to customise terminologies that are specific to a particular domain or industry, which helps in keeping product names, like Amazon Prime, intact during translation.

This is also a paid service and you are **charged for each character provided for translation.** However, Amazon Translate Free Tier provides free translation for 2 million characters per month for a period of 12 months. After that, you are billed monthly at a rate of $15.00 per million characters. You can check the pricing for this service here.

Amazon Rekognition

Amazon Rekognition is another Amazon service that performs visual analytics. It helps detect objects, text, scenes, and activities in a video or image. The tool offers some of the following features:

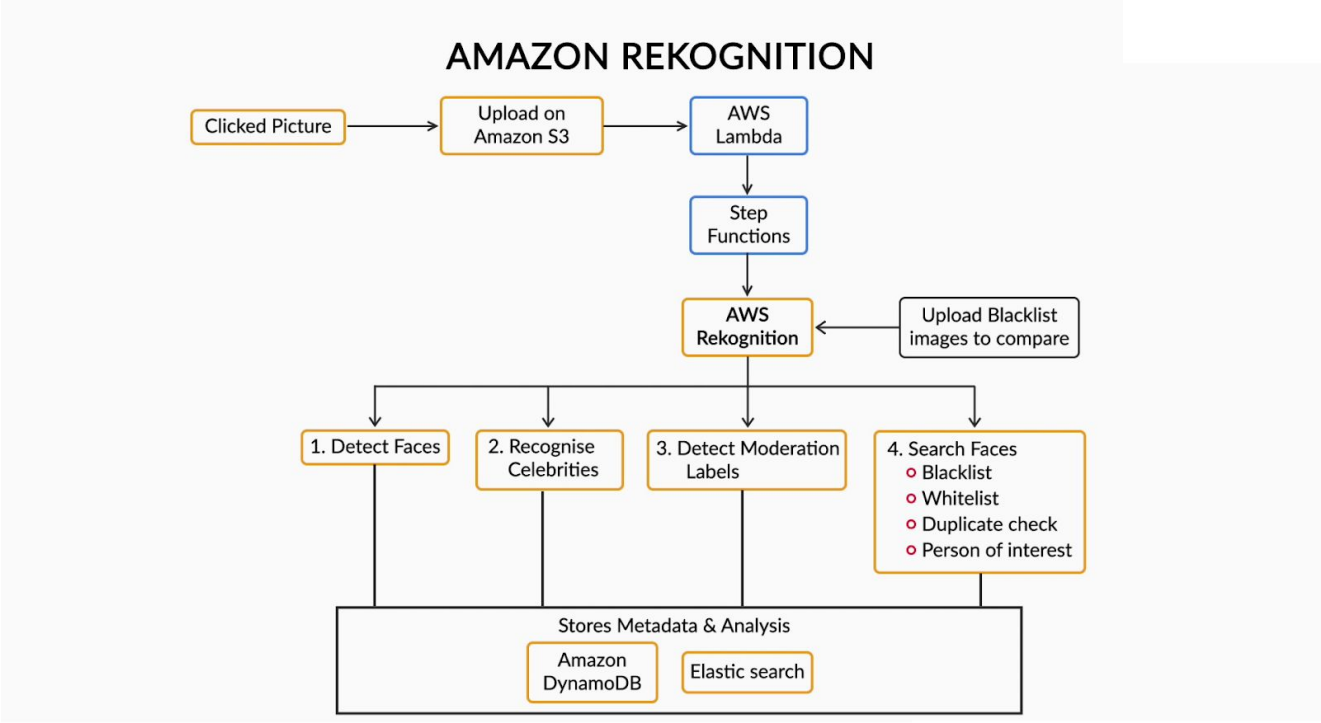
1. **Object, Scenes and Activities Detection**: Rekognition can automatically identify objects such as cars, animals, etc.
2. **Text Detection From Video or Image**: You can extract text from images and videos to further analyse them.
3. **Face Detection and Analysis**: Amazon Rekognition can also be used to detect faces that appear in an image or a video, and perform analysis to collect attributes such as gender, age, etc. This service can be further extended to incorporate user verification. It can also perform celebrity recognition on images and videos.
4. **Path Analysis**: You can also analyse the movements that occur in videos. Sports companies can use this feature to analyse patterns in the movements of players from other teams and devise strategies accordingly.

Other features of this service include **removing objectionable contents from videos and images**.

Amazon Rekognition does a great job in analysing the different elements in an image. It provides a bunch of similar, useful features for analysing videos . You can explore all of these features using the link provided here.

Amazon Rekognition can be a very powerful and useful tool for companies that work with images and videos. You can check how they can use this tool in their applications by clicking on the link provided here.

Amazon Rekognition can be used for a variety of use cases. One such use case is moderating user-generated content. The below architecture shows how Rekognition can be used to build effective solutions for such use cases. It shows a user-generated policing system that captures user data and identifies faces, celebrities, people of interest and explicit content in images.



As seen in the image above, the live images are stored in S3. When the files come in, the system triggers lambda service, which calls AWS Rekognition internally to detect and recognise faces, celebrities and any blacklisted persons. This data can then be stored for further analysis.

From a pricing standpoint, Amazon Rekognition costs between 0.0001$ to 0.0004$ per image based on the number of images processed per month. You can check the pricing information for this service here.

Amazon CloudWatch

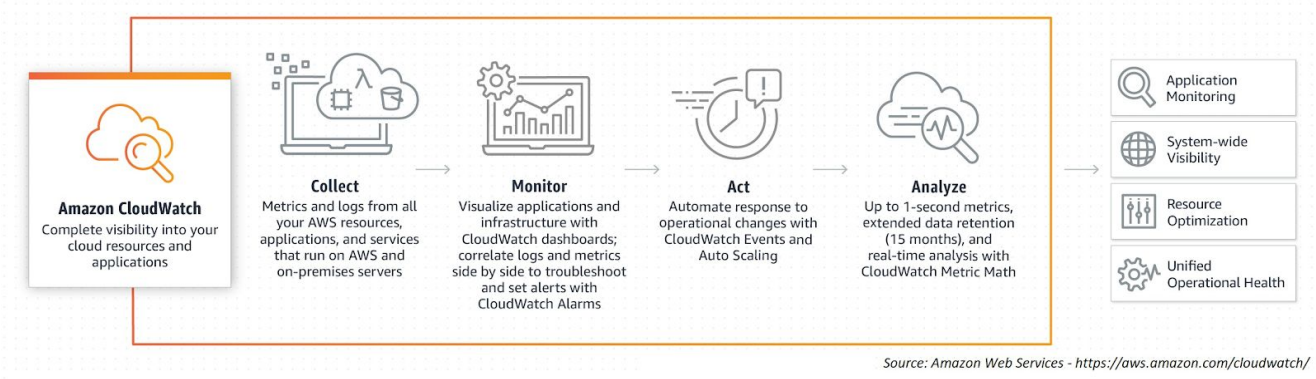
AWS CloudWatch is a **monitoring service** from Amazon Web Services. AWS lets you **collect logs or events based on system metrics, visualise them and set alerts** based on certain parameters such as CPU usage or memory usage from these logs.

Each service has several inbuilt metrics such as object storage size for S3 or CPU utilisation for EC2 instances, which can be directly visualised. The key feature of Amazon CloudWatch is that it **acts as a centralised logging mechanism** for these metrics. All the services that are involved in an application can log its activities and the infrastructure status through this service.

This would result in **better management,** as you will be well-informed about the activites of each resource. It would lead to **optimisation in costs** and even easier debugging in case of infrastructure failure.

CloudWatch also supports custom metrics that you can define based on your application. Its CloudWatch Events feature can track the events in the system in real time. Alarms can be set for these events. For example, you can scale up the EC2 instances in an autoscaling group, based on an event collected by the CloudWatch, such as the CPU utilisation of the EC2 instance. You can similarly invoke a Lambda function based on the CloudWatch event.

You can easily find areas of improvement or possible failures by keeping an eye on the desired metrics of each service. This leads to better management and optimisation in processes and costs.



Source: Amazon Web Services - https://aws.amazon.com/cloudwatch/

Amazon CloudWatch finds useful applications in every project, as it makes the task of handling the Cloud resources easier. You can explore how companies use this tool for different purposes in the following link.

Now, considering the pricing, the amount varies depending on the function that you want to use the service for. Since it is one of the most used services and it helps save a huge amount of additional costs, Amazon has priced the service accordingly. You can learn about the pricing by visiting the following page.

upGrad

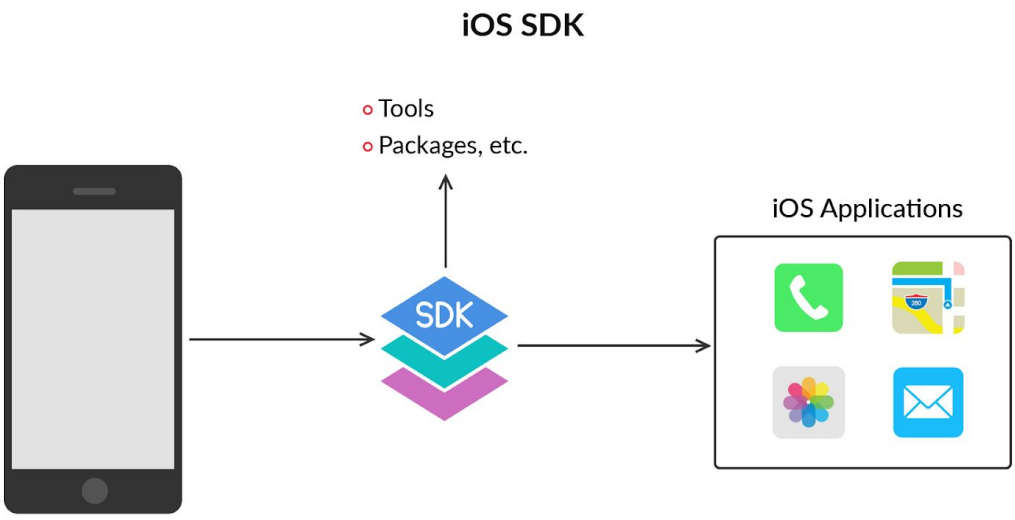
Summary

AWS Services

In this session, you learnt about Software Development Kits (SDKs) and how they can be used to build an application. You also learnt how to access its services using Python. Finally, you built an application that scans the image of a celebrity and returns the name of the celebrity in the image.

Software Development Kits (SDKs)

Software Development Kit (SDKs), as the term suggests, is a toolkit that helps you build or **develop applications for specific hardware or software platforms**. SDK acts as a development package that comes with the necessary tools required to build the desired software in a certain programming language for a specific platform or application. For example, an iOS SDK comes with all the tools that are necessary to build an application, using the Swift programming language, for an iOS device.



As part of the toolkit, an SDK comes with different tools, libraries, documentation, code samples, processes, guides, etc. that enable you to create software applications over a specific platform.
One of the core components of any SDK is language-specific APIs.

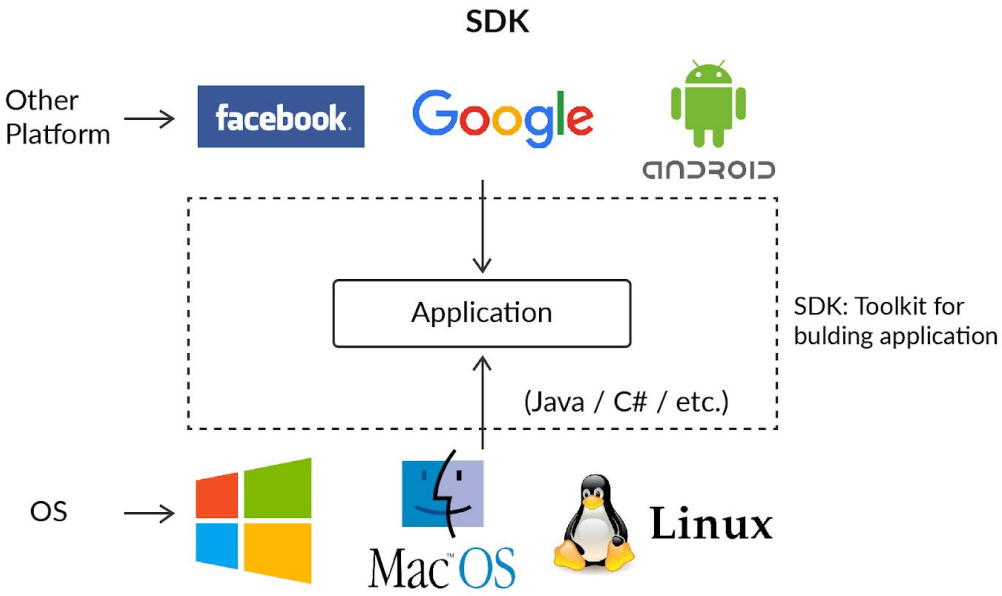
1. **APIs, or application programming interface**, refers to a set of functions that helps **access the features of the service** these APIs are published for. For example, in order to upload or download a file from S3, AWS provides a language-specific API that would help you perform the same function through code files.

2. An SDK has different tools to **help developers perform tasks, including debugging, compiling**, **running** and **testing** their applications.

3. In some cases, the SDK providers also **add examples or small test projects** to their SDKs, so that it is easy for the developers to build applications.

Whenever you are building an application, it requires a platform to run on, be it an operating system such as Windows, Mac or Linux. It needs a programming language in which the code for the application is written. Apart from these factors, it is also possible that you are designing an application for platforms such as Facebook, Google, etc.

Each of these platforms has a unique structure and is completely different from the other. Therefore, if you want to build an application for Windows, it must be compatible with it. Similarly, if you are developing a new application for Facebook, it must be integrable with Facebook's platform. Not all platform/language combinations will have inbuilt support for the above-mentioned platforms and services.



The following are some of the features of SDKs:

1. SDKs provide the **required tools and information on how to work over the platform**. They help you build applications that operate successfully on a given platform or with a particular service. Thus, SDKs are very important when it comes to developing new software and applications.

2. SDKs make life easier as they provide a **suitable environment to develop applications** based on the expertise of the developer.

    a. The AWS platform **offers multiple SDKs based on the use case and the platform** for which you are developing the application.
    b. When you want to use the cloud services as key components in your application, it is necessary that you have an appropriate method through which you can directly call the services in the application by using the language of your choice rather than a command line interface. AWS SDKs helps you with this feature by providing the option to **choose from multiple languages, such as Java, Python, C++, Ruby, etc**. If your application is developed in certain languages, your interaction with the AWS services is possible directly through library calls in these languages only.

upGrad

AWS SDK: Boto 3

Boto3 is the AWS SDK for Python. It will allow you to access all the AWS services through the Python environment. The following are some of the key advantages of using boto3:

1. Boto3 is actively maintained; hence, it remains updated with the changes applied to the services or parameters.
2. The functions of Boto3 follow a consistent pattern, i.e., first creating a client object and then calling the function. Such consistency helps in understanding and implementing functionalities easier and faster.
3. Boto3 supports both Python 2 and Python 3.
4. Finally, Boto3 provides inbuilt mechanisms through which your code can wait for the changes done to the resources to complete. For example, within an application, you can create a new bucket and wait until the bucket is created, and then continue to upload files into that bucket.

Key Demo Points:

1. Log in to one of the previously created instances and run the following command in the shell environment to install the boto3 SDK
   pip3 install boto3
2. In case you are facing any difficulty, you may refer to the Boto3 and Jupyter Installation Guide PDF by clicking on this link.

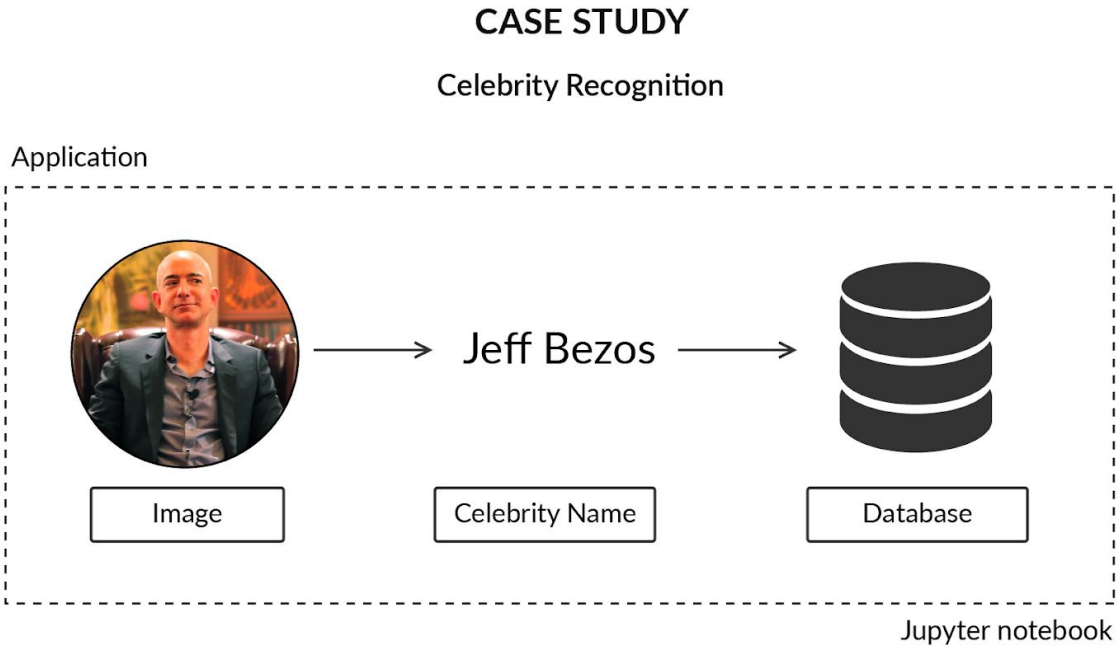
With Boto3, you can develop applications through Python that use different AWS services instead of manually performing tasks by visiting each service. You can simply run the 'import' command to load the package in Python and then use the documentation to work with different services.

The commands in Boto3 follow a simple structure mentioned below:

1. You first define a client associated with the service that you want to access.
2. Once the service is ready, you can call different functions associated with it. Here, the list_buckets() command was used to list all the buckets. You can find all the commands associated with the service in the Boto3 documentation.
3. The output received from the command above contains the metadata associated with the operation, along with the desired results. You can easily fetch the desired elements by iterating over the appropriate section.

Case Study: Introduction

As part of this session, you learnt how to build an application that scans an image for celebrities.



It is always better to have a complete understanding of the requirements before proceeding with the creation phase. In this respect, you should focus on the following two main aspects:

1. **Design Requirements:**
   The design requirements are useful for checking if the proposed solution is able to complete all the tasks successfully.
2. **Budget:**
   Since you have to pay for all the AWS resources used in the deployment, this must be accounted for in advance in order to prevent any complications in the future. Sometimes, you will be expected to either compromise on the requirements or come up with a different approach to execute the project in case the proposed solution does not fit the budget constraints.

Both of these aspects are important to keep the application running. As the application will not be deployed in the production phase, we will not discuss the budget yet. However, you must try to optimise the costs associated with the application by selecting appropriate service parameters and terminating the instances once the task is over.

Case Study: Architecture

In this segment, you worked with the following services to build an application that recognises celebrities in an image:

1. Amazon S3: To store images
2. Amazon EC2: To host the application/Jupyter Notebook
3. Amazon Rekognition: To detect celebrities
4. Amazon RDS: To store the results in a database
5. AWS Lambda: To call AWS services in the notebook

Application Architecture

1. An EC2 instance is l**aunched over which the Jupyter Notebook** is hosted. This will contain all the functions that are required to call and execute the tasks of the different services that are being used.

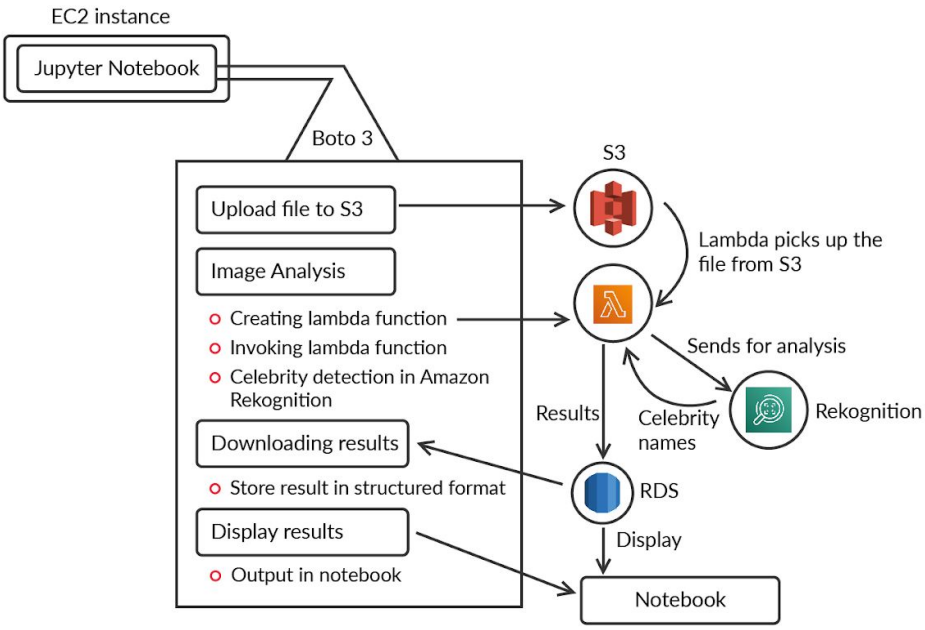
2. Once the computation environment is ready, you will require an image to perform the analysis on. Therefore, the next step is to **upload an image in the S3 source bucket**. You can upload any file from the local machine to the S3 bucket by using the **Boto3 library**.

3. After you have uploaded the file, the next task is to analyse the image. Here, you need to **invoke the Lambda function** that will pick the file from S3 and **send it to Amazon Rekognition**. Amazon Rekognition will analyse the image and search for any celebrity within the image. Later, it will **send the results back to S3 via Lambda in the form of a json file**.

4. Once you have the processed results, you need to put them in a **structured format in RDS**. To do this, you will need to read the **json response** and extract the required details from it. In this case, it will be the file name, which will help you to identify the file, and the name of the celebrities present in it.

5. Once you have all the required details in RDS, you can retrieve the list of all the images and details from the database and display it within the Jupyter Notebook.

APPLICATION ARCHITECTURE



Case Study: Preparatory Steps - I

The case study is divided into the following two parts:

1. **Preparatory Phase**: This phase involves using the CLI to create the required components and assign appropriate roles and permissions to the following services:
    a. EC2
    b. S3
    c. RDS
    d. Lambda (role creation)
2. **Implementation Phase**: This phase involves executing all the tasks associated with the application inside a Jupyter Notebook. All the commands associated with the Boto3 SDK will be executed, and a Lambda function will be created to call services inside a notebook.

Preprocessing steps for EC2: **Key Demo Points**

1. Log in to the CLI environment using the access key pair of your main account
2. Before creating the EC2 instance, you must create three things in advance, Role, Security group and Key pair. The role will help in attaching the policies for accessing different services. The security group will let you restrict the access to the instance. And the key-pair will serve as the credentials to log in to the instance.

3. Once you have created your EC2 instance, you will need the instance ID from the results to fetch the public DNS to log in to the instance.

4. Once you have the public DNS, log in to the instance through the created key pair.

Case Study: Preparatory Steps - II

Once the EC2 instance is ready, you can move to S3. You can either use an existing bucket or create a new one using the 'mb' attribute along with the S3 service in AWS CLI for uploading images.

Creating a new database instance using the CLI

1. RDS can be hosted on six different engines, which include Amazon Aurora, MySQL, MariaDB, PostgreSQL, Oracle and Microsoft SQL Server. For this case study, we had used the MySQL engine.

2. To specify the name of the database, you can use the command --db-name. The role of the function that is specified by --db-name differs based on the engine that you use.

3. Next, you will have to specify the db-instance-identifier. The db-instance-identifier is a helpful attribute in identifying a DB instance when a person is interacting with multiple DB instances in the region.

4. You must specify the DB instance class. This helps you define the processing capabilities of your database.

5. Next, you must specify the username and the password to access the database using the following attributes.
   --master-username and --master-user-password

6. You must make sure that you store the database name, master username and password somewhere for future use.

7. Once your DB instance is ready, you can run the 'describe instances' command to see all the information for every DB instance in your account in the selected region:
   aws rds --describe-db-instances
8. Add a new rule to the security group, so that you can access the database from your EC2 instance.

Creating the required role for the Lambda function that will be used to interact with S3 and Rekognition

1. Create a policy document, through which you can create the required role
2. Attach the policy to the created role
3. Once the role is ready, you can fetch the full ARN of the created role and store it safely, as it will be required to apply the role to the Lambda function that you will be creating

Writing the codes to call and execute the tasks that are needed to search the images for any celebrities

1. Upload an image from the local machine on S3. To upload a file on S3, you have to provide three things: first, the bucket name where you want to store the file; second, the path and the file name where the file is stored on on the local machine; and lastly, the name that you want for the uploaded file on S3.
2. Use the 'client' function to initiate the S3 service
3. Upload the file using the 'upload_file'command

Defining the lambda function

1. Since the starting point of any Lambda function is its handler, we must first define the function. The function takes standard inputs of event and context. This is passed by the AWS service whenever the function is called.
2. Within the Lambda handler, you must create a client for the Rekognition service through boto3. This will initiate the Rekognition when the function is invoked.
3. Next, you must use the 'recognize_celebrities' function call from the Rekognition service. This service requires an image to be passed to the API as a parameter. You can directly refer to a file in S3 to this call function. This can be done by calling the S3Object and passing in the bucket name and the file name.
4. AWS will run the Rekognition service on the provided image internally, and the results will be stored in the response variable.

Creation of Lambda and uploading the file

1. Once you have the code ready, the next step is to convert it into a Lambda function.
2. Next, you need to zip the created file and keep it in a local folder.
3. Finally, you will have the function 'celebrity_detector', which is ready to use. The next step is to invoke this function in your Python notebook.

Invoking AWS Lambda

1. Here, you will call the created lambda function, so that it can pick up the file that you have uploaded on S3, run rekognition APIs over the same and give back the results.

2. To do this, you must first create a new lambda client and then call the invoke function, where you pass the function name.

3. Next, Lambda logs its events in CloudWatch. As you want it to tail the logs every time you call the function, you need to set the log type as Tail and, finally send the payload.

4. You can read the bucket and the object name from the lambda through the events object.

5. Once you run the command and get the response, you can read the variable 'payload' and get the celebrity name.

Extracting the data and storing it in RDS

1. An RDS database has already been created at the beginning of this case study using the MySQL engine. You will use it to store the obtained output.

2. To perform any task associated with MySQL on Python, you must instal the mysql-connector library on the instance. Without this, you will not be able to access any feature of MySQL in the Jupyter Notebook.

3. In order to set up a connection with the SQL server, you need to create a connector object. You can do this by calling the connect api. Here, you need to pass in the rds hostname that you received from the aws cli command, the username and the password along with the default database name. This database will contain all the tables that you create now.

4. Once you have the connector object, you will need a cursor to execute the SQL commands on the database. To do this, you need to call the cursor function, which is a part of the connector object.

5. To create the table 'celebrities', you can call the ‘create table’ command. The table will have two columns - 'image' and 'celebrity', which will store image names in one and the celebrity names in the other. Both the columns will hold strings.

6. Using the cursor object, you can insert the output from the lambda function, which is the file name and the celebrity name, into the created table. You can do this by using the ‘Insert Into’ command.

7. Once you have stored the values in the table 'celebrities', you will need to read the data from the table. You can do this by using the 'Select' command in SQL. To do this, you need to call the ‘execute’ command followed by a ‘fetchall’ command, which returns a value.