

Fundamental Concepts

1. What is Amazon EC2?

- **Answer:** Amazon EC2 (Elastic Compute Cloud) is a core AWS service that provides scalable computing capacity in the cloud. It allows users to rent virtual servers, called "instances," on which they can run their own applications. EC2 eliminates the need to invest in physical hardware upfront, allowing you to develop and deploy applications faster.

2. What are the main benefits of using EC2?

- **Answer:**
 - **Elasticity/Scalability:** You can easily increase or decrease the number of instances (scale out/in) or change the instance size (scale up/down) based on demand.
 - **Cost-Effectiveness:** You pay only for the compute time you consume (or reserve) with various pricing models (On-Demand, Reserved Instances, Spot Instances).
 - **Control:** You have root access (Linux) or administrator access (Windows) to your instances, allowing full control over the operating system and applications.
 - **Flexibility:** Offers a wide variety of instance types optimized for different workloads (compute, memory, storage, GPU), various operating systems, and software packages.
 - **Integration:** Tightly integrated with other AWS services like S3, RDS, VPC, ELB, etc.
 - **Reliability:** Instances can be launched in multiple Availability Zones (AZs) within a Region for high availability.

3. What is an AMI (Amazon Machine Image)?

- **Answer:** An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your EC2 instance. You must specify an AMI when you launch an instance. AWS provides many public AMIs (Linux, Windows), you can create your own custom AMIs from existing instances, or use AMIs from the AWS Marketplace. Using AMIs ensures consistency and speeds up the launch process.

4. What are EC2 Instance Types? How do you choose one?

- **Answer:** Instance types are different combinations of CPU, memory, storage (type and size), and networking capacity optimized for various use cases. Examples include:
 - t-series (e.g., t3.micro, t4g.small): Burstable general-purpose, good for low-to-moderate baseline CPU usage (development, small websites).
 - m-series (e.g., m5.large): General purpose, balanced compute, memory, and network.
 - c-series (e.g., c5.xlarge): Compute-optimized, for CPU-intensive tasks.

- r-series (e.g., r5.large): Memory-optimized, for memory-intensive tasks like databases or caches.
- g-series, p-series: Accelerated computing (GPUs) for machine learning, graphics.
- You choose an instance type based on the specific requirements of your application (CPU needs, memory footprint, storage performance, network bandwidth). You often start with a general-purpose type and monitor performance to see if optimization is needed.

Instance Management & Lifecycle

1. What is the difference between stopping and terminating an EC2 instance?

- **Answer:**
 - **Stopping:** This is like shutting down a physical server. The instance performs a normal shutdown. The attached EBS root volume is preserved, along with its data. You are *not* charged for instance usage while it's stopped, but you *are* charged for the associated EBS volume storage. You can restart a stopped instance later, and it retains its private IP, (usually) its public IP if it's an Elastic IP, and its EBS volumes.
 - **Terminating:** This is like permanently deleting the server. The instance performs a shutdown, and by default, the attached EBS *root* volume is deleted (this can be changed via a setting). Any data on instance store volumes is always lost. The instance cannot be recovered. You stop incurring charges for the instance itself once terminated (though charges for snapshots or other resources might persist).

2. Can you briefly describe the steps to launch an EC2 instance using the AWS Management Console?

- **Answer:**
 1. Navigate to the EC2 Dashboard in the AWS Console.
 2. Click "Launch Instance".
 3. **Choose an AMI:** Select an operating system template (e.g., Amazon Linux 2, Ubuntu, Windows Server).
 4. **Choose an Instance Type:** Select the hardware configuration (CPU, RAM, etc., e.g., t2.micro).
 5. **Configure Instance Details:** Specify the number of instances, select the VPC and subnet, configure network settings (e.g., auto-assign public IP), IAM role (optional), etc.
 6. **Add Storage:** Configure the root volume (EBS usually) size and type. Add additional EBS volumes if needed.
 7. **Add Tags:** Assign key-value tags for organization and cost tracking.
 8. **Configure Security Group:** Create a new security group or select an existing one to act as a firewall, defining allowed inbound/outbound traffic (e.g., allow SSH on port 22 from your IP).
 9. **Review and Launch:** Review the configuration.

10. **Select Key Pair:** Choose an existing key pair or create a new one. You *must* download and save the private key file (.pem) if creating a new one – this is essential for connecting to the instance (especially Linux).

11. Click "Launch Instances".

3. What is a Key Pair used for with EC2? What happens if you lose the private key?

- **Answer:** A key pair consists of a public key (stored by AWS) and a private key (which you keep). For Linux instances, the public key is placed in the ~/.ssh/authorized_keys file on the instance at launch time. You use the corresponding private key file (.pem or .ppk) with an SSH client (like ssh on Linux/macOS or PuTTY on Windows) to securely authenticate and log in to your Linux instance without needing a password. For Windows instances, you use the private key to decrypt the administrator password.
- **If you lose the private key:** You will lose the ability to directly SSH into the instance (for Linux) or retrieve the initial administrator password (for Windows) using that key pair. AWS does not keep a copy of your private key and cannot recover it. Depending on the setup, recovery might involve stopping the instance, detaching the root volume, attaching it to another instance to modify authorized_keys (Linux), or using Systems Manager if configured, but direct login via that key pair is lost. It's critical to store private keys securely.

Storage Options

1. What are the main storage options for EC2 instances?

- **Answer:** The two main storage options are:
 - **EBS (Elastic Block Store):** Network-attached storage volumes that persist independently from the life of an instance. They behave like virtual hard drives. You can attach/detach them from running instances. Data persists even if the instance is stopped or terminated (unless the "Delete on Termination" flag is set for the root volume). EBS offers different volume types (SSD, HDD) optimized for performance or cost. Good for durable, persistent storage.
 - **Instance Store (Ephemeral Storage):** Storage volumes located on disks that are physically attached to the host computer running the instance. Instance store provides temporary block-level storage. Data on instance store volumes *persists only during the life of the instance*. If the instance is stopped, hibernated, or terminated, all data on instance store volumes is lost. It offers very high I/O performance but is not durable. Good for temporary data, caches, buffers.

2. What is the key difference between EBS and Instance Store? When would you use each?

- **Answer:** The key difference is **persistence**.
 - **EBS:** Persistent. Data survives instance stops and (usually) terminations. Network-attached. Use for operating systems (root volumes), databases, application data, or any data that needs to survive the instance lifecycle.
 - **Instance Store:** Non-persistent (ephemeral). Data is lost if the instance stops, hibernates, or terminates. Physically attached (faster I/O). Use for temporary storage like caches, buffers, scratch space, or data that is replicated across

other instances (like some distributed databases or processing jobs where data loss on one node is acceptable).

Security

1. What is a Security Group in the context of EC2? How does it work?

- **Answer:** A Security Group acts as a virtual firewall *at the instance level* to control inbound and outbound traffic. You define rules that specify which protocols (TCP, UDP, ICMP), port ranges, and source/destination IP addresses (or other security groups) are allowed. Security Groups are *stateful*, meaning if you allow inbound traffic (e.g., on port 80 for HTTP), the return outbound traffic for that connection is automatically allowed, regardless of outbound rules. By default, all inbound traffic is denied, and all outbound traffic is allowed. You associate one or more security groups with an EC2 instance when you launch it.

2. How is a Security Group different from a Network ACL (NACL)? (This might come up if you also mentioned VPC).

- **Answer:**
 - **Scope:** Security Groups operate at the instance level; NACLs operate at the subnet level.
 - **State:** Security Groups are stateful; NACLs are stateless (return traffic must be explicitly allowed).
 - **Rules:** Security Groups only support 'allow' rules; NACLs support 'allow' and 'deny' rules.
 - **Application:** An instance can belong to multiple Security Groups; a subnet is associated with only one NACL.

Pricing & Cost Management

1. Can you name some EC2 pricing models?

- **Answer:**
 - **On-Demand:** Pay for compute capacity by the hour or second (depending on instance type/OS) with no long-term commitments. Most flexible but potentially highest cost.
 - **Reserved Instances (RIs):** Provide a significant discount (up to 72%) compared to On-Demand pricing in exchange for committing to a specific instance type, region, and term (1 or 3 years). Good for steady-state workloads.
 - **Savings Plans:** A flexible pricing model offering lower prices in exchange for a commitment to a consistent amount of usage (measured in \$/hour) for a 1 or 3-year term. Apply across EC2, Fargate, and Lambda. More flexible than RIs.
 - **Spot Instances:** Allow you to bid on spare EC2 computing capacity at steep discounts (up to 90%) compared to On-Demand. Suitable for fault-tolerant, flexible workloads (batch processing, data analysis) that can handle interruptions, as AWS can reclaim Spot Instances with short notice if capacity is needed.

- **Dedicated Hosts:** Physical servers dedicated for your use. Often used for compliance requirements or specific software licenses.

Scalability & Availability

1. What is the difference between an AWS Region and an Availability Zone (AZ)? How does this relate to EC2?

- **Answer:**
 - **Region:** A physical geographic location in the world where AWS has multiple data centers (e.g., us-east-1 in N. Virginia, eu-west-2 in London). Regions are isolated from each other.
 - **Availability Zone (AZ):** Consists of one or more discrete data centers within a Region, each with redundant power, networking, and connectivity. AZs are physically separate but connected with low-latency links.
 - **Relation to EC2:** You launch EC2 instances *within* a specific AZ in a selected Region. Launching instances across multiple AZs within the same Region is a common strategy to build highly available applications, as an issue affecting one AZ is unlikely to affect others simultaneously. Services like Elastic Load Balancing (ELB) can distribute traffic across instances in multiple AZs.

2. What is EC2 Auto Scaling?

- **Answer:** EC2 Auto Scaling helps ensure you have the correct number of EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups (ASGs). You can configure the ASG to maintain a minimum number of instances, automatically increase the number of instances during demand spikes (scale out), and decrease capacity during lulls (scale in) based on defined metrics (like CPU utilization) or schedules. This improves availability and manages costs.

Scenario/Troubleshooting

1. You have launched a Linux EC2 instance, but you cannot connect to it via SSH. What are some common things you would check?

- **Answer:** I would check the following:
 1. **Instance State:** Is the instance running? Check the EC2 console.
 2. **Security Group Rules:** Does the security group associated with the instance allow inbound traffic on port 22 (SSH) from my current public IP address (or 0.0.0.0/0 if less secure access is intended)?
 3. **Network ACLs (NACLs):** Although less common for blocking SSH unless customized, does the NACL associated with the instance's subnet allow inbound traffic on port 22 *and* outbound traffic on the ephemeral port range (1024-65535) for the return connection?
 4. **Public IP Address:** Am I trying to connect to the correct public IP address or DNS name of the instance? Does the instance *have* a public IP (check if it's in a public subnet with the setting enabled, or has an Elastic IP)?

5. **Route Table:** Is the instance in a public subnet with a route to the Internet Gateway in its associated route table? (Needed for my connection to reach the instance).
6. **Key Pair:** Am I using the correct private key (.pem file) that corresponds to the key pair selected when launching the instance? Are the permissions on my .pem file correct (usually `chmod 400 my-key.pem` on Linux/macOS)?
7. **Local Firewall:** Is a firewall on my local machine blocking the outbound SSH connection?
8. **Instance OS Issues:** Is the `sshd` service running on the instance? (Less likely to check first, but possible if the above are correct).

Tips for Answering:

- **Focus on Fundamentals:** Demonstrate a solid grasp of what EC2 is, its core components, and basic operations.
- **Use Correct Terminology:** Use terms like instance, AMI, EBS, Security Group, Key Pair, Region, AZ accurately.
- **Explain the "Why":** Understand the purpose behind features (e.g., why use an AMI, why choose EBS over Instance Store).
- **Structure Your Answers:** For multi-part questions (like troubleshooting), list steps logically.
- **Be Prepared for Comparisons:** Understand the key differences between related concepts (Stop vs. Terminate, EBS vs. Instance Store, SG vs. NACL).

Good luck with your interview!