AWS Fundamental Concepts

I. Amazon Elastic Compute Cloud (EC2) and Security

A. EC2 Instance and AMI

An **EC2 Instance** is a virtual machine running on AWS infrastructure. It is defined by its Instance Type (like `t2.micro` in our guides, which determines CPU, memory, and networking capacity) and an **Amazon Machine Image (AMI)**.

- **AMI (Amazon Machine Image):** Think of the AMI as a blueprint or template. It is a snapshot of the operating system, the application server, and any application data we need. When we launch an instance, we use an AMI. We can also **create an AMI** from a running instance to capture its current state for backup or deployment purposes.

B. Key Pairs and Secure Access

A **Key Pair** is fundamentally a security mechanism used to prove our identity when we connect to a Linux EC2 instance using SSH (Secure Shell).

- **How it Works:** It uses **public-key cryptography**. When we create the key pair, AWS stores the **Public Key** on the instance, and we download and keep the corresponding **Private Key** (often a `.pem` file) locally. When connecting, the instance challenges us; if our private key matches the public key, access is granted.
- **Key Pair Recovery Principle:** The sources demonstrate that if we lose the private key file (`temp-key.pem`), **we cannot recover it**. Instead, we use the AMI method: we create a new image of the instance's hard drive, launch a *new* instance from that image, and associate it with a *new*, functional key pair (`recovered-key.pem`). This ensures data persistence but establishes a new secure access path.

C. Security Groups (SGs)

A **Security Group** acts as a **stateful virtual firewall** for your EC2 instances.

- **Purpose:** It controls inbound and outbound traffic at the instance level.
- **Rules:** Rules are mandatory and specific. For example, to allow SSH access, we must explicitly create an **Inbound Rule** authorizing the **TCP Protocol** on **Port 22** from a specific source (often `0.0.0.0/0` , meaning the entire internet).
- **Stateful Nature:** If you allow traffic *in* (inbound), the response traffic is automatically allowed *out* (outbound), and vice versa, without needing a separate outbound rule.

D. User Data and Bootstrapping

User Data is a powerful feature that allows us to inject scripts (usually shell scripts for Linux) into an EC2 instance **that run only once, upon first launch**.

- **Function:** It is used for **bootstrapping**. This means automatically performing initial setup tasks, such as installing necessary software (like `httpd`), starting services, and configuring initial content

(`/var/www/html/index.html`). This ensures instances launched by an Auto Scaling Group or a Load Balancer are ready to serve traffic immediately.

II. Amazon VPC: The Isolated Network (Problems 2 & 3)

The **Virtual Private Cloud (VPC)** defines your isolated, private network environment within the AWS cloud. Everything you build resides within a VPC.

A. VPC and CIDR Blocks

A VPC is defined by an **IPv4 CIDR block** (Classless Inter-Domain Routing), such as `10.0.0.0/16`.

- **Isolation:** The VPC is logically isolated from all other VPCs, including those belonging to other AWS customers.
- **CIDR Block:** The CIDR determines the range of private IP addresses available in your network. A `/16` block allows for 65,536 addresses.

B. Subnets and Availability Zones (AZs)

A **Subnet** is a segmented range of addresses *within* the VPC's overall CIDR block (e.g., `10.0.1.0/24`).

- **Mandatory AZ Association:** Crucially, every subnet **must be entirely contained within a single Availability Zone (AZ)**. AZs are physically distinct data centers in a region. By placing subnets across multiple AZs (like `us-east-1a` and `us-east-1b`), we achieve high availability and fault tolerance.

C. Internet Gateway (IGW)

The **Internet Gateway (IGW)** is a horizontally scaled, redundant VPC component that permits communication between instances in your VPC and the internet.

- **Function:** It must be **attached to the VPC**. It serves as the gateway for public traffic.

D. Route Tables (RTs)

A **Route Table** dictates where network traffic from a subnet is directed.

- **Main Route (VPC Local):** Every RT automatically contains a route for the VPC's CIDR block (e.g., `10.0.0.0/16`), targeting `local`.
- **Public Route Table:** To make a subnet "public" (internet accessible), the associated route table must have a default route (Destination: `0.0.0.0/0`, meaning all external traffic) pointing to the **Internet Gateway**.
- **Private Route Table:** To ensure a subnet is isolated, the associated route table **must NOT have a route pointing to the IGW**. This enforces security and isolation for database servers or backend application layers.

III. Identity and Access Management (IAM) (Problems 4 & 5)

IAM controls who can access your resources and what actions they can perform (authentication and authorization).

A. IAM Entities (Users, Groups, Roles)

- **IAM User (`example-user`):** Represents an individual person or application that needs to interact with AWS. Users can be assigned policies directly.
- **IAM Group (`example-group`):** A collection of IAM users. We attach policies to the group, and all users in that group inherit those permissions. *This is best practice for standard permissions.*
- **IAM Role (`ec2-s3-role`):** An identity meant to be *assumed*. Roles are critical for **delegating permissions** securely to AWS services (like an EC2 instance needing to talk to S3) or external identities. They require a **Trust Policy** which specifies which entity (e.g., `ec2.amazonaws.com` service) is allowed to assume the role.

B. IAM Policies (Managed, Custom, Inline)

Policies are the JSON documents that define permissions. They operate on the structure of `Effect` (Allow/Deny), `Action` (what they can do, e.g., `ec2:Describe*`), and `Resource` (on which resource, often `*` for global).

- **AWS Managed Policies:** Pre-built policies managed by AWS (e.g., `AmazonS3ReadOnlyAccess`).
- **Custom Policies:** User-created policies defined by JSON. These are reusable and attachable to multiple entities.
- **Inline Policies:** Policies that are **embedded directly** within a user, group, or role. They are unique to that entity and are deleted if the entity is deleted.

C. MFA

Multi-Factor Authentication (MFA) is a security layer that requires the user to provide their password plus a time-based code from a device (like a phone app). This significantly increases security by ensuring that even if a password is stolen, the account remains protected.

IV. Amazon S3 Storage (Problems 6 & 9)

Amazon S3 (Simple Storage Service) is an object storage service, often called "storage for the internet".

A. Buckets and Uploads

- **Bucket:** The fundamental container for storing objects (files). Bucket names must be globally unique.
- **Multipart Upload:** This is an efficiency feature for large files. Instead of uploading one massive chunk, S3 splits the file into smaller pieces, uploads them simultaneously (in parallel), and reassembles them at the end. The AWS CLI and Console usually automatically trigger this for large files.

B. Versioning

Versioning enables the storage of multiple versions of an object in the same bucket.

- **Benefit:** If an object is accidentally deleted or overwritten, previous versions remain recoverable. When an object is "deleted" while versioning is on, a **delete marker** is simply created, but the object contents remain stored.

C. Default Encryption

Default Encryption ensures that all new objects uploaded to a bucket are automatically encrypted when stored at rest.

- **SSE-S3 (Server-Side Encryption with S3-Managed Keys):** This common method (referenced in the guides as `AES256`) means S3 manages the encryption keys, offering strong security with minimal configuration effort.

V. Load Balancing and Auto Scaling (Problems 7, 8, 10, 11)

These services are crucial for building highly available, resilient, and cost-effective applications.

A. Application Load Balancer (ALB)

The **ALB** is a device that automatically distributes incoming application traffic (HTTP/HTTPS) across multiple targets (like EC2 instances).

- **Layer 7:** It operates at Layer 7 (the Application layer), meaning it understands web protocols.
- **Network Mapping:** ALBs must be deployed across **at least two subnets in different AZs** to ensure that if one AZ fails, the load balancer remains available.
- **Target Group:** A logical container that holds the destination EC2 instances (`web-server-1`, `web-server-2`). The Target Group is responsible for performing **health checks** to confirm if an instance is capable of serving traffic.
- **Listener:** A process that checks for connection requests on a defined protocol and port (e.g., HTTP 80) and forwards them according to the rules (e.g., to the `web-tg`).

B. Auto Scaling Group (ASG)

The **ASG** is a service that ensures you have the **correct number of EC2 instances** running to handle your application's load.

- **Capacity:** It manages the fleet size based on three numbers: **Minimum** (Min) size (guaranteed lowest number), **Maximum** (Max) size (cost control limit), and **Desired** capacity (the target running number).
- **Launch Template (LT):** The ASG's fundamental instruction set. The LT specifies the AMI, instance type, security groups, and User Data necessary to launch a *new, identical* instance automatically.
- **Scaling Policy:** Defines the rules for expanding or contracting the fleet. The guides use **Simple Scaling** based on metrics (e.g., if CPU utilization exceeds 70%, add one instance; if it falls below 40%, remove one instance).
- **Self-Healing:** A core feature of ASG. If an instance fails or is manually terminated, the ASG detects that the `Desired Capacity` is no longer met and automatically launches a replacement instance using the Launch Template.

VI. Configuration Methodology (CLI vs. Console)

The operational guides teach that there are two primary ways to interact with AWS.

- **AWS Management Console (GUI):** This is the web interface. It's ideal for visual understanding, learning, and immediate tasks, relying on navigation, forms, and clicking.
- **AWS Command Line Interface (CLI):** This is a text-based interface where we execute specific `aws` commands. It is mandatory for **automation, scripting, and repeatability**. Complex configurations (like IAM policies or S3 encryption) often require providing parameters as structured

JSON files (`file://policy.json`). For launch templates, User Data must be **Base64 encoded** within the JSON structure.