AWS (Amazon Web Services):

Que: What is cloud and how it works.

Ans: Cloud is a technology or technique which gives you ability to share your resources through the internet. Internet connection is must.

OR

Cloud is a service which gives you resources, which you can use from a remote location and you can pay according to your usages.

OR

Cloud computing is the on-demand delivery of, compute power, database, storage, applications, and other IT resources, through a cloud service platform via the internet with pay as you go model.

Cloud service platform:

1. AWS – Amazon web services.
2. GCP – Google cloud platform.
3. Azure – Microsoft Azure.
4. Alibaba.
5. Oracle cloud.
6. IBM cloud.
7. VMware.

Characteristics of cloud:

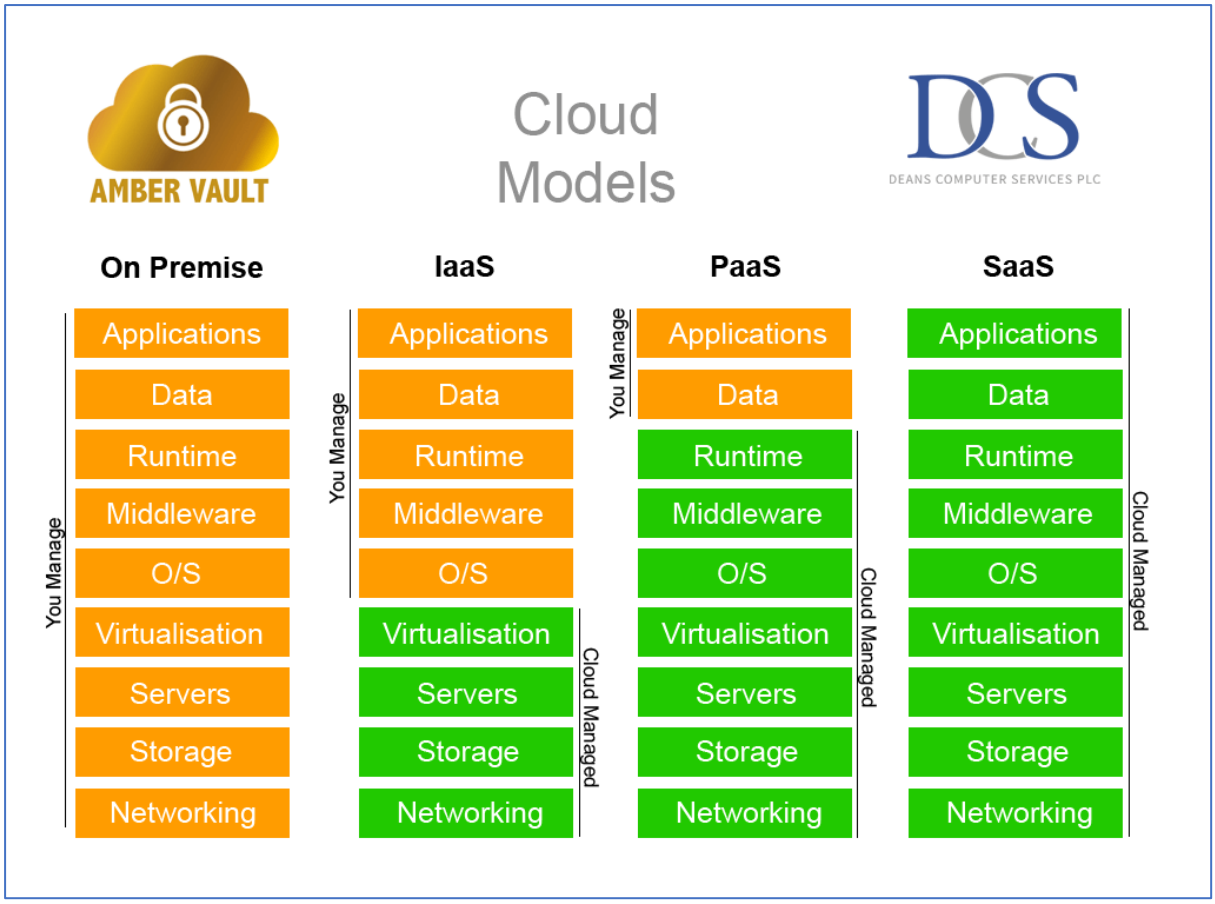
1. On demand self-service.
2. Broad network access.
3. Scalability.
4. Resource pooling.
5. Measured services.

Service model/services in cloud:

1. IAAS – Infrastructure as a service.
2. PAAS – Platform as a service.
3. SAAS – Software as a service.

AWS History:

2006 – AWS Launched.
2010 – Moved to Amazon.com
2013 – Certification launched.



The concepts of IaaS, PaaS, and SaaS:

1. Infrastructure as a Service (IaaS): IaaS is a cloud computing service model that provides virtualized computing resources over the internet. In an IaaS model, users can rent IT infrastructure components on a pay-as-you-go basis, avoiding the need to invest in and maintain physical hardware.

Key characteristics of IaaS include: -

Virtualization: IaaS providers create virtual instances of computing resources such as virtual machines, storage, and networking.

Scalability: Users can easily scale up or down based on their resource needs, often in real-time.

Self-Service: Users have control over the provisioning, management, and configuration of their virtualized resources.

Resource Management: IaaS allows users to control and manage various aspects of their infrastructure, such as configuring firewalls, load balancers, and storage solutions.

Examples: Amazon Web Services (AWS) Elastic Compute Cloud (EC2), Microsoft Azure Virtual Machines, Google Compute Engine.

2. Platform as a Service (PaaS): PaaS is a cloud computing service model that provides a platform and environment for developers to build, deploy, and manage applications without needing to worry about the underlying infrastructure. PaaS abstracts away much of the complexities of infrastructure management, allowing developers to focus on coding and application development.

Key characteristics of PaaS include:

Development Frameworks: PaaS provides development frameworks, runtime environments, and tools to streamline the development process.

Automatic Scaling: PaaS platforms often offer automated scaling options based on application demand.

Database Management: PaaS services may include managed database solutions, reducing the need for manual database administration.

Deployment Automation: PaaS platforms provide tools for continuous integration, continuous deployment (CI/CD), and DevOps practices.

Examples: Heroku, Google App Engine, Microsoft Azure App Service.

3. Software as a Service (SaaS): SaaS is a cloud computing service model where software applications are delivered over the internet on a subscription basis. In this model, users can access and use software applications without needing to install or maintain them locally. SaaS applications are hosted and maintained by the service provider.

Key characteristics of SaaS include:

Accessibility: SaaS applications can be accessed from anywhere with an internet connection and a compatible device.

Automatic Updates: Providers handle updates and patches, ensuring users are always using the latest version of the software.

Multi-Tenancy: A single instance of the software can serve multiple customers, sharing resources securely.

Subscription Model: Users typically pay a recurring fee, which often includes support and maintenance.

Examples: Salesforce, Google Workspace (formerly G Suite), Microsoft 365, Dropbox.

In summary, IaaS provides virtualized infrastructure resources, PaaS offers a platform for application development and deployment, and SaaS delivers fully functional software applications over the internet. These cloud service models allow businesses and developers to focus on their core activities while leveraging the benefits of cloud computing, such as scalability, cost-efficiency, and flexibility.

Virtualization:

We need a hypervisor to virtualise the infrastructure.

Hypervisor – A hypervisor is software that creates and runs virtual machines.

OR

A hypervisor is a software that you can use to run multiple virtual machines on a single physical machine.

Every virtual machine has its own operating system and applications. The hypervisor allocates the underlying physical computing resources such as CPU and memory to individual virtual machines as required.

The hypervisor treats resources—like CPU, memory, and storage—as a pool that can be easily reallocated between existing guests or to new virtual machines.

Hypervisor:

Microsoft: Hyper-V

AWS: AWS Nitro system (Xen).

VMware: Vsphere (ESXi).

AWS - Elastic compute cloud (EC2)

1. Amazon EC2 provides scalable computing capacity in the AWS Cloud.
2. You can use Amazon EC2 to launch as many or as few virtual servers as you need.
3. Amazon EC2 enables you to scale up or scale down the instance.
4. Amazon EC2 is having 2 storage options – EBS (Elastic block store) and instance store.
5. Pre-configured templates are available, known as amazon machine image (AMI).
6. By default, when you create an EC2 account with amazon, your account is limited to a maximum of 20 instances per EC2 region with two default high I/O instances.

Types of EC2 Instances:

- General purpose
- Compute optimized
- Memory optimized
- Accelerated computing
- Storage optimized
- HPC (High performance computing) optimized

Compute – Amazon EC2 Instance Types – AWS.html

EC2 Purchasing options

Amazon EC2 offers several purchasing options to cater to different usage scenarios and budget considerations. These purchasing options allow you to optimize costs and performance based on your specific needs. As of my last update in September 2021, here are the main EC2 instance purchasing options:

1. On-Demand Instances:

On-Demand instances allow you to pay for compute capacity by the hour or by the second without any long-term commitments. This option is ideal for short-term workloads, unpredictable workloads, or when you need instances quickly without upfront costs.

2. Reserved Instances (RIs):

Reserved Instances provide a discount (compared to On-Demand prices) in exchange for a commitment to use a specific instance type in a particular Availability Zone for a 1-year or 3-year term. There are three types of Reserved Instances:

- **Standard RIs:** Provide a significant upfront cost saving compared to On-Demand instances.
- **Convertible RIs:** Offer flexibility to change the instance type later if your needs evolve, while still benefiting from cost savings.
- **Scheduled RIs:** Allow you to reserve instances for specific time periods, which is useful for predictable workloads.

3. Spot Instances:

Spot Instances allow you to bid for unused EC2 capacity in the Spot market. These instances can provide significant cost savings, sometimes up to 90% off On-Demand prices. However, they can be terminated if the capacity is needed by someone willing to pay more than your bid.

4. Dedicated Hosts:

Dedicated Hosts are physical servers with EC2 instance capacity dedicated to your use. This option is useful for compliance requirements, software licensing, or when you need control over the underlying physical infrastructure.

5. Dedicated Instances:

Dedicated Instances are instances that run on hardware dedicated to a single customer. While not necessarily a separate purchasing option, they ensure isolation from instances of other customers on the same physical hardware.

6. Capacity Reservations:

Capacity Reservations allow you to reserve capacity for On-Demand instances in a specific Availability Zone. This option helps ensure you have the capacity you need even during peak times.

How to attach extra volume in your existing server?
-----Launch an EC2 instance (free tier one)

1. Go into your AWS account – Ec2 – Elastic block store – Volumes.
2. Create a Volume – Select the volume type – Give the volume size as per your requirement – Availability zone of the Volume and your instance must be the same.
3. Give the name tag to the volume, and create it.
4. Once volume creation completed, Select the volume, and go into the Action and attach it to your instance (EC2), where you wanted to attach this additional volume.
5. Now login your instance. Run the **lsblk** command to list the available block devices and their mount points. It will be showing in xvdf.
6. Now run the command **df -h** to check if the added volume is showing here or not. As of now you won't see it here.
7. Create a filesystem on the volume – Run **sudo mkfs -t ext4 /dev/xvdf (or mkfs.ext4 /dev/xvdf)**
8. Create a mount point – **sudo mkdir /mnt/myvolume**
9. Mount the volume – **sudo mount /dev/xvdf /mnt/myvolume**
10. Run **df -h** to check.
11. Verify and Test: **cd /mnt/myvolume** & **ls -l**
12. You can create files, read, and write data to this directory to ensure that the mounted volume is working as expected.
13. If you want to increase the volume size, then increase it from AWS console.
14. After increasing from AWS console – run command **resize2fs /dev/xvdf**

How to retrive meta data of your server/instance?

Run command: curl http://169.254.169.254/latest/meta-data

Virtual Private Cloud (VPC)

VPC is a Virtual Private Network. It allows you to create isolated and logically segmented virtual networks within a cloud environment.

- It is logically isolated from other virtual networks in the AWS Cloud.
- Max 5 VPC can be created in one region and 200 subnets in one VPC. =1000
- VPC is Region Specific.
- Subnet is Availability zone specific.
- We can allocate max 5 Elastic IPs.
- A VPC is confined to an AWS Region and does not extend between Regions.
- Once the VPC is created, you cannot change it's CIDR block range.
- If you need a different CIDR size then create a new VPC.

Components of VPC:

- CIDR and IP Address subnets.
- Implied router & Routing table.
- Internet gateway.
- Security groups.
- NACL (Networks access control list).
- Virtual private gateway.
- Peering connections.
- Elastic IP.

Private IP Range of CIDR:

10.0.0.0

172.0.0.0

192.0.0.0

Steps to creating a VPC:

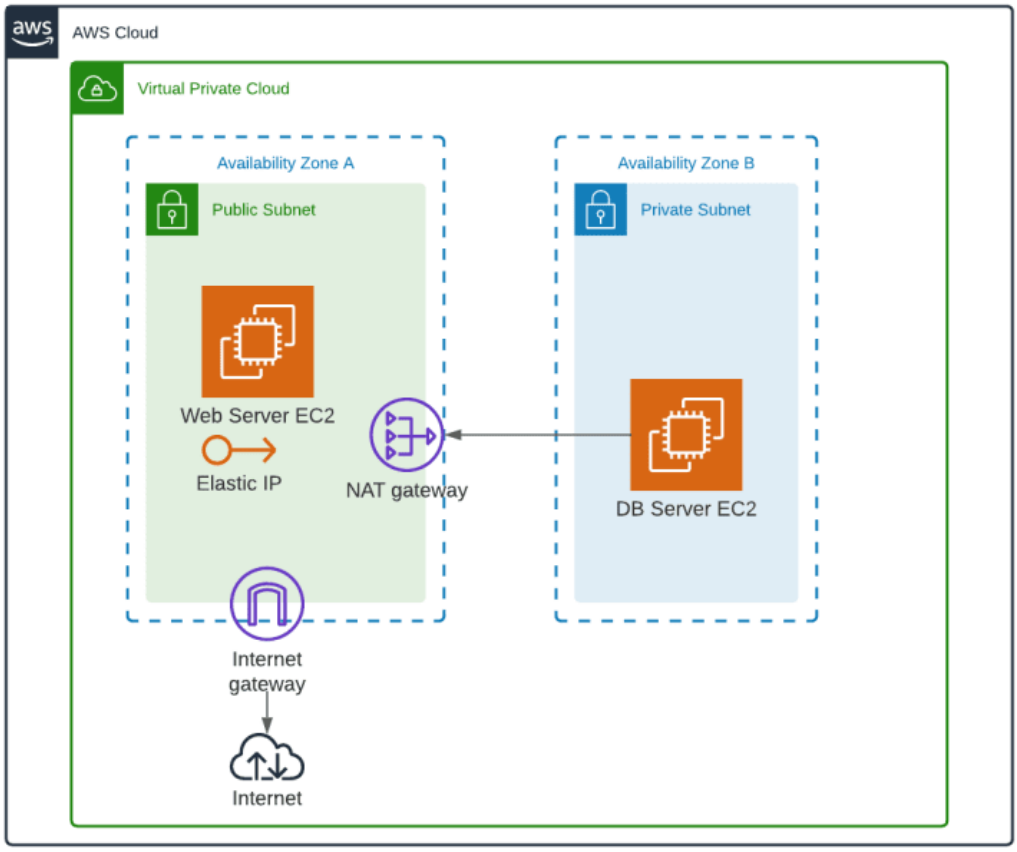
1. Create a VPC
2. Create the Subnet
3. Create Internet gateway
4. Create Route table

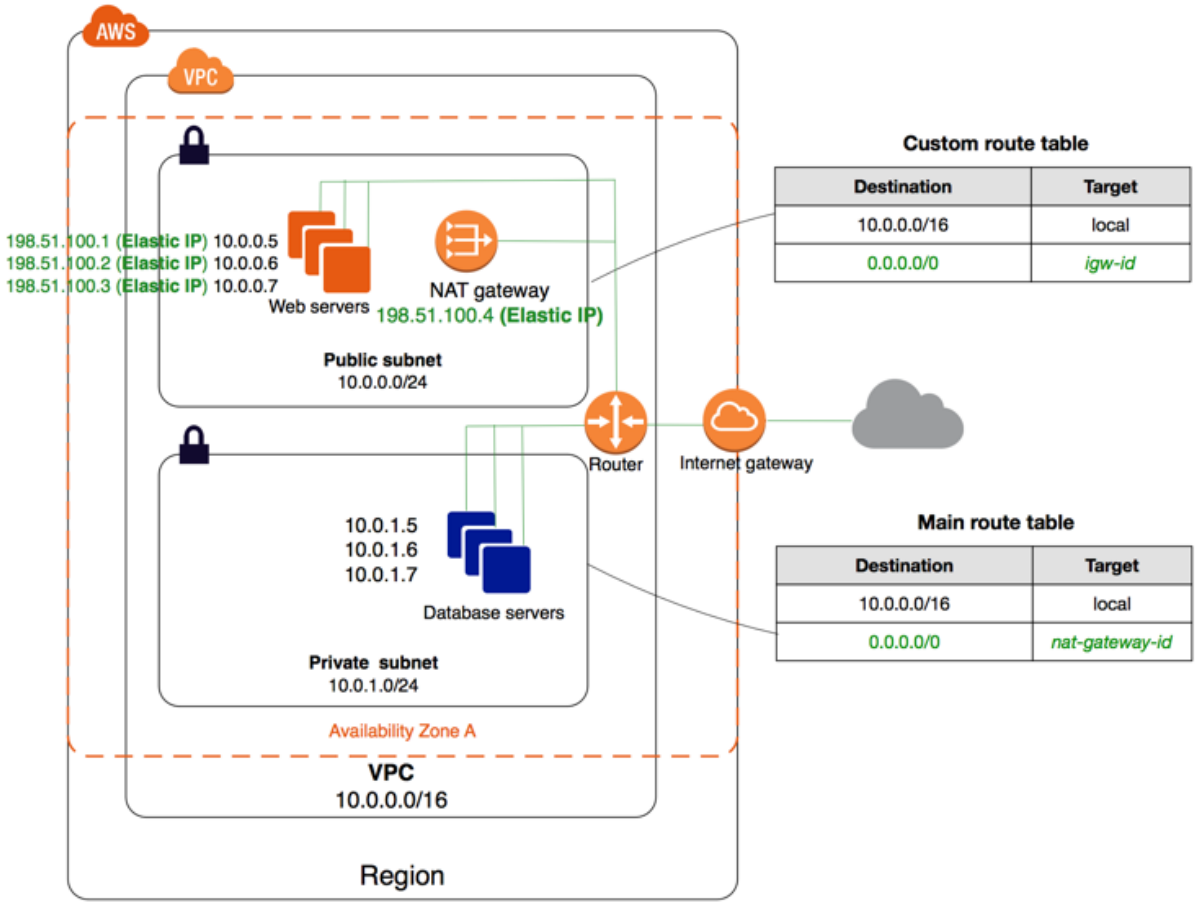
- When you create a VPC, you must specify an IPv4 CIDR Block for the VPC. The allowed block size is between /16 to /28.
- The fist four and last IP addresses of subnet cannot be assigned.
- For Example – **10.0.0.0/24**
  Total host = 256
  Reserved host = 5
  Available host = 256-5 = 251

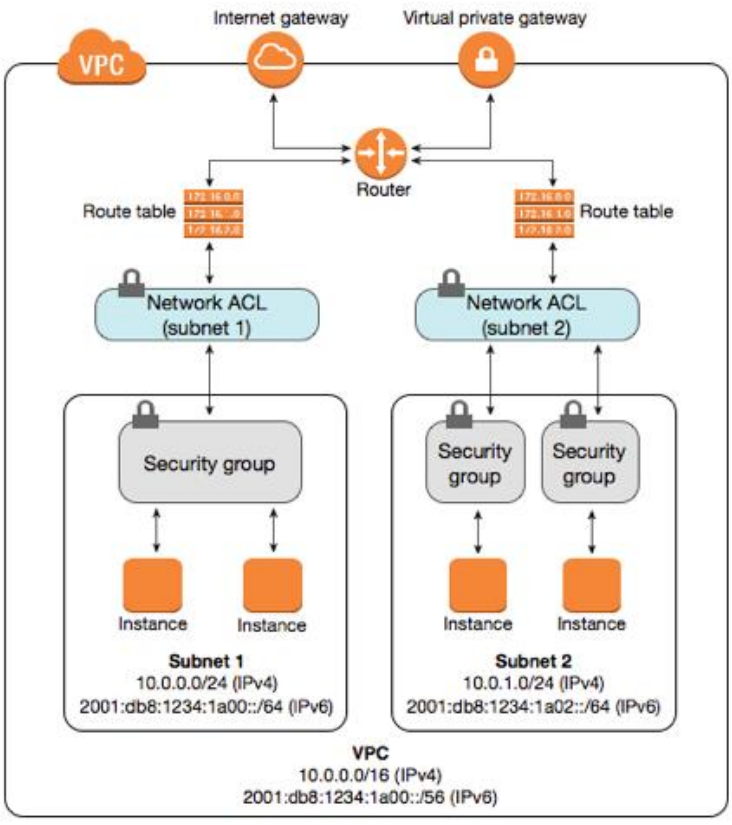
  Reserved host:
  1.0.0.0 – Networks address
  1.0.0.1 – Reserved by AWS for the VPC router
  1.0.0.2 – The IP address of DNS server (Reserved by AWS)
  1.0.0.3 – Reserved for future use
  .
  .
  10.0.0.255 – Broadcast address

ssh -i keyname.pem **ubuntu@**ip_address

chmod 600




AWS
VPC
198.51.100.1 (Elastic IP) 10.0.0.5
198.51.100.2 (Elastic IP) 10.0.0.6
198.51.100.3 (Elastic IP) 10.0.0.7
Web servers
NAT gateway
198.51.100.4 (Elastic IP)
Public subnet
10.0.0.0/24
Router
Internet gateway
10.0.1.5
10.0.1.6
10.0.1.7
Database servers
Private subnet
10.0.1.0/24
Availability Zone A
VPC
10.0.0.0/16
Region
Custom route table
Destination
Target
10.0.0.0/16
local
0.0.0.0/0
igw-id
Main route table
Destination
Target
10.0.0.0/16
local
0.0.0.0/0
nat-gateway-id


Internet gateway
Virtual private gateway
VPC
Router
Route table
Route table
Network ACL
(subnet 1)
Network ACL
(subnet 2)
Security group
Security group
Security group
Instance
Instance
Instance
Instance
Subnet 1
10.0.0.0/24 (IPv4)
2001:db8:1234:1a00::/64 (IPv6)
Subnet 2
10.0.1.0/24 (IPv4)
2001:db8:1234:1a02::/64 (IPv6)
VPC
10.0.0.0/16 (IPv4)
2001:db8:1234:1a00::/56 (IPv6)

VPC Endpoints

A VPC endpoint enables you to privately connect your VPC to supported AWS services.

Instances in your VPC do not require public IP address to communicate with resources in the service.

Practice :-

1. Create a VPC.
2. Create two subnets (Public & Private)
3. Create internet gateway and attach it to your VPC.
4. Create two route tables, One for Public subnet and one for private subnet.
5. Create two EC2 server – one in public subnet and another one in private subnet.

6. Now go into – VPC Services – Endpoints
7. Create an Endpoint – Select AWS Services – select S3, type – gateway.
8. Select your VPC – select your private subnet – create endpoint.

Login to your private instance through public instance.

To get your AWS access key ID and secret access key – Open (Login) the AWS – click on your name (**Account name**) on the right top side. Click on **security credentials** – go down and click on **create access key,** check the I understand box and proceed. And download the .csv file. You will see your Access key ID and secret access key in this file.

Commands:

aws configure

Fill your access key ID & secret access key

aws s3 mb s3://Bucketname (mb = make bucket) (This command is used to create a s3 bucket in your AWS account)

aws s3 ls (To see the list of existing buckets)

aws s3 ls s3://Bucketname (To see the content, what is inside your bucket)

aws s3 rb s3://Bucketname (To remove (delete) the bucket), (rb = remove bucket).

Bucket must be empty, only after you can remove (delete) it.

If you are not able to use the command aws configure:

1. Download the AWS CLI bundle installer, by below command:

wget https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip

once download done run the below command:

unzip awscli-exe-linux-x86_64.zip

sudo ./aws/install

Now try to run the aws configure command and fill the credentials.

AWS Storage

Block storage: Block storage is suitable for transactional databases, random read/write Loads and structured database storage.

Block storage divide the data into the blocks, to be stored in evenly sized blocks for instance, a file can we split into evenly sized blocks before it is stored.

Data blocks stored in block storage would not contain meta data (data created, data modified, content type etc).

Block storage only keeps the address (index) where the data blocks are stored. It does not care what is in that block, just how to retrieve it when required.

Example of block storage – EBS

Data can be retrieved only through instance, where the EBS is connected.

Object storage: Object storage stores the file as a whole and does not divide them.

In object storage an object is – The file/data itself – Its meta data – object global unique ID.

The object globally unique ID is a unique identifier for the object (can be the object name itself) and it must be unique such that it can be retrieved this regarding where its physical storage location is.

Object storage cannot be mounted as a drive.

Example of object storage solution, Amazon S3, Dropbox.

Data can be retrieved from anywhere via http or https.

Simple storage services (S3):

- S3 is a storage for the internet. It has a simple webservice interface for simple storing and retrieving of any amount of data, anytime from anywhere from the internet.
- S3 is an object-based storage.
- You can not install operating system on S3.
- Data is stored in bucket.
- Max capacity of a Bucket is 5TB.
- You can create folders in your bucket.
- You cannot create nested Buckets.
- Bucket ownership is non-transferable.
- S3 Bucket is region specific.
- You can have up to 100 Buckets per account.

S3 Bucket Naming rules:

- S3 bucket name are globally unique.
- Bucket names cannot be changed.
- Bucket name must be at least 3 and no more than 63 characters.
- Bucket names are the part of URL, used to access a bucket.
- Bucket names can contain lowercase letter, numbers, and hyphen. Cannot use uppercase letter.
- Bucket name should not be an IP address.
- By default, Buckets and its objects are private. So only owner can access this bucket.

S3 Bucket Versioning:

- Bucket versioning is a S3 Bucket sub-resource, used to protect data/object, against accidental deletion or overwrites.
- Once you enable versioning on a bucket, it cannot be disabled, however it can be suspended.

- When enabled, bucket versioning will protect existing and new objects, and maintain their versions as they are updated.
- When versioning is enabled, and you try to delete an object, a delete marker is placed on the object.
- You can still view the object and the delete marker.
- If you want to reconsider the deleted object, you can delete the “delete marker” and the object will be available again.
- Versioning applies to all the objects in the bucket.

Storage classes of Amazon S3:

S3 Standard: This is the default storage class, offering high durability, availability, and low latency. It is suitable for frequently accessed data and is designed for high-performance applications.

S3 Intelligent-Tiering: This storage class automatically moves objects between two access tiers: frequent and infrequent access, based on changing access patterns. It is designed to optimize costs for data with unknown or changing access patterns.

S3 One Zone-IA (Infrequent Access): Data stored in this class is stored in a single availability zone and is ideal for infrequently accessed data that can be recreated if lost.

S3 Glacier: This class is designed for archiving data that is rarely accessed. Data retrieval times can be longer compared to other classes, but it offers very cost-effective storage.

S3 Glacier Deep Archive: This storage class offers the lowest cost storage for archiving data, but with even longer retrieval times. It is suitable for data that is rarely, if ever, accessed.

S3 Outposts: This class is used with AWS Outposts, allowing you to extend Amazon S3 storage to on-premises environments.

S3 Replication: While not a traditional storage class, S3 provides options for replicating data to other S3 buckets, which can be in the same or different regions. This can help improve data durability and availability.

S3 Object Lock: This feature enforces retention policies on objects to ensure they are not deleted or modified for a specified period. It helps with compliance and data protection.

Lab to practice:

- S3 Bucket creation.
- Enable versioning on S3 Bucket.
- Cross region replication (CRR).
- Lifecycle management

https://docs.aws.amazon.com/AmazonS3/latest/userguide/replication.html

Requirements for replication

Replication requires the following:

- The source bucket owner must have the source and destination AWS Regions enabled for their account. The destination bucket owner must have the destination Region enabled for their account.

  For more information about enabling or disabling an AWS Region, see Managing AWS Regions in the *AWS General Reference*.
- Both source and destination buckets must have versioning enabled. For more information about versioning, see Using versioning in S3 buckets.
- Amazon S3 must have permissions to replicate objects from the source bucket to the destination bucket or buckets on your behalf. For more information about these permissions, see Setting up permissions.
- If the owner of the source bucket doesn't own the object in the bucket, the object owner must grant the bucket owner `READ` and `READ_ACP` permissions with the object access control list (ACL). For more information, see Access control list (ACL) overview.
- If the source bucket has S3 Object Lock enabled, the destination buckets must also have S3 Object Lock enabled.

  For more information, see Using S3 Object Lock. To enable replication on a bucket that has Object Lock enabled, contact AWS Support.

For more information, see Setting up replication.

If you are setting the replication configuration in a *cross-account scenario*, where the source and destination buckets are owned by different AWS accounts, the following additional requirement applies:

- The owner of the destination buckets must grant the owner of the source bucket permissions to replicate objects with a bucket policy. For more information, see Granting permissions when the source and destination buckets are owned by different AWS accounts.
- The destination buckets cannot be configured as Requester Pays buckets. For more information, see Using Requester Pays buckets for storage transfers and usage.

EFS (Elastic File System) LAB:

- It is a shared storage.
- Create two EC2 instance in different availability zone (diff subnet).
- Allow NFS in the security group.
- Go into storage – EFS – Create one EFS (Elastic file system).
- Choose the same security group, which you have made for your EC2 instances.
- Login your EC2 instance.
- Run command Sudo apt install nfs-common
- mkdir efs
- Copy the mount address (command) from EFS details, from attach option.
- Make sure the EFS security group must be same for both EC2 and EFS, otherwise you will not able to attach it.
- To check the security group in EFS – open your EFS – go into network option and check there.

EBS (Elastic block store): (NAS) (network attached storage).

It is persistent – Means data will be deleted only after termination.

It is network attached storage.

An EBS volume can attach to a single EC2 instance only, at a time.

Both EBS volume and EC2 instance must be in the same AZ.

EBS Volume Types:

- **Solid state drive (SSD) volumes** [Bootable]
  General purpose SSD Volumes - (GP2 & GP3).
  Provisioned IOPS SSD volumes – (iO2 & iO1)
- **Hard disk drive (HDD) volumes** [non-Bootable]
  Throughput optimized HDD Volumes – (st1)
  Cold HDD Volumes – (sc1)
- **Previous generation volumes** [Bootable]
  Magnetic

Instance store backed: Direct Attached Storage

EBS Snapshot:

- EBS snapshots are point in time images/copies of your EBS volume.
- Any data written to the volume after the snapshot process is initiated, will not be included in the resulting snapshot.
- Per AWS account, up to 5000 EBS volumes can be created.
- Per account, up to 10,000 snapshots can be created.
- While EBS volumes are AZ specific, the snapshots are region specific.
- Any AZ in region can use snapshot to create EBS volume.
- To migrate an EBS from one AZ to another – create a snapshot and create an EBS volume from the snapshot in the intended AZ.
- You can create a snapshot to an EBS volume of the same or larger size than the original volume.
- You can take a snapshot or a non-root (boot) EBS volume, while the volume is in the use on a running EC2 instance.
- This means, you can still access the non-root volume, while the snapshot is being processed.

- However, the snapshot will only include that data, that is already written to your volume.
- The snapshot is created immediately, but it may stay in pending state, until the full snapshot is completed. This may take few hours to complete, specially for the first-time snapshot of a volume.
- During the period when the snapshot status is pending, you can still access the volume (non-root), but I/O might be slower because of the snapshot activity.
- While in pending state, an in-progress snapshot will not include data from ongoing read and writes to the volume.
- To take complete snapshot of your non-root EBS volume – stop or unmount the volume.
- To create a snapshot for a root EBS volume, you must stop the instance first then take the snapshot.
- EBS snapshots are stored incrementally.
- After the first snapshot, further snapshots will only carry the changed blocks (incremental updates).
- EBS is AZ specific and snapshot is region specific.

Load balancer:

A load balancer is a service that distributes incoming network traffic across multiple servers to ensure that no single server becomes overwhelmed with too many requests. This helps improve the availability and fault tolerance of applications.

There are three main types of load balancers in AWS:

Classic Load Balancer (CLB): This is the original load balancer offered by AWS. It can distribute traffic across multiple EC2 instances and is capable of handling both HTTP and HTTPS traffic.

Application Load Balancer (ALB): ALB operates at the application layer (Layer 7) of the OSI model, and it is best suited for distributing HTTP and HTTPS traffic. It can route traffic based on the content of the request, such as the URL or the content of the request headers.
Network Load Balancer (NLB): NLB operates at the transport layer (Layer 4) of the OSI model. It is designed to handle TCP, UDP, and TLS traffic. NLB is capable of handling extremely high request rates and is often used for low-latency applications.

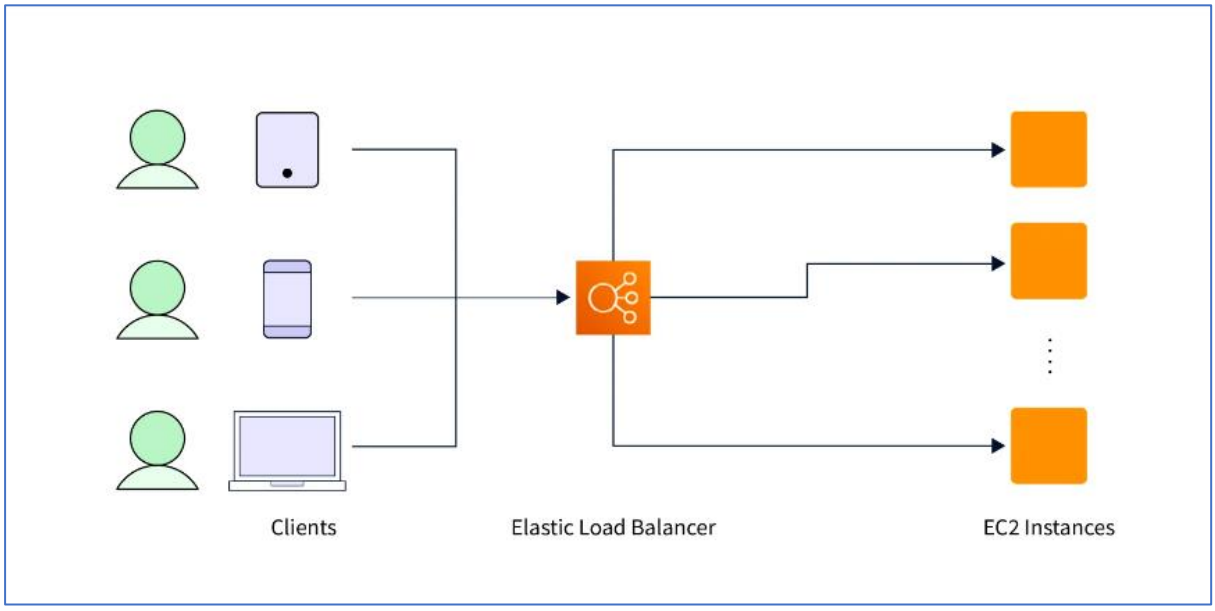
Key benefits of using a load balancer in AWS include:

High Availability: If one server becomes unavailable, the load balancer automatically routes traffic to the healthy servers, ensuring continuity of service.

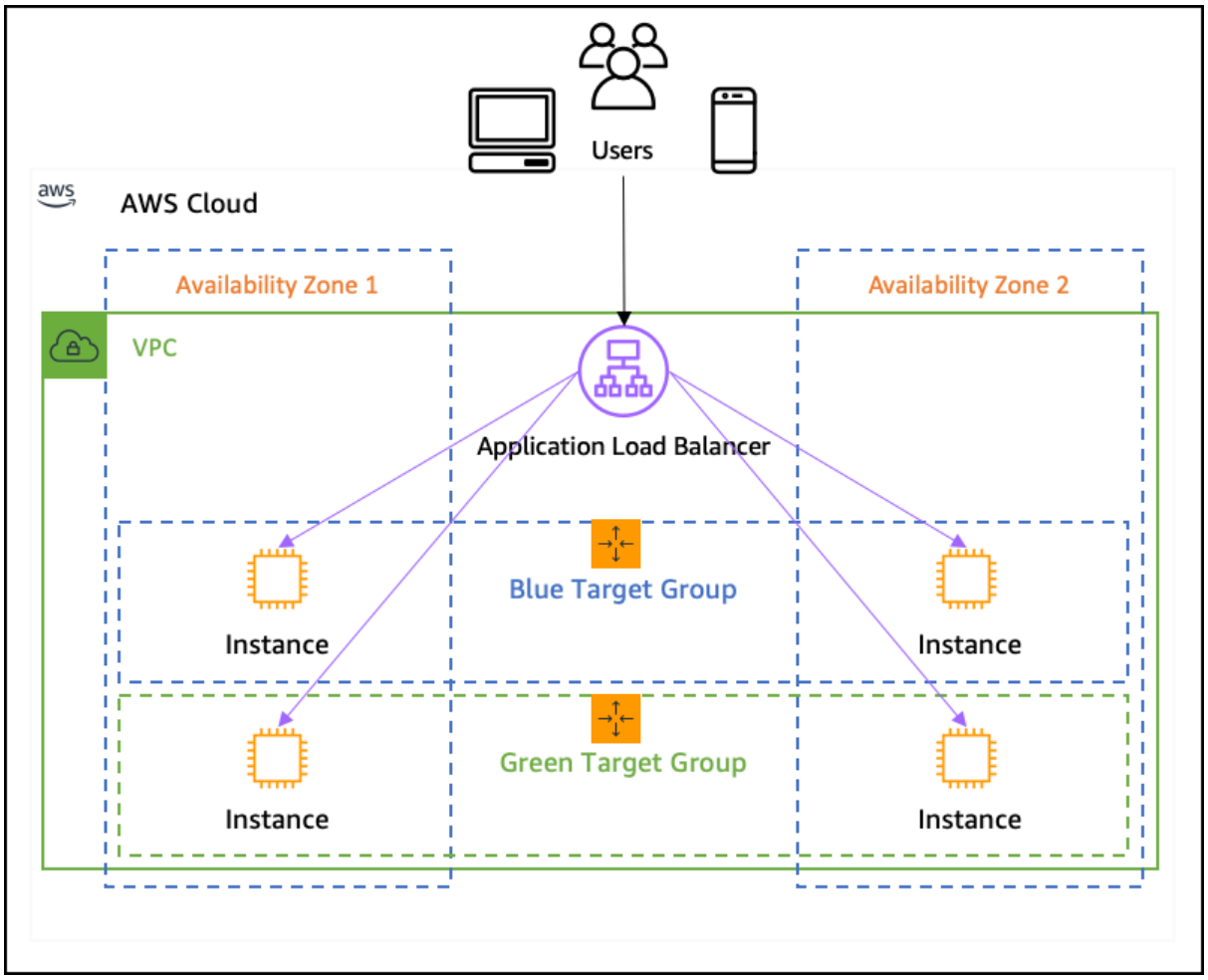
Scalability: Load balancers can distribute traffic evenly across a pool of servers, allowing you to easily scale your application horizontally by adding more servers.

Fault Tolerance: Load balancers can perform health checks on the servers to determine if they are responsive. Unresponsive servers are automatically taken out of rotation until they recover.

Classic load balancer:



Application load balancer:



Listeners and target group in ALB:

Listeners and target groups are fundamental components of an Application Load Balancer (ALB) in AWS. They work together to route incoming traffic to the appropriate targets, such as EC2 instances, containers, or Lambda functions.

1. **Listeners**:
    - A listener is a process that checks for connection requests. It is responsible for accepting incoming traffic on a specific port (or multiple ports) and protocol.
    - In the context of an ALB, listeners are configured to receive requests from clients.
    - ALB supports various protocols such as HTTP, HTTPS, and WebSocket. When you configure a listener, you specify a protocol and a port number.
    - For example, you might configure an HTTP listener on port 80 and an HTTPS listener on port 443.

- You can also configure rules within a listener to route traffic based on specific conditions (e.g., request path, host header).

2. **Target Groups**:
    - A target group is a logical grouping of targets (such as EC2 instances) that will receive traffic from a particular listener.
    - It acts as a destination for requests that the ALB receives. When a request arrives at the ALB, it determines which target group to forward the request to based on the listener's rules.
    - A target group can include targets in one or more Availability Zones within a single AWS Region.
    - You can specify health checks for the targets within a target group to ensure that traffic is only directed to healthy instances.
    - If a target group contains multiple targets, the ALB will route requests across them based on the load balancing algorithm configured (e.g., round robin, least outstanding requests).
3. **How They Work Together**:
    - When a request comes into the ALB, the listener checks which rule matches the request based on the conditions specified (e.g., path patterns, host headers).
    - Once a rule is matched, the listener forwards the request to the associated target group.
    - The target group then determines which target(s) should receive the request based on its own configuration and health status of the targets.
    - The request is finally sent to the selected target(s) within the target group.

In summary, listeners define the protocol and port where the ALB listens for incoming traffic, while target groups define where the traffic should be routed to. Listeners help determine which target group to send the request to based on the rules defined within the listener.

Auto Scaling

Autoscaling in AWS (Amazon Web Services) is a feature that allows you to automatically adjust the number and size of your compute resources (such as EC2 instances) based on the current demand for your application. The goal of autoscaling is to ensure that your application can handle varying levels of traffic without manual intervention.

Here's how autoscaling works:

1. Monitoring Metrics: Autoscaling relies on CloudWatch metrics or custom metrics that you define. These metrics could include things like CPU utilization, network traffic, or any other performance indicator relevant to your application.

2. Scaling Policies: You set up scaling policies that define the conditions under which new instances are launched or existing instances are terminated. These policies are based on the thresholds you establish in your chosen metrics. –

**Scale Out Policy: When the metric exceeds a certain threshold (e.g., high CPU usage), autoscaling will add more instances to your group. –

**Scale In Policy: When the metric falls below a certain threshold (e.g., low CPU usage), autoscaling will remove instances from your group.

3. Launch Configurations/Templates: Autoscaling uses a launch configuration or a launch template that defines the specifications for the new instances. This includes the Amazon Machine Image (AMI), instance type, security groups, and other settings.

4. Autoscaling Groups: These are the entities that manage your EC2 instances. An autoscaling group is a collection of Amazon EC2 instances that are created from a common Amazon Machine Image (AMI). The group ensures that a specified number of instances are running at all times.

5. Desired Capacity: You specify the desired number of instances that should be running at any given time. Autoscaling will automatically adjust the number of instances to meet this desired capacity.

6. Cooldown Period: After a scaling activity (launching or terminating instances), a cooldown period is enforced. During this period, further scaling activities are suspended to allow the newly launched instances to start and stabilize.

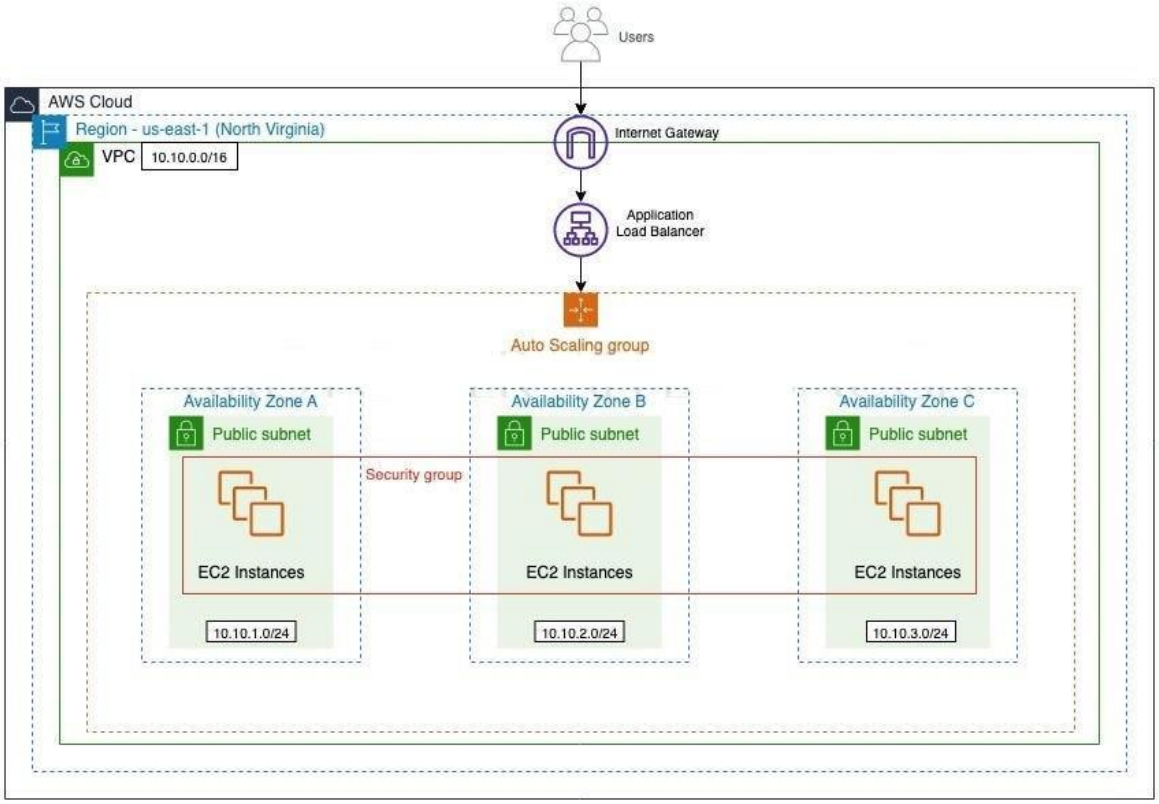
Benefits of Autoscaling:

1. Cost Optimization: Autoscaling helps to optimize costs by ensuring you only run the necessary number of instances to handle your current load. This prevents over-provisioning of resources.

2. High Availability: It increases the availability and fault tolerance of your applications by ensuring that there are always a sufficient number of instances available.

3. Performance Optimization: Autoscaling can dynamically adjust the capacity of your application to maintain consistent performance even during traffic spikes.

4. Time Efficiency: It automates the process of adding or removing instances, reducing the need for manual intervention. Overall, autoscaling is a crucial component of building scalable and resilient applications in the AWS cloud environment.



Identity and Access management (IAM)

AWS IAM, or Amazon Web Services Identity and Access Management, is a service that allows you to manage access to AWS resources securely. It enables you to control who can access your AWS services and what actions they can perform.

IAM is a global service means it's not region specific.

Here are some key points about AWS IAM:

1. **Users and Groups:**
    - IAM allows you to create users and groups. Users are individual AWS accounts, while groups are collections of users.
    - Each user can have specific permissions assigned to them, and users can be added or removed from groups to manage access more efficiently.
2. **Policies:**
    - IAM policies are JSON documents that define permissions. They can be attached to users, groups, or resources.
    - A policy specifies the actions a user, group, or resource can take, as well as the resources they can perform those actions on.
3. **Roles:**
    - IAM roles are similar to users, but they are not associated with a specific person or identity. Instead, they are intended to be assumed by other AWS services or resources.
    - Roles are commonly used in scenarios like granting permissions to AWS Lambda functions, EC2 instances, or other services.
4. **Access Key and Secret Key:**
    - IAM users can have access keys associated with their accounts. These consist of an Access Key ID and a Secret Access Key. These keys are used to authenticate API requests made to AWS services.
5. **Multi-factor Authentication (MFA):**
    - IAM supports the use of MFA, which adds an extra layer of security to user accounts. This requires users to provide a second form of authentication (e.g., a time-based one-time password) in addition to their username and password.
6. **Identity Federation:**
    - IAM can integrate with external identity providers, such as Active Directory or SAML-based systems, allowing you to grant access to AWS resources using existing credentials.
7. **Resource-based Policies:**

- In addition to attaching policies to users and groups, you can also attach policies directly to AWS resources like S3 buckets, SQS queues, etc.

8. **Policy Conditions:**
    - IAM policies can include conditions that must be met for the policy to take effect. For example, you can create a policy that allows certain actions only if they are requested from a specific IP range.
9. **Access Analyzer:**
    - IAM Access Analyzer helps you identify resources that can be accessed by external principals, allowing you to review and refine your policies for better security.
10. **Audit Trails and Logging:**
    - AWS IAM provides detailed logs that track user activity, which can be used for auditing and compliance purposes.

IAM is a critical component of AWS security and access management. It allows you to implement the principle of least privilege, ensuring that users and resources have only the permissions they need to perform their tasks. This helps enhance the security of your AWS environment.

THERE ARE SEVERAL WAYS TO APPLY POLICIES TO USERS:

1. **Attaching Policies Directly to Users**:
    - You can attach policies directly to individual IAM users. This means that the user will have the permissions specified in the attached policy.
    - When you attach a policy directly to a user, the permissions specified in that policy apply only to that user.
2. **Adding Users to Groups with Policies**:
    - Instead of attaching policies directly to users, you can create IAM groups and attach policies to these groups.
    - Users can then be added to these groups. By doing so, they inherit the permissions associated with the group.
    - This is a more scalable way to manage permissions, especially in situations where multiple users require similar levels of access.
3. **Applying Policies to Roles**:
    - IAM roles are not associated with a specific user, but instead, they are intended to be assumed by other AWS services or resources.
    - Policies can be attached directly to roles. These policies define what actions the role can take and what resources it can act on.
    - Roles are commonly used in scenarios like granting permissions to AWS Lambda functions, EC2 instances, or other services.

Difference between root and admin user

1. account level CONFIGURATION (contact info, etc.) cannot be done by IAM admin and it can only be done by root user.

2. Suppose by chance if admin user(A) has removed its own permission by mistake and in this case only root user is there who can give back the admin permission to A.

3. Billing and cost management access is not assigned to admin user by default but these access are assign to root user by default.

4. Only root user can close your AWS account and IAM admin can't close the account.

5. Support plan can only be changed by root user and not by IAM admin.

In AWS Identity and Access Management (IAM), when a user is associated with multiple groups, the permissions are cumulative. This means that if you attach a user to two groups—one with permissions to read and write in Amazon S3, and another with explicit denials to read and write in S3—the user will ultimately be denied those permissions. In IAM, explicit denials take precedence over permissions granted. This is known as the "deny overrides" principle. So, if any group the user belongs to denies a certain permission, even if the other groups grant that permission, the denial will take precedence and the user will be denied access.

CloudFront

Amazon CloudFront is a content delivery network (CDN) service provided by Amazon Web Services (AWS). It is designed to accelerate the delivery of web content to users by distributing it through a network of data centres around the world. CloudFront helps reduce latency and improve the overall performance of websites, APIs, and other web applications by caching and delivering content from the edge locations closest to the end-users.

Here are some key features and concepts associated with Amazon CloudFront:

Content Distribution: CloudFront helps distribute content, such as web pages, images, videos, and other static or dynamic assets, to multiple edge locations globally. These edge locations are strategically positioned in various geographic regions, allowing users to access content from a nearby location, reducing the time it takes to load web pages and resources

Content Caching: CloudFront caches copies of your content at its edge locations. When a user requests a piece of content, CloudFront checks if it already has a cached copy at the nearest edge location. If it does, CloudFront serves the cached copy, reducing the load on your origin server and improving response times.

Origin Server: The origin server is the source of the original content. This can be an Amazon S3 bucket, an AWS Elastic Load Balancer, an EC2 instance, or an on-premises server. CloudFront retrieves content from the origin server as needed and caches it at the edge locations.

Customization: CloudFront allows you to customize the behaviour of your distributions, including setting cache behaviours, content compression, and domain aliases (custom domain names)

Low Latency: By serving content from edge locations distributed worldwide, CloudFront helps reduce the latency for end-users, improving the overall user experience.

Amazon CloudFront is commonly used for websites, web applications, and APIs that require low latency, high availability, and efficient content delivery to a global audience. It is part of AWS's extensive suite of cloud computing services and can be easily integrated with other AWS services to create scalable and highly performant applications.

Databases

Q: What is data?

A: Data can be facts related to any object. For example, your age, your address, your contact no.

Q: what is database?

A: Database is a systematic collection of data. Databases support storage and manipulation of data.

Q: what is DBMS (database management system)?

A: DBMS is a collection of programme which enable its users to access database, manipulate data, reporting/representing of data.

Important types of databases :-

- relational database
- non-relational database

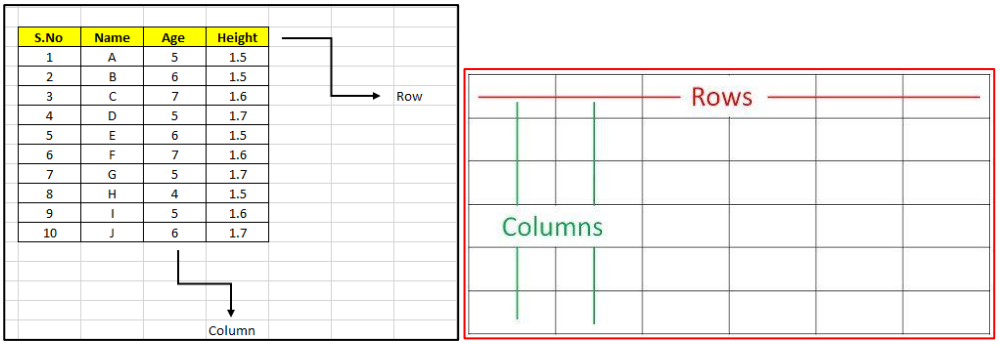
Relational database (SQL DB)

A relational database is a type of database that organizes and stores data in a structured format.

- A relational database is a data structure that allows you to link information from different 'tables' or different types of data bucket.
- tables are related to each other.
- all fields must be filled.
- best suited for OLTP (online transactional processing).
- Relational databases: MySQL, Oracle DBMS, IBM DB2

Name	Age	Location	Contact
X	21	India	12345623
Y	23	China	21345612
Z	34	USA	32345634

- Row
- Column



- A ROW OF A TABLE IS ALSO CALLED RECORD. IT CONTAINS THE SPECIFIC INFORMATION OF EACH INDIVIDUAL ENTRY IN THE TABLE.
- EACH TABLE HAS ITS OWN PRIMARY KEY.
- A SCHEMA (DESIGN OF A DATABASE) IS USED TO STRICTLY DEFINES, TABLE, COLUMN, INDEXES AND RELATION BETWEEN TABLES.
- RELATIONAL DB ARE USUALLY USED IN ENTERPRISES APPLICATION/SCENARIO. EXCEPTION IN MYSQL, WHICH IS USED FOR WEB APPLICATION.
- CANNOT SCALE OUT HORIZONTALLY.
- VIRTUALLY ALL RELATIONAL DB USES SQL.

Roll no	Name	Branch
101	Ajay	EC
102	Abhay	CS
103	Sonu	IT

Roll no	Subject	Grade
101	Math	A
102	ENG	A+
103	SCI	B

Roll no	Add.	Contact
101	USA	11111111
102	UK	22222222
103	IN	Null

Subject	Teacher
Math	Rakesh
ENG	Suresh
SCI	Mukesh

LAB:

1. Launch an Ubuntu EC2 instance.
2. Allow MYSQL/Aurora – port no 3306 in the security group.
3. Create a RDS Mysql database instance.
4. Connect your Ubuntu server via SSH.
5. Update and install the MYSQL-server client on your Ubuntu machine.
6. Now connect to your MYSQL database/instance
   MYSQL -h "endpoint-name-of-sql-db" -u "username" -p
   Enter your password which you have created, when u launched the SQL-DB.
7. Create one database - CREATE DATABASE databasename;
8. SHOW DATABASES;
9. USE databasename; (to swich into database)
10. Now-

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    email VARCHAR(100),
    created_at TIMESTAMP
);
```

11. SHOW TABLES;
12. Now insert something in this table –

```
INSERT INTO users (user_id, username, email, created_at) VALUES
('0','rahul','rahul@gmail.com',NOW());
```

13. SELECT * FROM tablename;
14. DELETE FROM users
    WHERE user_id = 5;

Once done this lab, delete the RDS instance and EC2 instance.

And in RDS instance don't take the backup. Just unmark that option.

Non-relational database (No-SQL DB)

- Non-relational databases, stores data without a structured mechanism to link data from different tables to one another.
- Require low cost hardware.
- Much faster performance (read/write) compare to relational database.
- Horizontal scaling is possible.
- Never provide table with flat fixed column records. It means schema free.
- Best suited for online analytical processing.

- examples of no-sql databases: MongoDB, Cassandra, DynamoDB, PostegreSQL.

Types of NoSQL Databases----------

NoSQL databases are designed to handle various types of unstructured or semi-structured data and provide flexibility, scalability, and high performance. There are several types of NoSQL databases, each designed for specific use cases. Here are the main types:

1. **Document-Oriented Databases:**

Example: MongoDB, CouchDB, RavenDB.

Description: These databases store data in documents, which are similar to JSON or XML files. Each document can have a different structure, making them very flexible. documents db are efficient for storing catalogue.

2. **Key-Value Stores:**

Examples: Redis, DynamoDB, Riak.

Description: Data is stored in a key-value pair format, where each piece of data is associated with a unique key. These databases offer high-speed access and are used for caching, real-time analytics, and session management.

3. **Column-Family Stores (Wide-Column Stores):**

Examples: Cassandra, HBase, ScyllaDB.

Description: Data is organized in columns, which are grouped together in column families. This design allows for efficient storage and retrieval of large amounts of data and is often used in big data applications. and it uses less disk space that a RDBMS containing the same data. it can easily perform the operation like – min, max & avg.

4. **Graph Databases:**

Examples: Neo4j, Amazon Neptune, OrientDB.

Description: These databases use graph structures to represent and store data. Nodes represent entities, and edges represent relationships between them. They are ideal for applications that involve complex relationships, such as social networks, recommendation engines, and fraud detection.

DATABASES TYPES:

1. **UNSTRUCTURED DATA**: IS INFORMATION THAT EITHER DOES NOT HAVE A PRE-DEFINED DATA MODEL OR IS NOT ORGANISED IN A PRE-DEFINED MANNER.
   IT CAN BE TEXT-HEAVY OR CONTAIN MEDIA FILES (IMAGES, AUDIO, VIDEO).
   IT LACKS A SPECIFIC SCHEMA, MAKING IT DIFFICULT TO PROCESS WITH TRADITIONAL DATABASE SYSTEMS.
   IT REQUIRES ADVANCED TECHNIQUES LIKE NATURAL LANGUAGE PROCESSING (FOR TEXT) OR COMPUTER VISION (FOR IMAGES AND VIDEOS) FOR MEANINGFUL ANALYSIS.
   EXAMPLES:
   TEXT DOCUMENTS (PDFS, WORD DOCUMENTS).
   SOCIAL MEDIA POSTS.
   IMAGES AND VIDEOS.
   EMAILS.
   USE CASES:
   IMAGE RECOGNITION IN SECURITY SYSTEMS.
   VOICE-TO-TEXT PROCESSING.

2. **SEMI-STRUCTURED DATA:** SEMI STRUCTURED DATA IS INFORMATION THAT DOES NOT RESIDE IN A RELATIONAL DATABASE, BUT IT DOES HAVE SOME ORGANISATIONAL PROPERTIES THAT MAKE IT EASIER TO ANALYSE
   CHARACTERISTICS:
   IT CAN BE REPRESENTED IN VARIOUS FORMATS LIKE JSON, XML, YAML.
   IT OFTEN CONTAINS METADATA THAT DESCRIBES THE CONTENT.
   EXAMPLES:
   JSON OR XML DOCUMENTS.
   LOG FILES.
   USE CASES:
   WEB APIS THAT RETURN DATA IN JSON FORMAT.
   CONFIGURATION FILES.
   DATA EXCHANGED BETWEEN DIFFERENT SYSTEMS.

3. **STRUCTURED DATA:** STRUCTURED DATA REFERS TO DATA THAT IS ORGANIZED IN A HIGHLY PREDICTABLE AND ORGANIZED MANNER. IT FOLLOWS A SPECIFIC, PRE-DEFINED DATA MODEL OR SCHEMA. THIS TYPE OF DATA IS TYPICALLY FOUND IN RELATIONAL DATABASES OR SPREADSHEETS.
   CHARACTERISTICS:
   IT IS ORGANIZED INTO TABLES, ROWS, AND COLUMNS.
   EACH FIELD HAS A SPECIFIC DATA TYPE (E.G., INTEGERS, STRINGS, DATES).
   IT'S EASILY SEARCHABLE, QUERYABLE, AND CAN BE PROCESSED USING STRUCTURED QUERY LANGUAGES LIKE SQL.
   EXAMPLES:
   RELATIONAL DATABASES (MYSQL, POSTGRESQL, ORACLE).
   EXCEL SPREADSHEETS.

USE CASES:

FINANCIAL RECORDS.

INVENTORY MANAGEMENT.

CUSTOMER INFORMATION IN CRM SYSTEMS.

DYNAMODB AND IT'S FEATURES:

1. **DYNAMODB TABLE:**
   - A TABLE IS A COLLECTION OF DATA ITEMS.
   - LIKE ALL OTHER DATABASE, DYNAMODB STORES DATA IN TABLES.
2. **ITEMS:**
   - EACH TABLES CONTAINS MULTIPLE ITEMS.
   - AN ITEM IS A GROUP OF ATTRIBUTES, THAT IS UNIQUELY IDENTIFIABLE AMONG ALL OF THE OTHER ITEMS.
   - AN ITEM CONSIST OF A PRIMARY OR COMPOSITE KEY AND A FLEXIBLE NUMBERS OF ATTRIBUTE.
3. **ATTRIBUTE:**
   - EACH ITEM IS COMPOSED OF ONE OR MORE ATTRIBUTES.
   - AN ATTRIBUTE CONSIST OF THE ATTRIBUTE NAME AND A VALUE OF A SET OF VALUES.
   - AN ATTRIBUTE IS A FUNDAMENTAL ELEMENT, SOMETHING THAT DOES NOT NEED TO BE BROKEN DOWN ANY FURTHER.

TABLE

ITEM

C. No	23
Name	Rahul
Age	30
Add	Del

ITEM

C. No	45
Name	Ajay
City	Gwl
Home no	C204

ITEM

C. No	76
Hobby	Singing
Position	Manager
Marital S.	Married

C. No = PRIMARY KEY

AND REST ALL THE INFORMATION = ATTRIBUTES

NOTES:

ITEM SIZE: MUST NOT MORE THAN 400KB. IF IT IS GREATER THAN 400KB THEN YOU MUST PUT THAT DATA IN S3 BUCKET AND PASTE THE URL HERE IN ITEMS.

- DYNAMODB ALLOWS LOW LATENCY READ/WRITE ACCESS TO ITEMS RANGING FROM 1 BYTE TO 400KB.
- DYNAMODB CAN BE USED TO STORE POINTERS TO S3 STORED OBJECTS, OR ITEMS OF SIZE LARGER THAN 400KB TOO IF NEEDED.
- DYNAMODB STORES DATA INDEXED BY A PRIMARY KEY – YOU SPECIFY THE PRIMARY KEY WHEN YOU CREATE THE TABLE.
- EACH ITEM OF THE TABLE HAS A UNIQUE IDENTIFIER OR PRIMARY KEY.
- THE PRIMARY KEY IS THE ONLY REQUIRED ATTRIBUTE FOR ITEMS IN A TABLE.
- DYNAMODB TABLES ARE SCHEMA LESS, WHICH MEANS THAT NEITHER THE ATTRIBUTES NOR THEIR DATA TYPES NEED TO BE DEFINED BEFOREHEAD.
- EACH ITEMS CAN HAVE ITS OWN DISTINCT ATTRIBUTES.

AWS SIMPLE NOTIFICATION SERVICE (SNS)

1. SNS IS A FAST, FLEXIBLE, FULLY MANAGED, PUSH NOTIFICATION SERVICE.
2. IT IS A WEBSERVICE THAT CO-ORDINATES AND MANAGES THE DELIVERY OR SENDING OF MESSAGES TO SUBSCRIBING ENDPOINTS OR CLIENTS.
3. IT ALLOWS FOR SENDING INDIVIDUAL MESSAGES OR FAN OUT MESSAGES TO A LARGE NUMBER OF RECIPIENTS OR TO OTHER DISTRIBUTED SERVICES.
4. MESSAGES PUBLISHED TO AN SNS TOPICS WILL BE DELIVERED TO THE SUBSCRIBER IMMEDIATELY.
5. INEXPENSIVE, PAY-AS-YOU-GO MODEL WITH NO UPFRONT COST.
6. RELIABLE – AT LEAST THREE COPIES OF THE DATA ARE STORE ACROSS MULTIPLE AZ IN SAME REGION.
7. IT IS A WAY OF SENDING MESSAGES, WHEN YOU ARE AUTOSCALING, IT TRIGGERS AN SNS SERVICE WHICH WILL EMAIL YOU THAT 'YOUR EC2 INSTANCE IS GROWING'.

PUBLISHER: PUBLISHERS ARE ALSO KNOWN AS PRODUCERS THAT PRODUCE AND SEND THE MESSAGE TO THE SNS WHICH IS A LOGICAL ACCESS POINT.

SUBSCRIBER: SUBSCRIBERS SUCH AS WEBSERVERS, EMAIL ADDRESSES, AMAZON SQS QUEUE, AWS LAMBDA, RECEIVE THE MESSAGES FROM THE SNS OVER ONE OF THE SUPPORTED PROTOCOLS.

SNS TOPIC:

- IT IS A LOGICAL ACCESS POINT AND COMMUNICATION CHANNEL.

- EACH TOPIC HAS A QUEUE NAME.

- A TOPIC NAME IS LIMITED TO 256 ALPHANUMERIC CHARACTERS.

- THE TOPIC NAME MUST BE UNIQUE WITHIN THE AWS ACCOUNT.

- EACH TOPIC IS ASSIGNED AN AWS ARN ONCE IT GETS CREATED.

- A TOPIC CAN SUPPORT SUBSCRIBERS AND NOTIFICATION DELIVERIES OVER MULTIPLE PROTOCOLS.

- MESSAGES/REQUESTS PUBLISHED TO A SINGLE TOPIC CAN BE DELIVERED OVER MULTIPLE PROTOCOLS AS CONFIGURED WHEN CREATING EACH SUBSCRIBER.

- DELIVERY FORMAT/TRANSFER PROTOCOLS (ENDPOINT) – SMS, E-MAIL, HTTP/HTTPS, SQS, LAMBDA