# # Scenario Based Interview Questions on EC2, IAM and VPC

# Q: You have been assigned to design a VPC architecture for a 2-tier application. The application needs to be highly available and scalable. How would you design the VPC architecture?

1. In this scenario, I would design a VPC architecture in the following way.
2. I would create 2 subnets: public and private. The public subnet would contain the load balancers and be accessible from the internet. The private subnet would host the application servers.
3. I would distribute the subnets across multiple Availability Zones for high availability. Additionally, I would configure auto scaling groups for the application servers.

# Q: Your organization has a VPC with multiple subnets. You want to restrict outbound internet access for resources in one subnet, but allow outbound internet access for resources in another subnet. How would you achieve this?

1. To restrict outbound internet access for resources in one subnet, we can modify the route table associated with that subnet. In the route table, we can remove the default route (0.0.0.0/0) that points to an internet gateway.
2. This would prevent resources in that subnet from accessing the internet. For the subnet where outbound internet access is required, we can keep the default route pointing to the internet gateway.

# Q: You have a VPC with a public subnet and a private subnet. Instances in the private subnet need to access the internet for software updates. How would you allow internet access for instances in the private subnet?

1. To allow internet access for instances in the private subnet, we can use a NAT Gateway or a NAT instance.
2. We would place the NAT Gateway/instance in the public subnet and configure the private subnet route table to send outbound traffic to the NAT Gateway/instance. This way, instances in the private subnet can access the internet through the NAT Gateway/instance.

# Q: You have launched EC2 instances in your VPC, and you want them to communicate with each other using private IP addresses. What steps would you take to enable this communication?

1. By default, instances within the same VPC can communicate with each other using private IP addresses.
2. To ensure this communication, we need to make sure that the instances are launched in the same VPC and are placed in the same subnet or subnets that are connected through a peering connection or a VPC peering link.
3. Additionally, we should check the security groups associated with the instances to ensure that the necessary inbound and outbound rules are configured to allow communication between them.

# Q: You want to implement strict network access control for your VPC resources. How would you achieve this?

1. To implement granular network access control for VPC resources, we can use Network Access Control Lists (ACLs).
2. NACLs are stateless and operate at the subnet level. We can define inbound and outbound rules in the NACLs to allow or deny traffic based on source and destination IP addresses, ports, and protocols.
3. By carefully configuring NACL rules, we can enforce fine-grained access control for traffic entering and leaving the subnets.

# Q: Your organization requires an isolated environment within the VPC for running sensitive workloads. How would you set up this isolated environment?

1. To set up an isolated environment within the VPC, we can create a subnet with no internet gateway attached.
2. This subnet, known as an "isolated subnet," will not have direct internet connectivity. We can place the sensitive workloads in this subnet, ensuring that they are protected from inbound and outbound internet traffic.
3. However, if these workloads require outbound internet access, we can set up a NAT Gateway or NAT instance in a different subnet and configure the isolated subnet's route table to send outbound traffic through the NAT Gateway/instance.

# Q: Your application needs to access AWS services, such as S3 securely within your VPC. How would you achieve this?

1. To securely access AWS services within the VPC, we can use VPC endpoints. VPC endpoints allow instances in the VPC to communicate with AWS services privately, without requiring internet gateways or NAT gateways.
2. We can create VPC endpoints for specific AWS services, such as S3 and DynamoDB, and associate them with the VPC.
3. This enables secure and efficient communication between the instances in the VPC and the AWS services.

# Q: What is the difference between NACL and Security groups? Explain with a use case?

1. For example, I want to design a security architecture, I would use a combination of NACLs and security groups. At the subnet level, I would configure NACLs to enforce inbound and outbound traffic restrictions based on source and destination IP addresses, ports, and protocols. NACLs are stateless and can provide an additional layer of defense by filtering traffic at the subnet boundary.
2. At the instance level, I would leverage security groups to control inbound and outbound traffic. Security groups are stateful and operate at the instance level. By carefully defining security group rules, I can allow or deny specific traffic to and from the instances based on the application's security requirements.
3. By combining NACLs and security groups, I can achieve granular security controls at both the network and instance level, providing defense-in-depth for the sensitive application.

# Q: What is the difference between IAM users, groups, roles and policies?

1. A: IAM User: An IAM user is an identity within AWS that represents an individual or application needing access to AWS resources. IAM users have permanent long-term credentials, such as a username and password, or access keys (Access Key ID and Secret Access Key). IAM users can be assigned directly to IAM policies or added to IAM groups for easier management of permissions.
2. IAM Role: An IAM role is similar to an IAM user but is not associated with a specific individual. Instead, it is assumed by entities such as IAM users, applications, or services to obtain temporary security credentials. IAM roles are useful when you want to grant permissions to entities that are external to your AWS account or when you want to delegate access to AWS resources across accounts. IAM roles have policies attached to them that define the permissions granted when the role is assumed.
3. IAM Group: An IAM group is a collection of IAM users. By organizing IAM users into groups, you can manage permissions collectively. IAM groups make it easier to assign permissions to multiple users simultaneously. Users within an IAM group inherit the permissions assigned to that group. For example, you can create a "Developers" group and assign appropriate policies to grant permissions required for developers across your organization.
4. IAM Policy: An IAM policy is a document that defines permissions and access controls in AWS. IAM policies can be attached to IAM users, IAM roles, and IAM groups to define what actions can be performed on which AWS resources. IAM policies use JSON (JavaScript Object Notation) syntax to specify the permissions and can be created and managed independently of the users, roles, or groups. IAM policies consist of statements that include the actions allowed or denied, the resources on which the actions can be performed, and any additional conditions.

# Q: You have a private subnet in your VPC that contains a number of instances that should not have direct internet access. However, you still need to be able to securely access these instances for administrative purposes. How would you set up a bastion host to facilitate this access?

1. To securely access the instances in the private subnet, you can set up a bastion host (also known as a jump host or jump box). The bastion host acts as a secure entry point to your private subnet. Here's how you can set up a bastion host:
2. Create a new EC2 instance in a public subnet, which will serve as the bastion host. Ensure that this instance has a public IP address or is associated with an Elastic IP address for persistent access.
3. Configure the security group for the bastion host to allow inbound SSH (or RDP for Windows) traffic from your IP address or a restricted range of trusted IP addresses. This limits access to the bastion host to authorized administrators only.
4. Place the instances in the private subnet and configure their security groups to allow inbound SSH (or RDP) traffic from the bastion host security group.
5. SSH (or RDP) into the bastion host using your private key or password. From the bastion host, you can then SSH (or RDP) into the instances in the private subnet using their private IP addresses.

# Load Balancer Questions and Answers

**Q1. What is a Load Balancer?**

A **Load Balancer** distributes incoming traffic across multiple servers (EC2 instances, containers, or IPs) to ensure:

* High availability ✅
* Fault tolerance ✅
* Better performance ✅

**Q2. What is Elastic Load Balancing (ELB) in AWS?**

**ELB** that is **automatically distributes incoming traffic** across multiple **EC2 instances**, to improve **availability**, **scalability**, and **fault tolerance**.

**Q3. What are the types of Load Balancers in AWS?**

AWS offers **3 types** under **Elastic Load Balancing**:

1. **Application Load Balancer (ALB)** → HTTP/HTTPS (Layer 7) ✅ *(Best for web apps & microservices)*
2. **Network Load Balancer (NLB)** → TCP/UDP/TLS (Layer 4) ⚡ *(Best for high-performance apps)*
3. **Gateway Load Balancer (GWLB)** → Security appliances 🔐 *(Best for firewalls & packet inspection)*

**Q4. Difference between ALB and NLB?**

| **Feature** | **Application LB (ALB)** | **Network LB (NLB)** |
| --- | --- | --- |
| **Layer** | Layer 7 | Layer 4 |
| **Protocol** | HTTP, HTTPS, WebSocket | TCP, UDP, TLS |
| **Routing** | Path-based, Host-based ✅ | No advanced routing ❌ |
| **Latency** | Higher (~10ms) | Very low (<1ms) ✅ |
| **Static IP** | ❌ Not supported | ✅ Supported |
| **Use Case** | REST APIs, microservices | Gaming, chat, IoT, high-performance apps |

**Q5. What is a Target Group?**

A **Target Group** is a collection of servers (EC2, containers, or IPs) that receive traffic from the **Load Balancer**.

**Q6. What is a Listener in AWS Load Balancer?**

A **Listener** checks for **incoming requests** using:

* **Protocol** → HTTP, HTTPS, TCP, or TLS
* **Port** → e.g., **80**, **443**, **8080**

Example:  
For a Spring Boot app running on **8080**, listener = **TCP:8080**.

**Q7. What is Cross-Zone Load Balancing?**

It distributes incoming traffic **evenly across EC2 instances** in **all Availability Zones (AZs)** instead of just one.  
✅ Recommended for **better load distribution**.

**Q8. What are Health Checks in Load Balancer?**

* **Health checks** automatically detect whether a target (EC2, container, or IP) is **healthy** or **unhealthy**.
* If a target is unhealthy, the LB **stops sending traffic** to it.

Example:  
For Spring Boot, use **/actuator/health** as a health check endpoint.

**Q9. What is the difference between Internet-facing and Internal Load Balancer?**

| **Type** | **Use Case** |
| --- | --- |
| **Internet-facing LB** | Used for **public-facing apps**, accessible over the internet |
| **Internal LB** | Used for **private apps**, accessible only inside a VPC |

**Q10. What are the benefits of using Load Balancers?**

* ✅ High availability
* ✅ Automatic failover
* ✅ Better performance
* ✅ Security (HTTPS + SSL termination)
* ✅ Integrates with Auto Scaling

**2. Intermediate Questions**

**Q11. How does AWS ALB route requests to microservices?**

* **Path-based routing** → /auth → Auth service, /payment → Payment service.
* **Host-based routing** → api.example.com → API service, app.example.com → Web app.

**Q12. What is the difference between Path-based and Host-based routing?**

| **Routing Type** | **Example** | **Use Case** |
| --- | --- | --- |
| **Path-based** | /api/\* → API service | Microservices |
| **Host-based** | api.example.com → API service | Multi-domain apps |

**Q13. How does NLB handle TCP and UDP traffic?**

* NLB **forwards TCP/UDP packets** directly to the target without inspecting them.
* It ensures **ultra-low latency** and **high throughput**.

**Q14. What happens if an EC2 instance behind the Load Balancer fails?**

* The **Health Check** detects the failure.
* LB **stops sending traffic** to the failed instance.
* Auto Scaling Group may launch a **new EC2 instance** automatically.

**Q15. How do you enable HTTPS in ALB or NLB?**

* Use **AWS Certificate Manager (ACM)** to issue an SSL/TLS certificate.
* Attach the certificate to the **HTTPS listener** (port **443**).

**Q16. What are Sticky Sessions in ALB?**

* Sticky sessions ensure a **user always connects to the same EC2 instance**.
* Useful for **session-based apps** like shopping carts.

**Q17. What is SSL Termination vs SSL Passthrough?**

| **Feature** | **SSL Termination** | **SSL Passthrough** |
| --- | --- | --- |
| Where SSL is decrypted | At Load Balancer | At EC2 instance |
| Performance | Faster ✅ | Slightly slower ❌ |
| Use Case | Normal web apps | Security-sensitive apps |

**Q18. How does Load Balancer integrate with Auto Scaling?**

* Load Balancer connects to an **Auto Scaling Group (ASG)**.
* When traffic increases → **ASG launches new EC2s**.
* When traffic decreases → **ASG terminates unused EC2s**.

**3. Advanced & Real-Time Questions**

**Q19. You have 3 microservices: /auth, /order, /payment. How do you set up ALB?**

* Create **one ALB**.
* Add **3 target groups**.
* Configure **path-based routing**:
  + /auth/\* → Auth Target Group
  + /order/\* → Order Target Group
  + /payment/\* → Payment Target Group

**Q20. Which Load Balancer should you choose for Spring Boot microservices?**

* **Application Load Balancer (ALB)** ✅
* Because:
  + Works with **HTTP/HTTPS**
  + Supports **path-based routing**
  + Integrates with **Spring Boot actuator health checks**

**Q21. When should you prefer NLB instead of ALB?**

Use **NLB** when:

* You need **TCP/UDP support**
* **Low latency** is critical (<1ms)
* You need **static IP addresses**
* Example: Gaming apps, chat apps, IoT apps, financial trading.

**Q22. How does Route 53 work with Load Balancers?**

* Use **AWS Route 53** to map your **domain name** (e.g., api.example.com) to the **ALB/NLB DNS**.
* Ensures high availability and global routing.

**Q23. What happens if all targets fail?**

* ALB/NLB **returns 503 Service Unavailable**.
* You must ensure **Auto Scaling** is enabled for recovery.

**Q24. How do you monitor Load Balancer performance?**

* Use **AWS CloudWatch Metrics**:
  + Request count
  + Target response time
  + Healthy/unhealthy hosts
  + 4xx/5xx errors

# RDS Interview Q/A

**1. What is Amazon RDS and its benefits?**

**Answer:** RDS is a **managed relational database service**.  
**Benefits:**

* Automatic backups & patching
* High availability (Multi-AZ)
* Read scaling (Read Replicas)
* Easy monitoring via CloudWatch

**2. Difference between Multi-AZ and Read Replica**

| **Feature** | **Multi-AZ** | **Read Replica** |
| --- | --- | --- |
| Purpose | High availability | Read scaling |
| Replication | Synchronous | Asynchronous |
| Failover | Automatic | No |
| Readable? | No | Yes |

**3. Explain RDS Backups**

* **Automated Backups:** Daily snapshots + transaction logs, point-in-time recovery (max 35 days).
* **Manual Snapshots:** Created anytime, retained until deleted.
* **Purpose:** Disaster recovery, safe maintenance, data protection.

**4. How does RDS failover work?**

* In **Multi-AZ**, if the primary DB fails:
  + AWS **promotes the standby** DB automatically.
  + **DNS endpoint** is updated.
  + Application reconnects without code changes.

**5. How to scale RDS?**

* **Vertically:** Change **DB instance class** (CPU/RAM).
* **Horizontally:** Use **Read Replicas** for read-heavy workloads.
* **Storage scaling:** Increase storage without downtime (depending on engine).

**6. Point-in-Time Recovery (PITR)**

* Allows restoring the database **to any second** within the **backup retention period**.
* Works with **automated backups + transaction logs**.

**7. How to create a Read Replica?**

1. Go to **RDS → Databases → Primary DB → Actions → Create Read Replica**.
2. Configure **instance class, VPC, AZ**.
3. Click **Create**.
4. Use the **replica endpoint** in the application for **read queries**.

**8. Difference between RDS and EC2-hosted database**

* **RDS:** Managed service, automatic backups, patching, monitoring, scaling.
* **EC2 DB:** Fully manual management, everything is your responsibility.

**9. Common Performance Tips**

* Use **Read Replicas** for heavy reads.
* Enable **Enhanced Monitoring & CloudWatch alarms**.
* Use **Multi-AZ** for high availability.
* Optimize **DB queries & indexes**.

10**. How do you restore an RDS instance to a specific point in time?**

**Answer:**

* Use **Point-in-Time Recovery (PITR)** with automated backups.
* Go to **RDS → Snapshots → Restore to point in time**.
* Select the **desired timestamp** within the backup retention period.
* AWS creates a **new DB instance** restored to that point.
* Update your application to point to the **new endpoint** if required.

**11. How do you monitor RDS performance in production?**

**Answer:**

* Use **Amazon CloudWatch** metrics like:
  + CPUUtilization, FreeStorageSpace, DBConnections, Read/Write IOPS, Latency.
* Enable **Enhanced Monitoring** for OS-level metrics.
* Use **Performance Insights** to identify slow queries.
* Set **alarms and notifications** for thresholds (CPU, storage, connections).
* Regularly review **slow query logs** and optimize indexes/queries.

**12. How do you handle high read traffic in RDS?**

**Answer:**

* Create **Read Replicas** to offload read-heavy queries from the primary DB.
* Use **Application Load Balancing** or routing logic to distribute reads to replicas.
* Enable **query caching** where applicable.
* Optimize **indexes** and database schema to improve read performance.
* Consider **Aurora** if you need **auto-scaling read replicas**.

**13. How do you migrate a database to RDS with minimal downtime?**

**Answer:**

* Use **AWS Database Migration Service (DMS)** for live migration.
* Steps:
  1. Set up **RDS target DB**.
  2. Configure **DMS replication task** to replicate existing data continuously.
  3. Keep **source DB writes** synced during migration.
  4. Switch application to **RDS endpoint** once replication is caught up.
* For MySQL/PostgreSQL, you can also use **native replication** + minimal downtime cutover.