**AWS Shared Responsibility Model**

The **AWS Shared Responsibility Model** defines the security and operational responsibilities between AWS and the customer.

**1. AWS Responsibilities ("Security of the Cloud")**

AWS is responsible for securing the infrastructure that runs AWS services.

* Physical security of data centers (e.g., access control, surveillance)
* Network security (e.g., DDoS protection, VPC isolation)
* Hardware and virtualization layer security
* Compliance with industry standards (ISO, SOC, PCI, etc.)

**2. Customer Responsibilities ("Security in the Cloud")**

Customers are responsible for configuring security **inside** their AWS accounts.

* Managing IAM users, groups, and roles
* Securing applications and data (encryption, backups)
* Network configurations (Security Groups, NACLs)
* OS and software updates (for EC2 instances and self-managed apps)
* Compliance with regulatory and internal security policies

**AWS Identity and Access Management (IAM)**

AWS IAM enables secure access management for AWS services and resources.

**1. IAM Users**

* Represents an **individual** (e.g., an employee, developer, administrator)
* Can have **login credentials** (username and password) for the AWS Management Console
* Can be assigned **Access Keys** for programmatic access via CLI/API
* Permissions are controlled via IAM **policies**

**2. IAM Groups**

* A **collection of IAM users** with similar permissions
* Users in a group **inherit** the permissions assigned to the group
* Helps simplify access management (e.g., "Developers", "Admins", "Finance")

**3. IAM Roles**

* IAM roles allow AWS services or applications to assume permissions temporarily
* Assigned to AWS resources like **EC2 instances, Lambda functions, and ECS tasks**
* No long-term credentials; instead, they use **temporary security tokens**
* Can be assumed by **other AWS accounts, services, or federated users**

**4. IAM Account (Root User)**

* The **AWS root account** is created when signing up for AWS
* Has **full administrative access** to all AWS services
* Best practice: **Avoid using the root user for daily operations**
* Secure the root user with **MFA (Multi-Factor Authentication)**

**AWS Virtual Private Cloud (VPC)**

AWS **VPC (Virtual Private Cloud)** enables users to launch AWS resources in an isolated network environment.

**1. Key Components of a VPC**

* **VPC**: A logically isolated network in AWS
* **Subnets**: Divide a VPC into smaller sections (Public & Private)
* **Route Tables**: Control traffic routing between subnets and the internet
* **Internet Gateway (IGW)**: Allows internet access for public subnets
* **NAT Gateway**: Enables private subnets to access the internet securely
* **Security Groups (SGs)**: Acts as a firewall at the **instance level**
* **Network ACLs (NACLs)**: Acts as a firewall at the **subnet level**

**2. Public vs. Private Subnets**

| **Subnet Type** | **Internet Connectivity** | **Use Case** |
| --- | --- | --- |
| Public Subnet | Has direct internet access via IGW | Web servers, Bastion hosts |
| Private Subnet | No direct internet access (uses NAT Gateway) | Databases, backend applications |

**3. Security in VPC**

* Use **Security Groups** to control inbound/outbound traffic at the instance level
* Use **NACLs (Network ACLs)** for subnet-level security rules
* Implement **VPC Peering** or **Transit Gateway** for secure VPC-to-VPC communication
* Use **AWS VPN or Direct Connect** for hybrid cloud networking

**Best Practices for IAM and VPC**

✅ **Use IAM roles instead of hardcoded credentials** for AWS services

✅ **Apply the Principle of Least Privilege (PoLP)** – Grant minimal permissions

✅ **Enable MFA** for root users and critical IAM users

✅ **Regularly rotate access keys and credentials**

✅ **Use Private Subnets** for sensitive resources like databases

✅ **Enable VPC Flow Logs** for monitoring network traffic