AWS imp questions

AWS Cloud computing:

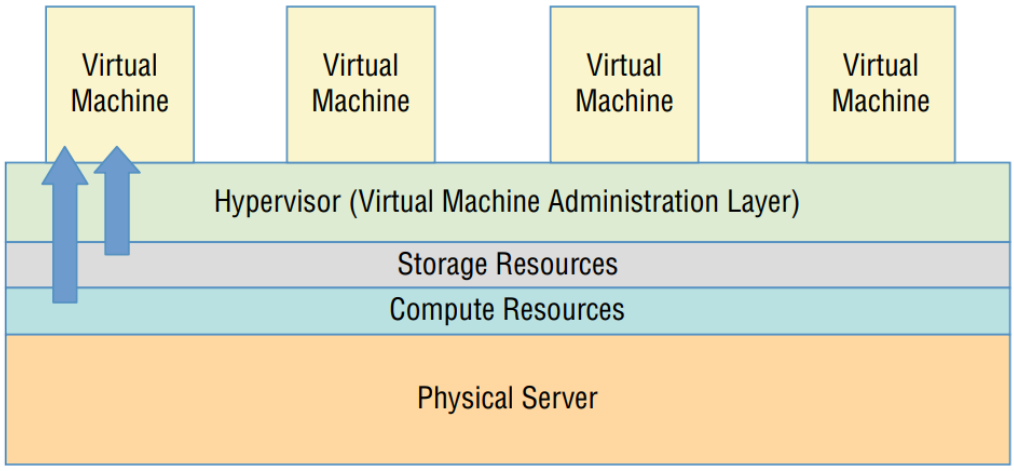
AWS Cloud Computing is a service that offers a wide range of cloud-based products, including computing power, storage, databases, and networking, all available on-demand with pay-as-you-go pricing. AWS allows businesses to replace upfront infrastructure costs with low, scalable costs based on their usage.

AWS lets businesses quickly set up and scale IT resources, such as servers, without the need for physical hardware. This enables faster deployment and flexibility, making it suitable for enterprises, startups, and public sector organizations.

Cloud Computing and Virtualization:

- Virtualization is the core of all cloud operations
- As illustrated in Figure1.1, virtualization lets you divide the hardware resources of a single physical server into smaller units.
- That physical server could therefore host multiple virtual machines (VMs) running their own complete operating systems, each with its own memory, storage, and network access.

FIGURE 1.1 A virtual machine host



- Virtualization's allows to start a virtual server in a matter of seconds, run it for exactly the time your project requires, and then shut it down
- It can be scalable depending upon the use case, and after using it can be quickly shutdown and release the resources. The resources released will become instantly

available to other workloads.
- Easy to Generate an Experimental and sandboxed environment.

The AWS Service Categories:

Compute:
Serverless architectures, auto scaling and virtual server functionalities.
Networking:
Application connectivity, access control and remote connectivity.
Storage:
Various storage platforms for immediate and long term needs.
Database:
Data solutions for use cases requiring multiple formats such as relational NoSQL or caching.
Application management:
Monitoring auditing and configuring AWS account services and running resources
Security and identity:
Authentication and authorization, data and connection encryption, integration with the third party authentication management system

---

Core AWS services (by category)

Compute:

- **EC2**: Virtual servers in the cloud that can be customized for any application needs.
- **Lambda**: Serverless service that runs code in response to events without needing always-on servers.
- **Auto Scaling**: Automatically adds or removes EC2 instances based on demand.
- **Elastic Load Balancing**: Distributes incoming traffic across multiple servers to prevent overloads.
- **Elastic Beanstalk**: Deploys and manages applications automatically, handling all infrastructure behind the scenes.

Networking:

- **VPC**: Private network in AWS to securely run your cloud resources.
- **Direct Connect**: Provides fast, private connections between your local data center and AWS.

- **Route 53**: Manages DNS, domains, and routing for your AWS resources.
- **CloudFront**: Delivers website content quickly through cached locations globally.

Storage:

- **S3**: Reliable and scalable cloud storage for data and backups.
- **S3 Glacier**: Low-cost, long-term storage for data with slower retrieval times.
- **EBS**: Persistent virtual storage drives for EC2 instances.
- **Storage Gateway**: Connects on-premise storage with AWS cloud storage.

Database:

- **RDS**: Managed SQL databases like MySQL, SQL Server, etc.
- **DynamoDB**: Fast, flexible NoSQL database for scalable applications.

Application Management:

- **CloudWatch**: Monitors your cloud resources and can trigger actions or alerts.
- **CloudFormation**: Automates AWS resource setup using template files.
- **CloudTrail**: Tracks all API activity within your AWS account for auditing.
- **Config**: Monitors changes in your AWS resources and ensures compliance.

Security:

- **IAM**: Manages access to AWS resources by setting permissions for users and services.
- **KMS**: Manages encryption keys to secure data in AWS.
- **Directory Service**: Connects AWS resources with identity services like Microsoft AD.

Application Integration:

- **SNS**: Sends notifications or messages to other services, apps, or users.
- **SWF**: Manages workflows across services and tasks, including human actions.
- **SQS**: Sends and stores messages between different parts of an application.
- **API Gateway**: Builds and manages APIs for your AWS applications.

AWS Platform Architecture:

- AWS has data centers all over the world, so you can reduce delays by running your services close to your users.

- This also helps meet legal requirements for storing data in specific locations.
- Costs and available services can differ between regions.

Cloud Computing Optimization

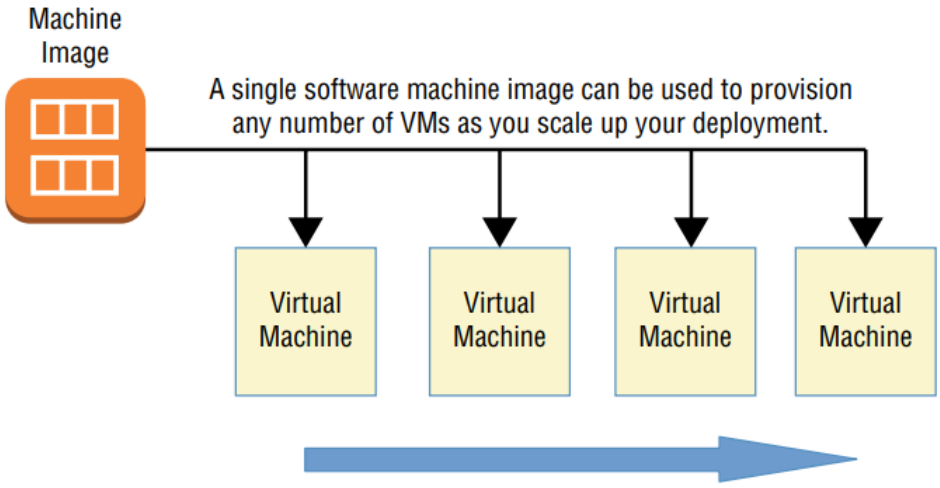
- Scalability
- Elasticity
- Cost management

Scalability:

```
it is a process of adding resources to meet the unexpected increase in demand
of the application or the server, by automatically adding resources.
```

AWS offers its **Autoscaling service** through which you define a machine image that can be instantly and automatically replicated and launched into multiple instances to meet demand.

FIGURE 1.2 Copies of a machine image are added to new VMs as they're launched.



Elasticity:

```
- Automatically reduces capacity when demand drops
- Helps control cost by running resources only
```

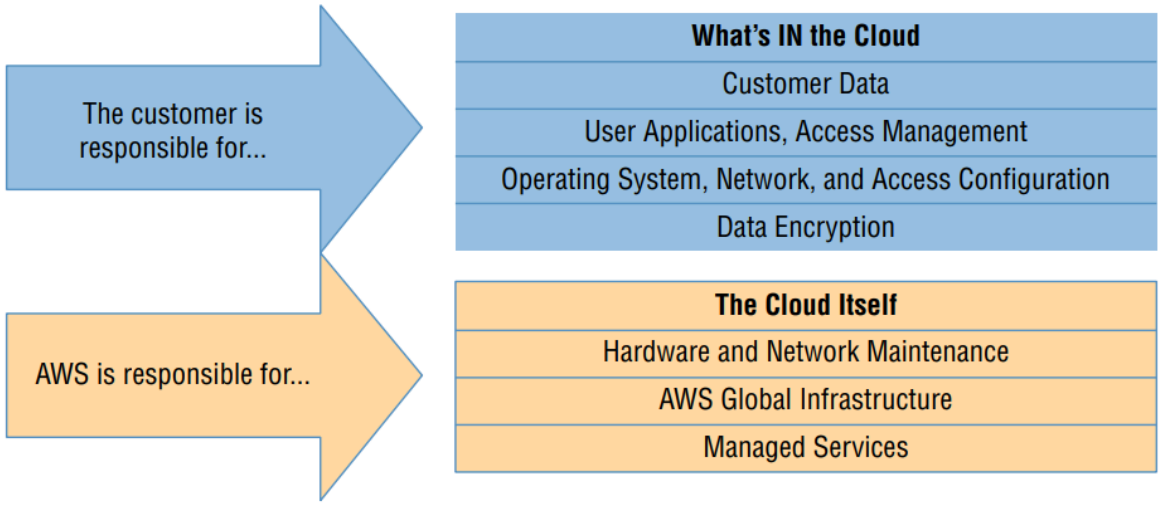
Cost Management:

- Transition from Capex to Opex (shifts IT spending from capital expenditure to operational expenditure)
- AWS provides a **Total Cost of Ownership** (TCO) to calculate for cost comparison.
- Reduces upfront cost of hardware and associated expenses

The AWS Shared Responsibility Model (!important)

- AWS guarantees the secure and uninterrupted operation of its “cloud.”- (servers, storage, networks, and managed services).
- AWS customers, as illustrated in Figure1.3, are responsible for whatever happens within that cloud.

FIGURE 1.3 The AWS Shared Responsibility Model



The AWS Service Level Agreement:

- By “guarantee,” AWS doesn’t mean that service disruptions or security breaches will never occur.
- Drives may stop spinning, major electricity systems may fail, and natural disasters may happen.
- But when something does go wrong, AWS will provide service credits to reimburse customers for their direct losses whenever uptimes fall below a defined threshold.

--main point

- The service level agreement (SLA) rate for AWS EC2, for instance, is set to 99.99 percent—meaning that you can expect your EC2 instances, ECS containers, and EBS storage devices to be available for all but around four minutes of each month.

The AWS CLI

- Using the AWS CLI can save you time by letting you automate tasks, instead of manually clicking through the console every time.
- You can create a simple script to run from your terminal or PowerShell.
- Setting up the AWS CLI on Linux, Windows, or macOS is easy, but steps may vary by platform.

EC2 Instances

- An **EC2 instance** is a virtual server that behaves like a real one, with storage, memory, and a network interface.
- It comes with a clean operating system, and you can choose the hardware, OS, and software stack, determining the cost based on your selections.

EC2 Amazon Machine Images

An **AMI** (Amazon Machine Image) is a template that tells EC2 what operating system and software to use when launching an instance.

Types of AMIs:

1. **Amazon Quick Start AMIs**: Pre-configured, official, and up-to-date Linux or Windows images for common operations.
2. **AWS Marketplace AMIs**: Production-ready images provided by industry vendors like SAP or Cisco.
3. **Community AMIs**: Over 100,000 AMIs created by independent vendors for specific use cases.
4. **Private AMIs**: Custom images you create for scaling or sharing, based on your own instance deployments.

Steps to Launch an EC2 Instance on AWS

1. **Access AWS Services**: Go to **Resources** and click on **Instances running** to check if any EC2 instances are active.
2. **Launch Instance**: Click on **Launch Instance** to go to the launch page.
3. **Name the Instance**: Create and enter a name for your instance.
4. **Select AMI**: Choose the required operating system from the available Amazon Machine Images (AMIs). For this example, select a **Windows AMI**.
5. **Storage Settings**: By default, a free tier eligible storage will be selected. Ensure you choose a storage option eligible for the free tier (up to 30 GB of EBS Storage).
6. **Select Instance Type**: The default instance type is **t2.micro**, which is free tier eligible. Do not select a different type to avoid extra charges.
7. **Create Key-Pair**:
    - Click on **Create new key pair**.
    - Enter a name and select **.pem**, then create. The key-pair will automatically download.
8. **Select Key-Pair**: Choose the created key-pair.
9. **Network Settings**: Keep the default network settings, adjusting if necessary.
10. **Launch Instance**: Review all selections to ensure they are free tier eligible, then click on **Launch Instance**.

Your EC2 instance will now be created!

Auto Scaling

Simple Auto Scaling in AWS *--ignore--*

Here is an explanation of Auto Scaling in AWS:

- **Auto Scaling** is a service that helps you **automatically adjust the number of EC2 instances** running in your application based on demand. This ensures your application can handle traffic spikes without crashing and scales down when demand is low to save money.
- It's like having an **automatic oven temperature controller**. The controller increases the heat when the oven temperature is too low and decreases it when it's too high. This ensures your cake is baked at the correct temperature without burning.
- Auto Scaling is essential for applications that experience fluctuating demand, such as e-commerce websites, online games, and streaming services.
- Here's how it works:
    - You create an **Auto Scaling group**, which defines the **minimum, maximum, and desired number of EC2 instances** you want running.

    - Auto Scaling monitors your application's performance using **metrics like CPU utilization, request count, and network traffic**.
    - When demand increases, Auto Scaling **automatically launches new EC2 instances** to handle the load.
    - When demand decreases, Auto Scaling **terminates unused instances** to save money.
- Auto Scaling also provides **health checks** to ensure that only healthy instances are handling traffic. If an instance becomes unhealthy, Auto Scaling will terminate it and launch a replacement.

In summary, Auto Scaling helps you maintain the availability and performance of your applications while optimizing costs by automatically adjusting the number of EC2 instances based on demand.

Types of Auto Scaling in AWS

1. **Horizontal Scaling**
    - **Definition**: Involves adding more instances to your application to handle increased demand.
    - **Implementation**:
        - **Manual Scaling**: Launch additional instances as needed.
        - **Automatic Scaling**: Utilize **Amazon EC2 Auto Scaling**, which monitors application workload and adjusts the number of instances based on predefined rules.
2. **Vertical Scaling**
    - **Definition**: Involves increasing the resources of existing instances, such as CPU, memory, or storage.
    - **Implementation**:
        - **Manual Scaling**: Resize instances to allocate more resources.
        - **Automatic Scaling**: Use **Amazon EC2 Auto Scaling** with launch configurations that specify instance sizes according to workload demands.
3. **Load Balancing**
    - **Definition**: Distributes incoming traffic across multiple instances to enhance performance and availability.
    - **Service**: **Amazon Elastic Load Balancing (ELB)** automatically distributes incoming traffic across multiple instances within one or more Availability Zones (AZs).
4. **Multi-Availability Zone Deployment**
    - **Definition**: Launching instances across multiple Availability Zones to improve availability and fault tolerance.

- **Implementation**:
    - Use **Amazon EC2 Auto Scaling** to automatically launch instances in additional AZs to maintain availability during an AZ outage.

5. **Containerization**
    - **Definition**: Involves using containers to package and deploy applications, enhancing portability and manageability.
    - **Service**: **Amazon Elastic Container Service (ECS)** allows for easy management of Docker containers on a cluster of EC2 instances, facilitating their running, stopping, and management.

Virtual Private Cloud (VPC)

Definition:
A Virtual Private Cloud (VPC) is a service that allows you to connect your on-premises resources to AWS infrastructure through a secure virtual private network. It resembles a traditional data center network but leverages AWS's scalable infrastructure.

Key Features:

- **Isolation**: VPC provides a logically isolated section of the AWS Cloud where you can launch resources.
- **Control**: You have complete control over your virtual networking environment, including:
    - **IP Address Range**: Select your own IP address range.
    - **Subnets**: Create subnets to organize your resources.
    - **Route Tables**: Configure routing for network traffic.
    - **Network Gateways**: Set up gateways for external connectivity.

Security:
You can implement multiple layers of security, including:

- **Security Groups**: Act as a virtual firewall to control traffic to your instances.
- **Network Access Control Lists (ACLs)**: Additional security layer to manage access to subnets.

Steps to Create a VPC

To create a VPC in AWS, you can use either the AWS Management Console or the AWS Command Line Interface (CLI). Here's a general outline of the steps involved using the AWS CLI:

1. **Create the VPC:**
   - Specify the **Classless Inter-Domain Routing (CIDR) block** for the VPC. The CIDR block is a range of IP addresses that defines the address space for your VPC. You must select an RFC 1918 CIDR block to avoid conflicts with public Internet addresses.
   - Execute the following command:

     ```
     aws ec2 create-vpc --cidr-block 172.16.0.0/16
     ```

   - Replace `172.16.0.0/16` with your desired CIDR block.
2. **Create Subnets:**
   - Divide the VPC's CIDR block into smaller CIDR blocks for each subnet. Subnets isolate instances from each other and control how traffic flows to and from your instances.
   - Specify the **VPC ID**, **CIDR block**, and **availability zone** for each subnet.
   - Execute a command like this for each subnet:

     ```
     aws ec2 create-subnet --vpc-id [vpc-id] --cidr-block 172.16.100.0/24
     --availability-zone us-east-1a
     ```

     - Replace `[vpc-id]`, `172.16.100.0/24`, and `us-east-1a` with your values.
3. **Create an Internet Gateway (Optional):**
   - An Internet gateway allows instances in your VPC to communicate with the Internet.
   - Execute this command:

     ```
     aws ec2 create-internet-gateway
     ```

4. **Attach the Internet Gateway to the VPC (Optional):**
   - If you created an Internet gateway, attach it to your VPC using the following command:

     ```
     aws ec2 attach-internet-gateway --internet-gateway-id [internet-
     gateway-id] --vpc-id [vpc-id]
     ```

     - Replace `[internet-gateway-id]` and `[vpc-id]` with the appropriate values.
5. **Create Route Tables (Optional):**
   - Route tables control how network traffic is routed within your VPC. Each subnet must

be associated with a route table.
    - You can use the default route table or create a custom one.
6. **Configure Routes (Optional):**
    - Add routes to your route tables to direct traffic to different destinations, such as the Internet or other VPCs.
7. **Create Security Groups:**
    - Security groups act as firewalls that control traffic to and from your instances. Define inbound and outbound rules to specify what traffic is allowed.
8. **Launch EC2 Instances:**
    - Launch EC2 instances into your subnets, specifying the desired instance type, operating system, and other configurations. Make sure to associate your security groups with the instances.

By following these steps, you can create a custom VPC to host your AWS resources in a secure and isolated environment that meets your specific needs.

Working of a Virtual Private Cloud (VPC)

A **Virtual Private Cloud (VPC)** is a secure, isolated virtual network in the AWS cloud where you can launch resources like EC2 instances. It closely resembles a traditional data center network but utilizes AWS's scalable infrastructure. Here's how a VPC works:

1. **Creating the VPC**:
    - Start by creating a VPC and defining its boundaries using a **CIDR block** (a range of IP addresses). This block determines the IP addresses that can be assigned to resources within the VPC.
    - **Analogy**: Think of this as defining the address of your virtual house.
2. **Subnets**:
    - Divide your VPC into **subnets**, each with its own smaller CIDR block.
    - **Public Subnets**: Have internet connectivity through an **Internet Gateway** (IGW).
    - **Private Subnets**: Are isolated from the internet.
    - **Analogy**: Subnets are like rooms in your house, allowing you to group resources logically and control communication.
3. **Routing**:
    - **Route Tables**: Determine how traffic flows within the VPC and to the internet. Define routes to specify which traffic goes where (to the internet, within the VPC, or to other networks).

- **Analogy**: Think of route tables as signposts directing traffic to the right rooms or outside.

4. **Connecting to the Internet**:
   - To enable instances in public subnets to access the internet, an **Internet Gateway (IGW)** is required. This acts as a bridge between your VPC and the internet.
5. **Security**:
   - **Security Groups**: Function like firewalls that control traffic to and from individual instances.
   - **Network Access Control Lists (NACLs)**: Control traffic at the subnet level, allowing you to define rules based on IP addresses, ports, and protocols.
   - **Analogy**: These are security guards ensuring that only authorized traffic enters or leaves your virtual house.
6. **Launching EC2 Instances**:
   - After setting up your VPC, you can launch EC2 instances into your subnets, choosing the instance type, operating system, and configurations as needed.

Primary IP Addresses and Routing Tables

- A **primary private IP address** is automatically assigned to the primary Elastic Network Interface (ENI) of an EC2 instance, based on the subnet's IP range. This address is permanent and cannot be changed or removed.
- **Elastic Network Interfaces (ENIs)** allow instances to communicate with other resources in the network.
- A **routing table** is used to direct network traffic within a VPC. Each subnet is linked to a routing table, which contains rules about where to send traffic.
  - **Main Route Table**: Automatically created with the VPC. It is associated with all subnets by default.
  - **Custom Route Tables**: You can create custom tables for more control, associating subnets as needed.
  - Every route table has a **local route** to allow communication between instances within the same VPC.
- To provide internet access:
  - You must add a **default route** to the internet gateway (IGW).
  - A subnet with a route to the IGW is a **public subnet**.
  - A subnet without an IGW route is a **private subnet**.

This setup ensures organized traffic flow and security within the VPC.

Amazon S3

Definition: Amazon Simple Storage Service (S3) provides flexible, reliable, and low-cost object storage. It is often used for data backups, file storage, and logs.

Key Features:

- **Buckets:** Files are stored in buckets, which must have globally unique names. By default, you can create up to 100 buckets per account.
- **Durability:** S3 offers 99.999999999% durability by replicating data across at least three availability zones, ensuring minimal data loss risk.
- **Availability:** S3 provides 99.99% availability annually.
- **Object Lifecycle:** Automate data retention using versioning and lifecycle policies. Objects can be transitioned between storage classes based on age.
- **Encryption:** Data can be encrypted both at rest and in transit.
- **Access Control:** Default access is restricted to your account. Access can be managed through ACLs, bucket policies, or IAM policies.
- **S3 Select:** Allows you to run SQL-like queries on your stored data for efficient retrieval.
- **Static Website Hosting:** S3 can host static websites, with DNS routing through Amazon Route 53.
- **Storage Classes:** S3 offers various storage options like S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, and S3 One Zone-IA.

Amazon S3 Glacier

Definition: Amazon S3 Glacier is designed for long-term, low-cost storage for archiving and backups. It is best suited for storing infrequently accessed data at a low cost.

Key Features:

- **Long-Term Storage:** Optimized for storing archival data that is rarely accessed.
- **Low Cost:** Provides much lower storage costs compared to S3.
- **Data Retrieval Time:** Retrieval times can vary from hours to days, depending on the storage tier.
- **Storage Tiers:** Includes Glacier Standard (retrieval within 24 hours) and Glacier Deep Archive (retrieval within 12-48 hours).

- **Vaults and Archives:** Data is stored in vaults, and individual objects (archives) can be up to 40 TB in size.
- **Glacier Select:** Enables querying data stored in Glacier using SQL queries.

Integration with S3: S3 lifecycle rules can transition objects to Glacier for long-term archiving based on predefined conditions.

Inbound and Outbound Rules in AWS Networking

Inbound rules allow traffic into a network resource (like an EC2 instance), while **outbound rules** control the traffic that can leave a resource. These rules act as firewalls, controlling the flow of data in and out of AWS network resources.

Security Groups

- **Purpose**: Security groups act as virtual firewalls for EC2 instances, controlling both inbound and outbound traffic. Every Elastic Network Interface (ENI) must have at least one associated security group.
- **Default-Deny**: Security groups deny all traffic unless explicitly allowed.
- **Stateful**: If traffic is allowed in one direction (e.g., inbound), return traffic in the opposite direction (outbound) is automatically allowed without needing additional rules.
- **Rules**: You define rules specifying allowed traffic (protocols, ports, and IP ranges). Rules are processed in order of priority.

Network Access Control Lists (NACLs)

- **Purpose**: NACLs control traffic at the subnet level, acting as a firewall for entire subnets within a VPC. Each subnet can only be associated with one NACL.
- **Stateless**: Unlike security groups, NACLs are stateless. Traffic allowed in one direction (e.g., inbound) must have an explicit outbound rule to allow return traffic.
- **Default Behavior**: By default, NACLs allow all inbound and outbound traffic unless configured otherwise.
- **Rules**: NACL rules are processed in order of priority. Like security groups, they use a default-deny approach if no rules apply.

Key Differences: Security Groups vs. NACLs

Feature	Security Group	NACL
Scope	Instance	Subnet
Statefulness	Stateful	Stateless
Default Behavior	Deny all inbound, allow all outbound	Allow all inbound and outbound by default
Rule Order	Rules processed as a set	Rules processed in order of priority

Best Practices

- **Layered Security**: Use both security groups and NACLs for layered security. Security groups give instance-level control, while NACLs control traffic at the subnet level.
- **Efficiency**: If you only need instance-level control, security groups are often more efficient. NACLs may require more complex configuration due to their stateless nature.
- **Matching Rules**: In NACLs, make sure outbound rules correspond with inbound rules (e.g., if allowing inbound HTTPS on port 443, allow outbound on port 443 too).

Example Use Cases

- **Security Groups**: Allow web traffic to EC2 instances by configuring security groups to allow HTTP/HTTPS traffic (ports 80 and 443).
- **NACLs**: Protect a database subnet by limiting access to specific IP ranges or security groups.

Understanding inbound and outbound rules and the differences between security groups and NACLs helps create a more secure AWS architecture.