Module 2

S3 (Simple Storage Service) is a highly scalable, **object-based storage service** offered by Amazon Web Services (AWS). It provides reliable, durable, and secure storage for a wide range of data, including images, videos, documents, and applications.

Object-based storage is a type of data storage technology that stores data as individual objects, rather than as files or blocks. Each object is assigned a unique identifier and metadata that describes the object, such as its size, content type, and creation date. This metadata is stored along with the object, making it easy to manage and access.

Key components include:

- **Data Centres:** S3 data is distributed across multiple data centres globally, ensuring redundancy and fault tolerance.
- **Storage Nodes:** Each data centre has numerous storage nodes that store data in a distributed manner.
- **Network:** A high-speed network connects storage nodes and data centres, enabling fast data transfer.

Durability and Availability

- **Durability:** S3 offers 99.999999999% durability, meaning your data is designed to **endure (continue to exist for a long time)** for years. Data is replicated across multiple storage nodes within multiple data centers.
- **Availability:** S3 provides 99.99% availability, ensuring your data is accessible at all times. Even if a data center or storage node fails, your data remains available through replication.

Object Lifecycle

- **Versioning:** S3 supports versioning, allowing you to track and manage multiple versions of an object. This helps prevent accidental deletion and provides a historical record of your data.
- **Expiration:** You can set expiration policies for objects, automatically deleting them after a specified time.
- **Lifecycle Rules:** S3 allows you to create lifecycle rules that define actions to be taken on objects based on their age or other criteria. This includes transitioning objects to different storage classes for cost optimization or archiving.

Accessing S3 Objects

- **AWS Management Console:** You can use the AWS Management Console to manage S3 buckets, objects, and permissions.
- **AWS CLI:** The AWS Command Line Interface (CLI) provides a powerful way to interact with S3 programmatically.

- **AWS SDKs:** AWS Software Development Kits (SDKs) are available for various programming languages, allowing you to access S3 from your applications.
- **REST API:** S3 provides a RESTful API that you can use to interact with the service directly.

Amazon S3 Glacier

- **Designed for long-term data storage:** Glacier is optimized for storing data that needs to be infrequently accessed, such as backups, historical records, and archival data.
- **Lower cost:** Glacier offers significantly lower storage costs compared to other S3 storage classes due to its optimized infrastructure for long-term retention.
- **Data retrieval time:** Retrieving data from Glacier can take several hours, so it's not suitable for frequently accessed data.
- **Storage classes:** Glacier offers two storage classes:
    - **Glacier Standard:** For data that needs to be retrieved within 24 hours.
    - **Deep Archive:** For data that can be retrieved within 12 hours.

AWS CLI

- **Command-line interface for AWS services:** The AWS CLI is a powerful tool for managing AWS resources from the command line.
- **S3 commands:** The AWS CLI provides numerous commands for interacting with S3, including creating buckets, uploading and downloading objects, managing permissions, and more.

Example: Uploading a file to S3 Glacier using the AWS CLI

1. **Install the AWS CLI:** If you haven't already, install the AWS CLI from the official AWS website.
2. **Configure the AWS CLI:** Configure the AWS CLI with your AWS credentials using the '**aws configure**' command.
3. **Create a Glacier vault:** Create a Glacier vault using the '**aws glacier create-vault**' command:
    1. aws glacier create-vault --vault-name my-glacier-vault
4. **Upload a file to the vault:** Upload a file to the vault using the '**aws glacier upload-archive**' command:
    1. aws glacier upload-archive --vault-name my-glacier-vault --description "My backup file" --client-request-token $(uuidgen) --body my-file.txt

This command will upload the file my-file.txt to the my-glacier-vault and return a job ID. You can use the job ID to check the status of the upload and retrieve the file later.

VPC (Virtual Private Cloud) is a service provided by AWS that allows you to create a virtual network within the AWS cloud. This virtual network provides you with complete control over your network environment, including the ability to define your own IP address range, create subnets, and control security settings.

Key Components of a VPC

1. **CIDR Blocks:** A CIDR (Classless Inter-Domain Routing) block is a contiguous range of IP addresses that you assign to your VPC. This range defines the entire IP address space available within your VPC.
2. **Subnets:** Subnets are divisions of your VPC that are used to group instances. You can create public subnets that have internet connectivity and private subnets that are isolated from the public internet.
3. **Elastic Network Interfaces (ENIs):** ENIs are virtual network interfaces that are attached to your instances. They provide the instances with connectivity to the VPC and the internet.
4. **Internet Gateways:** Internet Gateways are used to provide your VPC with internet connectivity. You can attach an internet gateway to a public subnet to enable instances in that subnet to communicate with the internet.
5. **Route Tables:** Route tables are used to determine how network traffic is routed within your VPC. Each subnet is associated with a route table, which contains rules that specify where network traffic should be sent.

Example Scenario

Imagine you want to create a web application with a public-facing frontend and a private backend database. You could create a VPC with two subnets:

- **Public subnet:** This subnet would be associated with an internet gateway and would contain instances that run your web application.
- **Private subnet:** This subnet would not be associated with an internet gateway and would contain instances that run your database.

You could then create a route table for the public subnet that includes a route to the internet gateway, and a route table for the private subnet that includes a route to a NAT gateway (a special type of instance that allows instances in a private subnet to access the internet). This configuration would ensure that your web application can be accessed from the public internet, while your database remains private.

Security Groups and Network Access Control Lists (NACLs)

Security groups and **network access control lists (NACLs)** are both used to control network traffic in AWS VPCs, but they have different scopes and functionalities.

Security Groups

- **Scope:** Apply to individual EC2 instances and other resources within a VPC.
- **Functionality:** Control inbound and outbound traffic based on source and destination IP addresses, ports, and protocols.
- **Rules:** Rules are stateful, meaning they track the state of connections and allow return traffic from authorized sources.
- **Priority:** Rules are evaluated in order of priority, with higher-priority rules taking precedence.

Example use case: Allow SSH traffic (port 22) from a specific IP address range to an EC2 instance.

Network Access Control Lists (NACLs)

- **Scope:** Apply to subnets within a VPC.
- **Functionality:** Control inbound and outbound traffic based on source and destination IP addresses, ports, and protocols.
- **Rules:** Rules are stateless, meaning they do not track the state of connections.
- **Priority:** Rules are evaluated in order of priority, with higher-priority rules taking precedence.
- **Default rules:** NACLs include default rules that allow all traffic from within the VPC and deny all traffic from outside the VPC.

Example use case: Deny all traffic to a subnet except for a specific set of IP addresses.

Key differences:

- **Scope:** Security groups apply to individual resources, while NACLs apply to subnets.
- **Statefulness:** Security groups are stateful, while NACLs are stateless.
- **Default rules:** NACLs have default rules that allow or deny all traffic, while security groups do not.

Best approach:

- Use both security groups and NACLs to create a layered security model.
- Apply security groups to individual resources to control access to specific applications or services.
- Apply NACLs to subnets to control traffic at the subnet level.

- Use a combination of security groups and NACLs to create a firewall that is both granular and effective.
- By understanding the differences between security groups and NACLs, you can effectively control network traffic in your AWS VPCs and protect your resources from unauthorized access.

Public IP Addresses, Elastic IP Addresses, NAT, and VPC Peering

Public IP Addresses

A **public IP address** is an IP address that is accessible from the internet. It allows you to connect to resources within your AWS account from outside the AWS network. When you launch an EC2 instance, it is assigned a public IP address by default.

Elastic IP Addresses

An **Elastic IP address** is a static IP address that you can associate with an EC2 instance or network interface. This allows you to maintain a consistent public IP address for your resources, even if the underlying EC2 instance changes.

Benefits of using Elastic IP addresses:

- **Consistent public IP:** Maintain a fixed IP address for your applications.
- **DNS management:** Use the Elastic IP address with your DNS records.
- **Load balancing:** Distribute traffic across multiple EC2 instances using a load balancer.

Network Address Translation (NAT)

NAT is a technique used to translate private IP addresses to public IP addresses. This allows you to protect your internal network from direct internet access.

Benefits of using NAT:

- **Security:** Protect your internal network from unauthorized access.
- **Cost savings:** Reduce the number of public IP addresses required.
- **Flexibility:** Easily manage your network topology.

VPC Peering

VPC peering allows you to connect two VPCs within the same AWS region. This enables you to securely communicate between resources in different VPCs, without exposing them to the public internet.

Benefits of using VPC peering:

- **Connectivity:** Establish secure connections between VPCs.
- **Isolation:** Maintain isolation between different environments.
- **Cost savings:** Reduce the need for VPN connections.

Basically:

- **Public IP addresses** are used to access resources from the internet.
- **Elastic IP addresses** provide a consistent public IP for your resources.
- **NAT** translates private IP addresses to public IP addresses for security and cost savings.
- **VPC peering** allows you to connect VPCs securely and privately.