EC2:-

Instance:- it's a **running virtual machine or service** that uses allocated resources from a cloud provider, such as AWS EC2, RDS, or Lambda

If another user want to access your instance what are two main things you will give to client?

1)IP address

2)Key pair

1)**keypair:- Key Pair Basics (Public and Private Keys)**

* **Public Key:**
  + This is the key that is stored on the AWS platform (specifically within the EC2 instance metadata).
  + It is shared openly, and it is used to encrypt data or verify a connection.
* **Private Key:**
  + The private key is never shared and is kept securely by the user.
  + It is used to decrypt data or to authenticate the identity of the person or system trying to access the instance.
  + The private key must be kept secure, because anyone with access to this key can potentially access the associated EC2 instance.

**2. How SSH Key Pairs Work**

**When you create an EC2 instance, you associate it with an SSH key pair. Here's how the authentication process works:**

* **During EC2 Instance Launch:**
  + When you launch an EC2 instance, you specify an SSH key pair.
  + AWS generates a public key for you, and you download the private key (a .pem file).
  + The public key is automatically stored on the EC2 instance in a special file (e.g., /home/ec2-user/.ssh/authorized\_keys for Amazon Linux). This allows the instance to recognize and authenticate anyone who presents the correct private key.
* **SSH Login Process:**
  + When you try to log in to the EC2 instance, you use an SSH client (like OpenSSH or PuTTY).
  + The SSH client uses the private key to sign a message (prove ownership of the private key).
  + The instance uses the public key stored in its authorized\_keys file to verify the message.
  + If the verification is successful, access is granted to the instance.

**This method is extremely secure because:**

* + The private key never leaves your computer.
  + Only the corresponding public key is stored on the instance, making it impossible for an attacker to deduce the private key from the public key alone.

**3. Why AWS Stores the Public Key and You Store the Private Key**

* **Public Key on AWS:**
  + The public key is stored on AWS (within your EC2 instance), but anyone can access it. The public key doesn’t need to be kept secret because it’s used for verification or encryption rather than for decryption.
  + The public key serves as a key reference for the EC2 instance to identify and authenticate the user trying to connect. Since it can’t be used to access the instance without the corresponding private key, it doesn’t pose a security risk.
* **Private Key with the User:**
  + The private key is your secret key, and it must be kept safe. This is the key that allows you to authenticate and establish a secure SSH connection to your EC2 instance.
  + The private key is stored on your local machine, and only you should have access to it.
  + If an attacker gains access to your private key, they could potentially access any EC2 instance that is associated with that key pair. Therefore, it is crucial to protect the private key (such as storing it in a secure location with proper permissions, using encrypted storage, etc.).

**4. Why is This Method Secure?**

* **Asymmetric Encryption:**
  + The SSH key pair works based on asymmetric encryption, where:
    - The public key is used to encrypt messages.
    - The private key is used to decrypt them.
  + Even if an attacker gains access to the public key, they cannot decrypt the messages or log in to the instance without the private key.
* **No Password Transmission:**
  + Unlike traditional password-based authentication, where the password is transmitted (which could be intercepted or guessed), SSH keys provide a more secure method of login, as the authentication occurs through a secure challenge-response mechanism.
* **Harder to Guess:**
  + **A private key is typically a long, random sequence of characters,** making it extremely hard to guess. This is in contrast to passwords, 6. What Happens if You Lose Your Private Key?
* If you lose the private key for your EC2 instance, you will not be able to access the instance through SSH unless you have an alternative access method.
* Options to regain access:
  + Create a new key pair and then replace the old key on the EC2 instance. This can be done by attaching the volume to another instance and modifying the authorized\_keys file.
  + Use EC2 Instance Connect (for Amazon Linux 2 and Ubuntu) to connect via the console if it's enabled.
* **What cases we attach multiple security groups?**

Suppose you have an EC2 instance running a web server and a database. You might apply two security groups:

* **Security Group 1**: For web access, allowing inbound traffic on ports 80 (HTTP) and 443 (HTTPS) from anywhere (0.0.0.0/0).
* **Security Group 2**: For database access, allowing inbound traffic on port 3306 (MySQL) only from internal network IPs or a specific application server security group

**2)IP Address:** An IP address is essential to identify, secure, route, and manage access to your resources (like EC2 instances) in AWS, helping to maintain connectivity and control within the network.

Ex:-  Launch an EC2 instance.

 You are assigned a **Public IP** (e.g., 203.0.113.12) to access it over the internet and a **Private IP** (e.g., 10.0.1.5) for internal communication.

 You set up a **Security Group** to allow SSH access to the instance only from a specific IP range (e.g., your office IP 192.168.1.0/24).

 You can attach an **Elastic IP** to ensure the instance retains the same public IP even if you stop and restart it.

 When setting up a **Load Balancer**, the public IP of the Load Balancer is used to distribute traffic to backend instances with private IPs.

**Launch template:-**It is used to give configuration for ec2 instances like key pairs,security group,AMI,subnet,ebs storage,instance type etc

With this it atomatically scale the instances.

How to create launch templete

1)templete name

2)templete tag(Optional)(it will give name to the launch templete so if we have many then we can find with this name)

3)Source Template(Optional)(If we want to attach any previous data of instance to the newly created launch templete then we can do with the help of this).

4)AMI selection with quick start thenc rate

**Network Interface:-**Main User of Network interface is if instance fails then we can attach same network interface with another instance also

**Why should we attach network interface?**

Communicate with Vpc and external network with in network

What will it store network interface?

It will store security group ,subnet etc..

**Example**: If a **web application** hosted on EC2 is using an **Elastic Load Balancer** and each EC2 instance has multiple ENIs (one for public-facing traffic, another for internal communication), when a failure occurs in the **public subnet** of one instance (e.g., due to an AZ issue), the ELB can route traffic to the same instance’s **secondary ENI** in the private subnet, or to other instances in another AZ. This failover mechanism minimizes the impact of network failure and helps the system stay online

How to create:

1)selecting subnet

2)austo assign or custom ipv4 address 3)tags (Optional)

**PRIMARY NETWORK INTERFACE:-**The interface directly attach to instance when created It is primary

**SECONDARY NETWORK INTERFACE:-**if we have many network interfaces that is created by user then it is called secondary network

**To create a network interface using the console**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Network Interfaces**.
3. Choose **Create network interface**.
4. (Optional) For **Description**, enter a descriptive name.
5. For **Subnet**, select a subnet. The options available in the subsequent steps change depending on the type of subnet you select (IPv4-only, IPv6-only, or dual-stack (IPv4 and IPv6)).
6. For **Private IPv4 address**, do one of the following:
   * Choose **Auto-assign** to allow Amazon EC2 to select an IPv4 address from the subnet.
   * Choose **Custom** and enter an IPv4 address that you select from the subnet.
7. (Subnets with IPv6 addresses only) For **IPv6 address**, do one of the following:
   * Choose **None** if you do not want to assign an IPv6 address to the network interface.
   * Choose **Auto-assign** to allow Amazon EC2 to select an IPv6 address from the subnet.
   * Choose **Custom** and enter an IPv6 address that you select from the subnet.
8. (Optional) If you’re creating a network interface in a dual-stack or IPv6-only subnet, you have the option to **Assign Primary IPv6 IP**. This assigns a primary IPv6 global unicast address (GUA) to the network interface. Assigning a primary IPv6 address enables you to avoid disrupting traffic to instances or ENIs. Choose **Enable** if the instance that this ENI will be attached to relies on its IPv6 address not changing. AWS will automatically assign an IPv6 address associated with the ENI attached to your instance to be the primary IPv6 address. Once you enable an IPv6 GUA address to be a primary IPv6, you can't disable it. When you enable an IPv6 GUA address to be a primary IPv6, the first IPv6 GUA will be made the primary IPv6 address until the instance is terminated or the network interface is detached. If you have multiple IPv6 addresses associated with an ENI attached to your instance and you enable a primary IPv6 address, the first IPv6 GUA address associated with the ENI becomes the primary IPv6 address.
9. (Optional) To create an Elastic Fabric Adapter, choose **Elastic Fabric Adapter**, **Enable**.
10. (Optional) Under **Advanced settings**, for **Idle connection tracking timeout**, modify the default idle connection timeouts. For more information about these options, see [Idle connection tracking timeout](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/security-group-connection-tracking.html#connection-tracking-timeouts).
    * **TCP established timeout**: Timeout (in seconds) for idle TCP connections in an established state. Min: 60 seconds. Max: 432000 seconds (5 days). Default: 432000 seconds. Recommended: Less than 432000 seconds.
    * **UDP timeout**: Timeout (in seconds) for idle UDP flows that have seen traffic only in a single direction or a single request-response transaction. Min: 30 seconds. Max: 60 seconds. Default: 30 seconds.
    * **UDP stream timeout**: Timeout (in seconds) for idle UDP flows classified as streams which have seen more than one request-response transaction. Min: 60 seconds. Max: 180 seconds (3 minutes). Default: 180 seconds.
11. For **Security groups**, select one or more security groups.
12. (Optional) For each tag, choose **Add new tag** and enter a tag key and an optional tag value.
13. Choose **Create network interface**.

**Hibernation:-**Amazon Hibernation store the data from RAM and store in EBS(Elastic Block Store) and we can see that in Recycle bin and we can restore it

* The EBS root volume is restored to its previous state
* The RAM contents are reloaded
* The processes that were previously running on the instance are resumed
* Previously attached data volumes are reattached and the instance retains its instance ID
* How to create?

While creating Instance enable hibernation in advanced settings and whole Instance CONNECT if we click Uptime the time will coniunue if we delete instance and restore it also

**What happens when you hibernate an instance**

When you hibernate an instance, the following happens:

* The instance moves to the stopping state. Amazon EC2 signals the operating system to perform hibernation (suspend-to-disk). The hibernation freezes all of the processes, saves the contents of the RAM to the EBS root volume, and then performs a regular shutdown.
* After the shutdown is complete, the instance moves to the stopped state.
* Any EBS volumes remain attached to the instance, and their data persists, including the saved contents of the RAM.
* Any Amazon EC2 instance store volumes remain attached to the instance, but the data on the instance store volumes is lost.
* While your instance is in the stopped state, you can modify certain attributes of the instance, including the instance type or size.
* In most cases, the instance is migrated to a new underlying host computer when it's started. This is also what happens when you stop and start an instance.
* When the instance is started, the instance boots up and the operating system reads in the contents of the RAM from the EBS root volume, before unfreezing processes to resume its state.
* The instance retains its private IPv4 addresses and any IPv6 addresses. When the instance is started, the instance continues to retain its private IPv4 addresses and any IPv6 addresses.
* Amazon EC2 releases the public IPv4 address. When the instance is started, Amazon EC2 assigns a new public IPv4 address to the instance.
* The instance retains its associated Elastic IP addresses. You're charged for any Elastic IP addresses that are associated with a hibernated instance.

**Limitations**

* When you hibernate an instance, the data on any instance store volumes is lost.
* (Linux instances) You can't hibernate a Linux instance that has more than 150 GB of RAM.
* (Windows instances) You can't hibernate a Windows instance that has more than 16 GB of RAM.
* If you create a snapshot or AMI from an instance that is hibernated or has hibernation enabled, you might not be able to connect to a new instance that is launched from the AMI or from an AMI that was created from the snapshot.
* (Spot Instances only) If Amazon EC2 hibernates your Spot Instance, only Amazon EC2 can resume your instance. If you hibernate your Spot Instance ([user-initiated hibernation](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hibernating-instances.html)), you can resume your instance. A hibernated Spot Instance can only be resumed if capacity is available and the Spot price is less than or equal to your specified maximum price.
* You can't hibernate an instance that is in an Auto Scaling group or used by Amazon ECS. If your instance is in an Auto Scaling group and you try to hibernate it, the Amazon EC2 Auto Scaling service marks the stopped instance as unhealthy, and might terminate it and launch a replacement instance. For more information, see [Health checks for instances in an Auto Scaling group](https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-health-checks.html) in the *Amazon EC2 Auto Scaling User Guide*.
* You can't hibernate an instance that is configured to boot in UEFI mode with [UEFI Secure Boot](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/uefi-secure-boot.html) enabled.
* If you hibernate an instance that was launched into a Capacity Reservation, the Capacity Reservation does not ensure that the hibernated instance can resume after you try to start it.
* You can’t hibernate an instance that uses a kernel below 5.10 if Federal Information Processing Standard (FIPS) mode is enabled.
* We do not support keeping an instance hibernated for more than 60 days. To keep the instance for longer than 60 days, you must start the hibernated instance, stop the instance, and start it.
* We constantly update our platform with upgrades and security patches, which can conflict with existing hibernated instances. We notify you about critical updates that require a start for hibernated instances so that we can perform a shutdown or a reboot to apply the necessary upgrades and security patches.

**Considerations for hibernating a Spot Instance**

* If *you* hibernate your Spot Instance, you can restart it provided capacity is available and the Spot price is less than or equal to your specified maximum price.
* If *Amazon EC2* hibernates your Spot Instance:
  + Only Amazon EC2 can resume your instance.
  + Amazon EC2 resumes the hibernated Spot Instance when capacity becomes available with a Spot price that is less than or equal to your specified maximum price.
  + Before Amazon EC2 hibernates your Spot Instance, you'll receive an interruption notice two minutes before hibernation starts.

**Prerequisists For Hibernation**

* + You can enable hibernation support for an On-Demand Instance or a Spot Instance when you launch it. You can't enable hibernation on an existing instance, whether it is running or stopped.

**SECTION 7:-**

**EC2 Instance Storage**

**56)EBS volume:-**Ebs volume is used to stre data and we can terminate also

* It cannot be done in different Availability Zones
* We can attach Multiple ebs to one instance but we cannot attach multiple instances to one EBS
* That is new instance new EBS
* We can detach to one instance and attach to another instance very quickly

**How can we achieve in different availability zones?**

* If we take a snapshot we can keep it in different availability zones
* By default if we delete a instance the default EBS will be deleted but the storage will be there

**57)EBS HandsOn:-**

* We have to specify volume type
* General purpose(gp2 or gp3) or provosined(io1 or io2)
* Specify size or throughput like in gigabytes
* IOps(default)
* Availaibilityzone(must be same as instance availability zone)
* Create Volume
* Attach that volume to the instance

**58)EBS snapshot:-**

* Ebs snapshot is used to back up the ebs data
* We can copy that data to different availability zones

**Q1: What is an EBS snapshot, and how does it work?**

* **Answer:** An EBS snapshot is a point-in-time backup of an EBS volume. It captures only the changes made to the volume since the last snapshot, which makes it incremental and cost-effective. Snapshots are stored in Amazon S3 for durability and can be used to create new volumes in the same or different region.

**Q2: How is an EBS snapshot different from a traditional backup?**

* **Answer:** EBS snapshots are incremental, meaning after the first snapshot, only the changes to the volume are stored, while traditional backups usually copy the entire data each time. This makes snapshots more efficient and cost-effective. Also, snapshots are online and don't require downtime, unlike some traditional backups.

**Q3: Can you restore an EBS snapshot to a different region?**

* **Answer**: Yes, you can copy EBS snapshots across regions. This is useful for disaster recovery, data migration, or replicating a setup across multiple regions.

**Q4: What happens when you take an EBS snapshot of a running instance?**

* **Answer:** Taking a snapshot of a running EC2 instance is non-disruptive. The instance continues to run, and the snapshot captures the state of the EBS volume at the block level. However, it may not always guarantee consistency for running applications (like databases) unless they are prepared for it (e.g., paused or quiesced).

**Q5: How do you ensure that EBS snapshots are consistent when backing up databases?**

* **Answer:** To ensure application consistency, it's recommended to either stop the database before taking a snapshot or use application-level tools or services (e.g., Amazon RDS snapshots) that ensure consistency. Alternatively, you can use Volume Shadow Copy Service (VSS) for Windows-based applications.

**Q6: How does AWS handle snapshot encryption?**

**Answer:** EBS snapshots can be encrypted. If the EBS volume is encrypted, its snapshot will automatically be encrypted. You can also create encrypted snapshots from unencrypted volumes, and these encrypted snapshots can only be restored to encrypted volumes.

**59)EBS Snapshots Handson:-**

1. Create Volume
2. Create Snapshot from Volume(so that the traffic will be stored)By Copying the snapshot
3. After creating snapshot then we can create volume from snapshot so that the same data will be there
4. After we can create retention rule so that if in case deleted by mistake it can available in Recycle bin.

**60)AMI Review:-**

**An Amazon Machine Image (AMI)** is a pre-configured template that contains the operating system, application server.

* For example a person is building a web application using Apache,Spark then instead of repeating process we can create AMI
* From that we can create instance so that we can get same confirmation of AMI
* You can launch instance in 3 ways

1. Public AMI
2. Your Own AMI
3. From Market place:privately owned that is placed in market place

61)**AMI HandsOn:-**

1. Create instance
2. Create Image from instance
3. Remove check box
4. Noow create another instance from image

62)**EC2 Instance Store:-**

* Better Input output Higher Performance we use this.If we want high performance we use it.It is a hardware attached to it
* Loss:-Used for shor term periods
* If the instance is deleted the data will loss and we cannot restore it
* If we have buffer data we cam use the instance store

63)**EBS Volume Types:-**

* **GP2**:-
* Base for 3 iops
* Bursts upt o 3000
* Upto 25Mbbs
* Limited IOPs

**GP1:-**Base up to 3000 to 16000

Low and for large data bases

**Io1:-**0.001% fail rate **Io2:-**0.1-0.2 failure rate

40000 iops,1000 Mb/s,4 gib-16gib If need High availaibi;ity and high Scalability

Use Io1 and IO2

St1 (for frequent access) and Sl1 for in frequent access

64)**Ebs Multi Attach:-**

* Attaching the same EBS volume to multiple Ec2 instances in same Availability Zone
* Each instance have full read and write permssions

Use case limitations:-

* Achieve High Application Availability
* We can attach with in same availability zone
* Up tio 16 instances only

Main use of IO1,IO2 and Gp1 and GP2:

IO1 and IO2(provisioned ):-we can attach up to 16 instances only

**EBS Encryption(65):-**

Data,volumes,snapshots will be encrypted..

Any snapshot from Non encrypted volume is not encrypted we can encrypt through short cut or we can encrypt by copying snapshot

Encryption key is used to perform is managed by AWS KMS

KMS:-for Managing encryption keys

AES256:-Encrypt data

**Hands On:-**

Create EBS snapshot-> Encrypt using copy->create New Ebs From Snapshot-> Create Volume from Encrypted Snapshot

66)**Elastic file System:-** if we have different instances in different availability zones then they have to connect then the common storage will be efs

* Only used in Linux
* When we create a instance in one availability zone or selected different availability zones then the data will be same in other availability zone also

There are some parts in EFS

1. Performance:-

Generalpurpose

MaxI/o

1. Throughput:-

Bursting,provisioned,elastic

3.Storage tiers:-

If we want to move files to different storage tiers we use this

3 types:-

Standard :frequent access

Infrequent ace

Archieve:-access rarely

67) **EFS** **HandsOn:-**

* Create filesystem
* We can create directly but by customization we can see
* Name (file system name)
* Filesystem type

Regional(most used)

Onezone(it will deploy only in one AZ)

* Lycycle management(infrequent,standard,archieve)
* Performance or throughput(enhanced(elastic(generalpurpose or MaxIo), provisional)),Bursting
* Vpc
* Availability zones
* Security groups
* Attach security groups to Azs
* Create EFS

Difference between EFS AND EBS is we EFS can be done in multiple AZs but EBS can be done in single Availability zone

**SECTION-8 \*:-**

**AVAILAIBILITY**

**AND**

**SCALABILITY**

Availability:greater loads adapting it self

2 typpes

1)vertical:-increase in size vertically

2)horizontal:- increasing in instances ex:-Autoscaling group and LoadBalancer

High Availability:-Run instances from Same application across availability Zones

* **Use Case**: When you want to invoke AWS Lambda functions in response to requests routed through the Application Load Balancer (ALB).
* **When to Use**:
  + You have serverless applications where the backend logic is implemented using Lambda functions.
  + You want to handle HTTP(S) requests directly with Lambda, allowing for automatic scaling and reduced operational overhead.
  + You are building microservices that can be triggered by HTTP requests without needing a dedicated server.

**2. IP Addresses**

* **Use Case**: When you want to route traffic to specific IP addresses, which could be on-premises servers, EC2 instances, or other resources outside of AWS.
* **When to Use**:
  + You have a hybrid architecture where some of your services are hosted on-premises or in another cloud provider.
  + You want to route traffic to specific instances or services that are not registered with the load balancer.
  + You need to target resources that are not part of an AWS VPC (Virtual Private Cloud).

**3. Application Load Balancer (ALB)**

* **Use Case**: When you want to route traffic to multiple targets (like EC2 instances, containers, or Lambda functions) based on rules and conditions.
* **When to Use**:
  + You are building a web application that requires advanced routing capabilities, such as path-based or host-based routing.
  + You need to distribute incoming application traffic across multiple targets to ensure high availability and fault tolerance.
  + You want to take advantage of features like WebSocket support, HTTP/2, and SSL termination.

**Scenario Overview**

1. **User Access**: A user wants to access an application.
2. **Load Balancer Setup**:
   * **Network Load Balancer (NLB)**: Handles TCP traffic.
   * **Application Load Balancer (ALB)**: Handles HTTP requests and performs application-level load balancing.
   * **Gateway Load Balancer (GWLB)**: Integrates with firewall services for traffic inspection and security.

**Detailed Configuration Steps**

1. **Network Load Balancer (NLB)**:
   * **Purpose**: To handle TCP traffic from users.
   * **Target Group**: Create a target group with the target type set to **IP**. This allows the NLB to route traffic to specific IP addresses of your backend instances or services.
   * **Configuration**:
     + Set up listeners for TCP traffic on the desired port (e.g., port 80 for HTTP or port 443 for HTTPS).
     + Route incoming TCP traffic to the appropriate target group.
2. **Application Load Balancer (ALB)**:
   * **Purpose**: To handle HTTP/HTTPS requests and perform application-level load balancing.
   * **Target Group**: Create a target group for the ALB that includes the instances or services that will handle HTTP requests.
   * **Configuration**:
     + Set up listeners for HTTP (port 80) and/or HTTPS (port 443).
     + Define routing rules based on URL paths or hostnames to direct traffic to the appropriate backend services.
3. **Gateway Load Balancer (GWLB)**:
   * **Purpose**: To provide firewall services and traffic inspection.
   * **Target Group**: Create a target group for the GWLB that includes the virtual appliances (e.g., firewalls) that will inspect the traffic.
   * **Configuration**:
     + Set up the GWLB to route traffic through the firewall appliances.
     + Ensure that the GWLB is integrated with the ALB to inspect traffic before it reaches the application.

**Combinationof Application+Network+Gateway**

**i have a scenario** i have user he wan to access applicatrion hen i will create 3 different tarbet groups for 3 load balancer so first user will access Netwrok load balancer so the tcp or networks nternally handle iw ill keep TG as Ip the Application load balancer i will create for HTTp request and balancing on same machine and i will keep Gateway load balancer for firewall then move to application can i do like this?give in correct format

**Traffic Flow**

1. **User Request**:
   * The user initiates a request to access the application.
2. **Network Load Balancer (NLB)**:
   * The request first hits the NLB, which handles the TCP connection.
   * The NLB routes the TCP traffic to the appropriate target group based on the configured rules.
3. **Gateway Load Balancer (GWLB)**:
   * The traffic is then sent to the GWLB, where it is inspected by the firewall appliances.
   * The GWLB ensures that only safe and compliant traffic is allowed to proceed.
4. **Application Load Balancer (ALB)**:
   * After passing through the GWLB, the traffic reaches the ALB.
   * The ALB processes the HTTP requests, applying any routing rules defined (e.g., directing traffic to specific services based on URL paths).
5. **Application**:
   * Finally, the ALB forwards the request to the appropriate backend application instance or service.
   * The application processes the request and sends the response back through the same path (ALB → GWLB → NLB → User).

**Summary**

In this setup, you effectively use three different load balancers to manage traffic at different layers:

* **NLB** for TCP traffic management.
* **GWLB** for security and traffic inspection.
* **ALB** for application-level routing and load balancing.

This architecture provides a robust and secure way to handle user requests while ensuring that traffic is properly managed and inspected at each stage