AWS Interview Questions and Answers

1. What is AWS?

Answer:
AWS (Amazon Web Services) is a comprehensive cloud computing platform provided by Amazon. It offers on-demand services such as compute, storage, database, networking, machine learning, and analytics over the internet. AWS allows businesses to scale infrastructure without upfront hardware costs.

Key points to remember:

- On-demand, pay-as-you-go pricing.
- Global infrastructure with Regions and Availability Zones.
- Highly scalable, secure, and reliable.

---

2. What are the benefits of AWS?

Answer:
AWS provides several benefits for businesses and developers:

- **Scalability:** Scale up or down based on demand.
- **Cost-effective:** Pay only for what you use.
- **Global Reach:** Multiple Regions and Availability Zones.
- **High Availability & Reliability:** Built-in redundancy.
- **Security:** Shared responsibility model and compliance certifications.
- **Flexibility:** Supports multiple OS, programming languages, and databases.

---

3. Explain the AWS shared responsibility model.

Answer:
AWS security responsibilities are **shared** between AWS and the customer:

- **AWS (Security *of* the Cloud):** Hardware, network, facilities, managed services security.
- **Customer (Security *in* the Cloud):** Data, applications, IAM, OS configurations, encryption, and access management.

4. What is Amazon EC2?

Answer:
Amazon EC2 (Elastic Compute Cloud) is a web service that provides resizable virtual servers (instances) in the cloud. You can launch, stop, and manage servers on-demand and only pay for the compute resources you use.

Key points:

- Supports multiple instance types for different workloads.
- Can run Windows, Linux, or custom AMIs.
- Integrates with other AWS services like VPC, EBS, and Auto Scaling.
- Pay-as-you-go or Reserved/Spot instance options.

---

5. What is Amazon S3?

Answer:
Amazon S3 (Simple Storage Service) is an object storage service that stores and retrieves any amount of data from anywhere. It is highly durable, scalable, and secure.

Key points:

- Stores data as **objects** in **buckets**.
- 99.999999999% (11 9's) durability.
- Supports versioning, lifecycle policies, and encryption.
- Pay only for storage used.

---

6. What is the difference between S3 and EBS?

Answer:

Feature	S3	EBS
Type	Object storage	Block storage
Usage	Backup, static files, media	EC2 instance disk storage
Access	Via HTTP(S)	Mounted to EC2 instances
Durability	11 9's	99.999% (depends on replication)
Scalability	Virtually unlimited	Limited by volume type & region

Key point: S3 is for scalable object storage, EBS acts like a hard drive for EC2 instances.

---

7. What is Amazon RDS?

Answer:
Amazon RDS (Relational Database Service) is a managed service for relational databases. It handles database provisioning, patching, backups, and scaling automatically.

Key points:

- Supports MySQL, PostgreSQL, Oracle, SQL Server, and Aurora.
- High availability with Multi-AZ deployments.
- Automated backups, snapshots, and failover support.
- Reduces database management overhead.

8. What is AWS Lambda?

Answer:
AWS Lambda is a **serverless compute service** that runs your code without provisioning or managing servers. You pay only for the compute time your code consumes.

Key points:

- Supports multiple languages: Python, Node.js, Java, Go, etc.
- Automatically scales based on the number of requests.
- Commonly used with API Gateway, S3 events, or DynamoDB streams.
- No server management is needed.

---

9. What is the difference between vertical and horizontal scaling in AWS?

Answer:

Scaling Type	Description	Example
Vertical Scaling	Increasing resources (CPU, RAM) on a single instance	Upgrading an EC2 t2.medium to t2.large
Horizontal Scaling	Adding more instances to distribute the load	Adding 3 more EC2 instances behind a load balancer

Key point: Horizontal scaling is preferred in cloud environments for high availability and fault tolerance.

---

10. What is an AMI?

Answer:
An AMI (Amazon Machine Image) is a **pre-configured template** used to launch EC2 instances. It contains:

- OS (Linux, Windows, etc.)
- Applications and software configurations
- Instance permissions

Key points:

- You can create your own custom AMI for reproducible server setups.
- AMIs can be shared across AWS accounts.

11. What is a VPC in AWS?

Answer:
VPC (Virtual Private Cloud) allows you to create a **logically isolated network** within AWS. You can launch AWS resources like EC2, RDS, and Lambda in this network.

Key points:

- You define **IP address ranges (CIDR blocks)**.
- Can create **subnets** (public/private).
- Control traffic with **route tables, security groups, and NACLs**.
- Can connect to on-premises networks via VPN or Direct Connect.

---

12. What is the difference between Security Groups and NACLs?

Feature	Security Group	NACL (Network ACL)
Scope	Instance level	Subnet level
Type	Stateful	Stateless
Rules	Allow only (inbound/outbound)	Allow and deny rules
Default Behavior	Deny all inbound, allow all outbound	Allow all by default
Use Case	EC2 instance firewall	Subnet traffic control

Key point: Security Groups are instance firewalls, while NACLs control subnet-level traffic.

---

13. What is AWS IAM?

Answer:
IAM (Identity and Access Management) allows you to **securely manage access** to AWS resources.

Key points:

- Create **users, groups, and roles**.
- Apply **policies** (permissions) for fine-grained access control.
- Supports **MFA** (Multi-Factor Authentication).
- Essential for security and compliance in AWS.

---

14. What are AWS Availability Zones (AZs) and Regions?

Answer:

- **Region:** A geographical area with multiple data centers (e.g., US-East-1).
- **Availability Zone (AZ):** An isolated data center within a region.

Key points:

- AZs provide **high availability and fault tolerance**.
- You can deploy resources across AZs for **redundancy**.
- Some services replicate automatically across AZs (like RDS Multi-AZ).

---

15. What is CloudFront?

Answer:
CloudFront is AWS's **Content Delivery Network (CDN)** that delivers data, videos, applications, and APIs to users **globally with low latency**.

Key points:

- Caches content at **edge locations** worldwide.
- Integrates with **S3, EC2, and Lambda@Edge**.
- Improves **performance and reliability** for web applications.

16. What is AWS CloudWatch?

Answer:
CloudWatch is a **monitoring and observability service** for AWS resources and applications.

Key points:

- Collects **metrics, logs, and events** from EC2, RDS, Lambda, etc.
- Can create **alarms** to trigger actions based on thresholds.
- Supports **dashboards** to visualize performance.
- Helps in **performance tuning and operational health monitoring**.

---

17. What is AWS CloudTrail?

Answer:
CloudTrail is a **logging and auditing service** that records all **API calls and account activity** within AWS.

Key points:

- Provides **event history** for security analysis and compliance.
- Tracks **who did what, when, and from where**.
- Can integrate with CloudWatch Logs for real-time monitoring.

---

18. What is Auto Scaling?

Answer:
Auto Scaling automatically adjusts the **number of EC2 instances** based on demand.

Key points:

- Ensures **high availability** by launching or terminating instances.
- Can scale based on **CPU usage, network traffic, or custom metrics**.
- Reduces cost by **removing unused instances**.

---

19. What is Elastic Load Balancer (ELB)?

Answer:
ELB distributes incoming traffic across **multiple EC2 instances** to ensure fault tolerance and scalability.

Key points:

- Supports **Classic, Application (ALB), and Network Load Balancers (NLB)**.
- Provides **health checks** to route traffic only to healthy instances.
- Works with **Auto Scaling** for dynamic scaling.

---

20. What is AWS SQS?

Answer:
SQS (Simple Queue Service) is a **fully managed message queuing service** that enables decoupling of applications.

Key points:

- Supports **standard queues** (high throughput) and **FIFO queues** (order guaranteed).
- Helps **reliably transmit messages** between distributed components.
- Ensures **asynchronous processing** and reduces system dependency.

21. What is AWS CloudFormation?

Answer:
CloudFormation is an **infrastructure-as-code (IaC) service** that allows you to define AWS resources in **JSON or YAML templates**.

Key points:

- Automates **provisioning and deployment** of resources.
- Ensures **consistent environments** across development, staging, and production.
- Supports **stack updates, rollbacks, and dependencies**.

- Integrates with **CI/CD pipelines**.

---

22. What is AWS CodePipeline?

Answer:
CodePipeline is a **fully managed continuous integration and continuous delivery (CI/CD) service**.

Key points:

- Automates **build, test, and deploy workflows**.
- Integrates with **CodeCommit, GitHub, Jenkins, and CloudFormation**.
- Supports **parallel and sequential actions** in the pipeline.

---

23. What is AWS CodeDeploy?

Answer:
CodeDeploy automates the **deployment of applications** to EC2 instances, Lambda functions, or on-premises servers.

Key points:

- Supports **blue/green and rolling deployments**.
- Reduces **manual deployment errors**.
- Integrates with **CodePipeline** for full CI/CD automation.

---

24. What is the difference between CloudFormation and Terraform?

Answer:

Feature	CloudFormation	Terraform
Provider	AWS only	Multi-cloud (AWS, Azure, GCP, etc.)
Language	JSON/YAML	HCL (HashiCorp Configuration Language)
State Management	Managed by AWS	Maintains state files (local or remote)
Community Support	AWS supported	Open-source, large community

Key point: Terraform is preferred for **multi-cloud IaC**, while CloudFormation is **AWS-native**.

---

25. What is AWS Elastic Beanstalk?

Answer:
Elastic Beanstalk is a **Platform-as-a-Service (PaaS)** that simplifies application deployment.

Key points:

- Automatically handles **provisioning, load balancing, scaling, and monitoring**.

- Supports **Java, Python, Node.js, PHP, .NET, Ruby, Go, and Docker**.
- You only need to **upload your code**, and Beanstalk handles the rest.

26. What is Amazon DynamoDB?

Answer:
DynamoDB is a **fully managed NoSQL database service** that provides fast and predictable performance with **automatic scaling**.

Key points:

- Stores data in **key-value and document formats**.
- Fully managed, serverless, and highly available.
- Supports **streams, TTL, and global tables**.
- Ideal for **high-traffic web applications and mobile apps**.

---

27. What is Amazon Aurora?

Answer:
Aurora is a **high-performance, fully managed relational database** compatible with MySQL and PostgreSQL.

Key points:

- Up to **5x faster than standard MySQL**.
- Supports **replication across multiple AZs** for high availability.
- Automated backups, snapshots, and failover support.
- Fully managed, reduces operational overhead.

---

28. What is Amazon EBS?

Answer:
EBS (Elastic Block Store) provides **block-level storage** for EC2 instances.

Key points:

- Acts like a **virtual hard drive** attached to an EC2 instance.
- Supports **SSD (gp3/io1) and HDD (st1/sc1)** volume types.
- Data persists even if the EC2 instance is stopped.
- Snapshots can be taken for **backup and disaster recovery**.

---

29. What is Amazon Glacier?

Answer:
Amazon Glacier is a **low-cost archival storage service** for long-term backup and data retention.

Key points:

- Very low cost but **retrieval times range from minutes to hours**.
- Ideal for **compliance and archival data**.
- Supports lifecycle policies to **move data from S3 to Glacier** automatically.

---

30. What is the difference between Amazon RDS and DynamoDB?

Answer:

Feature	RDS	DynamoDB
Database Type	Relational	NoSQL (key-value/document)
Scaling	Vertical (and read replicas)	Horizontal, automatic
Schema	Fixed	Flexible, schema-less
Use Case	OLTP, structured data	High-traffic web/mobile apps, unstructured data

31. What is Serverless Architecture in AWS?

Answer:
Serverless architecture allows you to **build and run applications without managing servers**. AWS handles server provisioning, scaling, and maintenance.

Key points:

- Services like **Lambda, API Gateway, DynamoDB, and S3** are commonly used.
- Pay only for **actual compute time or usage**.
- Ideal for **event-driven applications** and microservices.

---

32. What is Amazon ElastiCache?

Answer:
ElastiCache is a **fully managed in-memory caching service** for improving application performance.

Key points:

- Supports **Redis** and **Memcached**.
- Reduces **database load** by caching frequently accessed data.
- Provides **low latency and high throughput** for read-heavy workloads.

---

33. What is AWS CloudFront? *(Already briefly discussed, but key for advanced topics)*

Answer:
CloudFront is a **Content Delivery Network (CDN)** that delivers content globally with **low latency** by caching at edge locations.

Advanced use:

- Supports **dynamic content delivery** via Lambda@Edge.
- Works with **S3, EC2, and custom origins**.

---

34. What is High Availability (HA) in AWS?

Answer:
HA ensures that your applications remain **operational even if some components fail**.

Key points:

- Use **multiple AZs** for critical resources.
- Deploy **load balancers** and **auto scaling**.
- RDS supports **Multi-AZ deployments** for databases.

---

35. What is Disaster Recovery (DR) in AWS?

Answer:
DR ensures your applications and data can be **recovered in case of a failure or disaster**.

Key strategies:

1. **Backup and Restore** – Store backups in S3/Glacier.
2. **Pilot Light** – Keep minimal resources running, scale when needed.
3. **Warm Standby** – Reduced capacity environment running continuously.
4. **Multi-Region Active** – Fully replicated environments in another region.

36. What is the AWS Shared Responsibility Model?

Answer:
AWS and the customer **share security responsibilities**:

- **AWS (Security of the Cloud):** Physical infrastructure, networking, hardware, and managed services.

- **Customer (Security in the Cloud):** Data, applications, IAM, OS configuration, encryption, and access management.

Key point: Understanding this model is critical for security and compliance questions.

---

37. What is AWS KMS (Key Management Service)?

Answer:
KMS is a **managed service for creating and controlling encryption keys** used to encrypt your data.

Key points:

- Supports **envelope encryption** with S3, EBS, and RDS.
- Fully integrated with **IAM and CloudTrail**.
- Enables **automatic key rotation**.

---

38. What is AWS Cognito?

Answer:
Cognito provides **user authentication, authorization, and management** for web and mobile apps.

Key points:

- Supports **sign-up, sign-in, and access control**.
- Integrates with **social identity providers** like Google, Facebook, and SAML.
- Can manage **user pools and identity pools**.

---

39. What is AWS Shield?

Answer:
AWS Shield is a **managed Distributed Denial of Service (DDoS) protection service**.

Key points:

- **Standard:** Automatic protection for all AWS services at no extra cost.
- **Advanced:** Real-time attack detection and mitigation with **24/7 DDoS response team**.
- Protects **CloudFront, Route 53, ELB, and Global Accelerator**.

---

40. What is AWS WAF (Web Application Firewall)?

Answer:
AWS WAF protects web applications from **common web exploits** like SQL injection or XSS attacks.

Key points:

- Can define **custom rules** to block/allow traffic.
- Integrates with **CloudFront, ALB, and API Gateway**.
- Monitors and filters HTTP/HTTPS requests in real-time.

41. What is VPC Peering?

Answer:
VPC Peering allows you to **connect two VPCs privately** using AWS's network without going over the internet.

Key points:

- Can peer VPCs **within the same region or across regions**.
- Traffic is **private and secure**.
- No bandwidth bottleneck or single point of failure.
- Cannot have **overlapping CIDR blocks**.

---

42. What is AWS Direct Connect?

Answer:
Direct Connect provides a **dedicated network connection** from your on-premises data center to AWS.

Key points:

- Reduces **latency and bandwidth costs** compared to VPN.
- Supports **private VIF (Virtual Interface) to VPC**.
- Reliable for **high-throughput applications**.

---

43. What is Route 53?

Answer:
Route 53 is AWS's **scalable DNS (Domain Name System) service**.

Key points:

- Provides **domain registration and routing**.
- Supports **health checks, failover, and latency-based routing**.
- Can route traffic to **EC2, S3, ELB, or external endpoints**.

---

44. What is a VPN in AWS?

Answer:
AWS VPN enables **secure communication** between your on-premises network and your VPC over the internet.

Key points:

- Supports **Site-to-Site VPN** (IPSec) and **Client VPN**.
- Provides **encrypted tunnels** for data in transit.
- Often used with **Direct Connect** for hybrid architectures.

---

45. What is AWS Transit Gateway?

Answer:
Transit Gateway simplifies **VPC-to-VPC and on-premises connectivity**.

Key points:

- Acts as a **hub to connect multiple VPCs** and VPNs.
- Reduces the complexity of **VPC peering meshes**.
- Supports **centralized routing and policy enforcement**.

46. What is the AWS Free Tier?

Answer:
AWS Free Tier provides **limited access to AWS services** for free for **12 months** after account creation.

Key points:

- Helps new users **explore AWS without cost**.
- Includes free **EC2 hours, S3 storage, Lambda invocations, and more**.
- Some services also offer **always free usage limits**.

---

47. What is AWS Cost Explorer?

Answer:
Cost Explorer allows you to **visualize, understand, and manage AWS costs** over time.

Key points:

- Provides **graphs, reports, and filtering** by service, region, or tag.
- Helps identify **cost trends and anomalies**.
- Supports **budget planning and forecasting**.

---

48. What is AWS Budgets?

Answer:
AWS Budgets allows you to **set custom cost and usage budgets** and get alerts when thresholds are exceeded.

Key points:

- Supports **cost, usage, and reservation budgets**.
- Sends **email or SNS notifications** on exceeding limits.
- Helps enforce **cost discipline and governance**.

---

49. What is the difference between On-Demand, Reserved, and Spot Instances?

Type	Description	Use Case
On-Demand	Pay per hour/second	Short-term workloads, unpredictable traffic
Reserved	Prepay for 1 or 3 years	Steady-state workloads
Spot	Bid for unused EC2 capacity	Cost-sensitive, flexible workloads

---

50. What is the Total Cost of Ownership (TCO) Calculator?

Answer:
AWS TCO Calculator helps **estimate the cost savings** of migrating on-premises workloads to AWS.

Key points:

- Compares **hardware, software, maintenance, and labor costs**.
- Helps justify **cloud adoption to stakeholders**.
- Supports **various workloads like servers, storage, and databases**.

51. What is AWS Server Migration Service (SMS)?

Answer:
AWS SMS automates the **migration of on-premises servers** to AWS.

Key points:

- Supports **incremental replication**, reducing downtime.
- Works with **VMware vSphere, Hyper-V, and cloud servers**.
- Ideal for **lift-and-shift migrations**.

---

52. What is AWS Database Migration Service (DMS)?

Answer:
DMS helps **migrate databases** to AWS quickly and securely.

Key points:

- Supports **homogeneous (e.g., MySQL → MySQL)** and **heterogeneous (e.g., Oracle → Aurora)** migrations.
- **Minimal downtime** during migration.
- Can replicate ongoing changes to keep source and target in sync.

---

53. What is AWS Snowball?

Answer:
Snowball is a **physical data transport solution** for moving large amounts of data to AWS.

Key points:

- Handles **petabyte-scale data transfers**.
- Encrypted and tamper-resistant.
- Useful when **network transfer is too slow or expensive**.

---

54. What is AWS Storage Gateway?

Answer:
Storage Gateway connects **on-premises environments with AWS cloud storage**.

Key points:

- Supports **File, Tape, and Volume gateways**.
- Enables **hybrid cloud storage** with S3, Glacier, or EBS.
- Provides **low-latency local caching** for frequently accessed data.

---

55. What is Hybrid Cloud Architecture in AWS?

Answer:
Hybrid cloud combines **on-premises infrastructure with AWS cloud** to create a seamless environment.

Key points:

- Supports **disaster recovery, data backup, and bursting workloads**.
- Uses **Direct Connect, VPN, Storage Gateway, and VPC Peering**.
- Ideal for organizations **gradually migrating to cloud** or with regulatory requirements.

56. What is Amazon Redshift?

Answer:
Redshift is a **fully managed data warehouse** that allows fast querying and analysis of large datasets.

Key points:

- Columnar storage for **high-performance analytics**.
- Integrates with **BI tools like Tableau, QuickSight, and Power BI**.
- Scales easily using **Redshift clusters**.
- Supports **SQL-based queries**.

---

57. What is Amazon Athena?

Answer:
Athena is an **interactive query service** that allows you to analyze data in S3 using **standard SQL**.

Key points:

- Serverless, so no infrastructure management.
- Pay only for the **data scanned**.
- Works well for **ad-hoc analytics and quick reporting**.

---

58. What is Amazon EMR?

Answer:
EMR (Elastic MapReduce) is a **managed big data platform** for processing large datasets using **Hadoop, Spark, and Presto**.

Key points:

- Scales automatically based on workload.
- Supports **data processing pipelines and analytics**.
- Integrates with S3, Redshift, and RDS.

---

59. What is Amazon Kinesis?

Answer:
Kinesis is a **real-time streaming data service** for collecting, processing, and analyzing data.

Key points:

- Supports **Kinesis Data Streams, Firehose, Analytics, and Video Streams**.
- Ideal for **real-time dashboards, monitoring, and data pipelines**.

- Scales automatically to handle **millions of events per second**.

---

60. What is Amazon SageMaker?

Answer:
SageMaker is a **fully managed machine learning service** for building, training, and deploying ML models.

Key points:

- Provides **built-in algorithms, notebooks, and model hosting**.
- Supports **training at scale** and hyperparameter tuning.
- Integrates with **Lambda, S3, and other AWS services** for end-to-end ML workflows.

61. What is Amazon ECS?

Answer:
ECS (Elastic Container Service) is a **fully managed container orchestration service** that runs Docker containers on AWS.

Key points:

- Supports **EC2 launch type** (self-managed instances) and **Fargate launch type** (serverless).
- Integrates with **ALB, CloudWatch, IAM, and ECR**.
- Simplifies **deployment, scaling, and management** of containerized applications.

---

62. What is Amazon EKS?

Answer:
EKS (Elastic Kubernetes Service) is a **managed Kubernetes service** on AWS.

Key points:

- Runs standard Kubernetes without managing control planes.
- Integrates with **IAM, VPC, and CloudWatch**.
- Supports **scaling, monitoring, and secure container deployments**.

---

63. What is AWS Fargate?

Answer:
Fargate is a **serverless compute engine for containers** that works with ECS and EKS.

Key points:

- Removes the need to **provision or manage servers**.
- Automatically scales based on **container resource requirements**.

- Pay only for **CPU and memory used by containers**.

---

64. What is Amazon ECR?

Answer:
ECR (Elastic Container Registry) is a **fully managed Docker container registry**.

Key points:

- Stores, manages, and deploys container images.
- Integrated with **ECS, EKS, and Fargate**.
- Supports **image scanning for vulnerabilities**.

---

65. What is the difference between ECS and EKS?

Answer:

Feature	ECS	EKS
Type	AWS-native container service	Managed Kubernetes service
Management	Simplified AWS control	Full Kubernetes features
Complexity	Easier for AWS-only workloads	Supports multi-cloud and advanced orchestration
Use Case	Quick container deployments	Complex microservices with Kubernetes standards

66. How do you monitor ECS and EKS containers?

Answer:
Monitoring involves **collecting metrics, logs, and traces** for containers using AWS services.

Key points:

- **CloudWatch**: Monitors CPU, memory, network, and custom metrics.
- **CloudWatch Logs**: Captures container application logs.
- **AWS X-Ray**: Traces requests to identify **performance bottlenecks**.
- Can integrate with **Prometheus and Grafana** for advanced visualization.

---

67. What is AWS X-Ray?

Answer:
X-Ray helps **analyze and debug distributed applications**, including microservices running in ECS or EKS.

Key points:

- Shows **latency, errors, and request traces**.

- Helps identify **slow services or dependencies**.
- Supports **Lambda, API Gateway, and EC2/ECS services**.

---

68. How do you log container applications in AWS?

Answer:
Container logs can be captured using:

- **CloudWatch Logs**: Push container stdout/stderr logs.
- **Fluentd/Fluent Bit**: Forward logs to CloudWatch or S3.
- **EKS logging**: Use **Container Insights** for Kubernetes logs.

Key point: Centralized logging helps in **debugging, compliance, and monitoring**.

---

69. What is CloudWatch Container Insights?

Answer:
Container Insights provides **performance monitoring and resource utilization metrics** for ECS and EKS.

Key points:

- Tracks **CPU, memory, disk, and network metrics**.
- Provides **cluster-level, service-level, and pod-level views**.
- Simplifies **alerting and troubleshooting** for containerized workloads.

---

70. How do you handle container scaling in AWS?

Answer:
Container scaling can be done using:

- **ECS Service Auto Scaling**: Automatically adjusts the number of tasks.
- **Kubernetes HPA (Horizontal Pod Autoscaler)** in EKS: Scales pods based on CPU/memory or custom metrics.
- **Fargate** scales containers automatically with resource requirements.

Key point: Proper scaling ensures **cost optimization and high availability**.

71. What is the VPC CNI plugin in EKS?

Answer:
The **VPC Container Network Interface (CNI)** plugin allows Kubernetes pods in EKS to receive **native VPC IP addresses**.

Key points:

- Each pod gets an **ENI (Elastic Network Interface)** IP from the VPC subnet.
- Enables pods to **communicate directly with other AWS resources**.
- Simplifies **network security and routing** using VPC features.

---

72. What is an ALB Ingress Controller in EKS?

Answer:
ALB Ingress Controller allows Kubernetes services in EKS to be **exposed to the internet using an Application Load Balancer (ALB)**.

Key points:

- Supports **path-based and host-based routing**.
- Integrates with **Kubernetes Ingress resources**.
- Handles **SSL termination** and load balancing for container workloads.

---

73. What is a Service Mesh in AWS?

Answer:
A Service Mesh manages **microservices communication**, security, and observability.

AWS options:

- **AWS App Mesh**: Service mesh for ECS, EKS, and EC2.
- Supports **traffic routing, service discovery, encryption, and monitoring**.

Key point: Ideal for **microservices with multiple interdependent services**.

---

74. How do you expose ECS services publicly?

Answer:

- Use **Application Load Balancer (ALB)** or **Network Load Balancer (NLB)**.
- Configure the **service with a public-facing load balancer** in ECS.
- Use **security groups** to allow inbound traffic.

---

75. How do you implement pod-to-pod security in EKS?

Answer:

- Use **Kubernetes Network Policies** to restrict traffic between pods.
- Integrate with **AWS Security Groups for pods** (with VPC CNI).
- Combine with **IAM roles for service accounts** for fine-grained permissions.

76. What are IAM Best Practices in AWS?

Answer:
IAM (Identity and Access Management) best practices ensure secure access to AWS resources.

Key points:

- **Use least privilege** – grant only necessary permissions.
- Enable **MFA** (Multi-Factor Authentication) for all users.
- Avoid using **root account** for daily tasks.
- Use **roles** for EC2, Lambda, and cross-account access.
- Regularly **rotate access keys**.

---

77. What is AWS KMS (Key Management Service)?

Answer:
KMS manages **encryption keys** for securing data in AWS services.

Key points:

- Supports **envelope encryption** for S3, EBS, RDS, and Lambda.
- Enables **automatic key rotation** and auditing via CloudTrail.
- Can create **customer-managed keys (CMK)** or use **AWS-managed keys**.

---

78. How do you encrypt data at rest in AWS?

Answer:

- Use **S3 server-side encryption (SSE-S3, SSE-KMS)** for objects.
- Enable **EBS volume encryption** for EC2 instances.
- Enable **RDS encryption** for databases.
- Use **KMS for managing encryption keys**.

---

79. How do you encrypt data in transit in AWS?

Answer:

- Use **HTTPS/TLS** for communication with S3, API Gateway, and websites.
- Enable **SSL/TLS for RDS and Aurora**.
- Use **VPN or Direct Connect with IPSec** for on-premises connections.

---

80. How do you implement compliance in AWS?

Answer:
AWS provides **compliance frameworks and tools** to meet regulatory requirements.

Key points:

- Use **AWS Artifact** for access to compliance reports.
- Enable **CloudTrail** for auditing API calls.
- Use **Config and Security Hub** to monitor compliance with policies.
- Leverage **encryption, IAM, and VPC security controls**.

81. What is High Availability (HA) in AWS?

Answer:
HA ensures that applications **remain operational even if some components fail**.

Key points:

- Deploy resources across **multiple Availability Zones (AZs)**.
- Use **Elastic Load Balancers (ELB)** to distribute traffic.
- Combine with **Auto Scaling** for demand spikes.
- Example: Multi-AZ RDS deployment for database HA.

---

82. What is Fault Tolerance (FT) in AWS?

Answer:
FT ensures that applications **continue functioning without downtime** despite failures.

Key points:

- Redundant components across **AZs or regions**.
- Use **Elastic IPs, S3 replication, Multi-AZ databases**.
- System automatically **fails over to healthy resources**.

---

83. What is the difference between HA and FT?

Feature	High Availability	Fault Tolerance
Goal	Minimize downtime	Zero downtime
Redundancy	Some redundancy	Full redundancy
Cost	Moderate	Higher
Example	Multi-AZ EC2 + ELB	Active-active across multiple regions

---

84. What is AWS Disaster Recovery (DR)?

Answer:
DR ensures that applications and data can be **restored quickly after a disaster**.

Key strategies:

1. **Backup and Restore:** Store backups in S3/Glacier.
2. **Pilot Light:** Minimal resources always running, scale during DR.
3. **Warm Standby:** Reduced capacity environment always running.
4. **Multi-Region Active:** Fully replicated environment across regions.

---

85. What is Multi-AZ vs Multi-Region deployment?

Answer:

- **Multi-AZ:** Resources deployed across AZs **within the same region** for HA.
- **Multi-Region:** Resources deployed across **different regions** for FT and DR.

Key point:

- Multi-AZ is for **resiliency within a region**.
- Multi-Region is for **disaster recovery and global failover**.

86. What is Amazon API Gateway?

Answer:
API Gateway is a fully managed service that **creates, publishes, and manages APIs** at any scale.

Key points:

- Integrates with **Lambda, ECS, or HTTP endpoints**.
- Supports **REST, HTTP, and WebSocket APIs**.
- Provides **throttling, caching, monitoring, and authentication**.

---

87. What is AWS Step Functions?

Answer:
Step Functions is a **serverless orchestration service** that coordinates multiple AWS services into **workflows**.

Key points:

- Helps build **resilient, event-driven applications**.
- Visual workflow designer for **step-by-step process automation**.
- Integrates with **Lambda, ECS, Batch, and DynamoDB**.

---

88. What is Amazon EventBridge?

Answer:
EventBridge is a **serverless event bus** that connects **applications using events**.

Key points:

- Enables **event-driven architectures** without polling.
- Supports **custom and SaaS event sources**.
- Routes events to **Lambda, Step Functions, SQS, or SNS**.

---

89. How can S3 trigger Lambda functions?

Answer:
S3 can automatically invoke Lambda on events like **object creation, deletion, or modification**.

Key points:

- Enables **real-time processing** of uploaded files.
- Example: Image processing, log ingestion, or ETL pipelines.
- Combined with **EventBridge or SNS** for complex workflows.

---

90. What is the difference between SNS and SQS?

Answer:

Feature	SNS	SQS
Type	Pub/Sub messaging	Message queue
Delivery	Push to subscribers	Pull by consumers
Use Case	Notifications, fan-out	Decoupling microservices, asynchronous processing
Order	Not guaranteed	FIFO queues available

91. How can SageMaker integrate with serverless applications?

Answer:
SageMaker can be integrated with **Lambda, API Gateway, and Step Functions** to provide **real-time or batch ML inference**.

Key points:

- Lambda can invoke SageMaker **endpoint for predictions**.
- Step Functions orchestrate **complex ML workflows**.
- Serverless integration reduces **infrastructure management**.

---

92. What is Amazon Rekognition?

Answer:
Rekognition is a **computer vision service** for detecting objects, faces, text, and activities in images and videos.

Key points:

- Can be triggered by **S3 events or Lambda**.
- Supports **facial recognition, celebrity detection, and moderation**.
- Used in **security, media, and retail applications**.

---

93. What is Amazon Comprehend?

Answer:
Comprehend is a **natural language processing (NLP) service** for analyzing text.

Key points:

- Detects **entities, sentiment, key phrases, and language**.
- Can process **S3 files, streams, or Lambda events**.
- Useful for **chatbots, content analysis, and social media monitoring**.

---

94. What is Amazon Lex?

Answer:
Lex is a service to **build conversational chatbots** using voice or text.

Key points:

- Integrates with **Lambda for business logic**.
- Powers **Amazon Alexa** skills.
- Supports **multi-turn conversations and context management**.

---

95. What is Amazon Polly?

Answer:
Polly is a **text-to-speech service** that converts written text into **natural-sounding speech**.

Key points:

- Can be used with **Lambda for dynamic content generation**.
- Supports **multiple languages and voices**.
- Ideal for **voice-enabled applications, accessibility, and notifications**.

96. What is AWS Glue?

Answer:
Glue is a **fully managed ETL (Extract, Transform, Load) service** that prepares data for analytics.

Key points:

- Automates **data discovery, cataloging, and schema management**.
- Supports **Python (PySpark) scripts** for transformation.
- Integrates with **S3, Redshift, RDS, and Athena**.
- Serverless, so no infrastructure management.

---

97. What is Amazon Kinesis Data Streams?

Answer:
Kinesis Data Streams allows **real-time ingestion and processing of streaming data**.

Key points:

- Supports **high-throughput data pipelines**.
- Can integrate with **Lambda, Firehose, and Analytics**.
- Ideal for **log processing, clickstream analysis, and IoT applications**.

---

98. What is Amazon Kinesis Data Firehose?

Answer:
Firehose delivers **real-time streaming data to destinations** like S3, Redshift, or Elasticsearch.

Key points:

- Fully managed and **serverless**.
- Automatically **buffers, batches, and compresses** data.
- Can transform data with **Lambda functions** before delivery.

---

99. What is Amazon Athena in Data Analytics?

Answer:
Athena allows **ad-hoc SQL queries directly on S3 data**.

Key points:

- Serverless, pay per query scanned.
- Integrates with **Glue Data Catalog** for metadata management.
- Ideal for **quick analysis, reporting, and prototyping**.

---

100. What is AWS Lake Formation?

Answer:
Lake Formation simplifies **building and managing data lakes** in S3.

Key points:

- Centralizes **data ingestion, cataloging, and access control**.
- Integrates with **Athena, Redshift Spectrum, and Glue**.
- Supports **fine-grained security and compliance policies**.

==========

==========
==========

Specific Services

EC2 (Elastic Compute Cloud)

Question 1:

You have an EC2 instance hosting a critical application that occasionally experiences high CPU utilization. How would you **monitor and optimize performance** without over-provisioning?

Answer:

- Use **CloudWatch metrics** to monitor CPU, memory, disk, and network utilization.
- Set up **CloudWatch alarms** to notify or trigger Auto Scaling when thresholds are crossed.
- Enable **Enhanced Monitoring** for detailed OS-level metrics.
- Optimize the application (e.g., caching, multithreading) to reduce CPU usage.
- Consider **burstable instances (T3/T4)** if the workload is intermittent.

Question 2:

Explain how you would design a **highly available EC2 architecture** across multiple AZs for a web application.

Answer:

- Deploy EC2 instances across **multiple Availability Zones (AZs)**.
- Place instances behind an **Elastic Load Balancer (ALB or NLB)** to distribute traffic.
- Use **Auto Scaling Groups** to automatically launch or terminate instances based on load.
- Store **application state externally** (S3, DynamoDB, or RDS) for stateless EC2 instances.

- Enable **CloudWatch monitoring and alarms** for proactive issue detection.

Question 3:

How would you **migrate a running EC2 instance** to another region with minimal downtime?

Answer:

- Create an **AMI (Amazon Machine Image)** of the instance.
- Copy the AMI to the target region.
- Launch a new EC2 instance from the AMI in the target region.
- Update **DNS records** or Route 53 to point to the new instance.
- Optionally, use **Elastic IP or Elastic Load Balancer** to reduce downtime during cutover.

Question 4:

If your EC2 instance is not reachable via SSH, what **troubleshooting steps** would you perform?

Answer:

1. Check **security group rules** to ensure port 22 is open.
2. Verify the **network ACLs** allow inbound/outbound traffic.
3. Ensure the **EC2 instance has a public IP** (if connecting from internet).
4. Check **route tables and subnet configuration**.
5. Confirm the **SSH key pair** is correct and permissions on `.ssh/authorized_keys` are proper.
6. Check **EC2 instance health status** in the console.

==========

Question 5:

You have an EC2 instance serving a production web application. Traffic is unpredictable. How would you **ensure cost optimization while maintaining performance**?

Answer:

- Use **Auto Scaling Groups** to scale instances up/down based on demand.
- Select **right instance types**: mix of **On-Demand, Reserved, and Spot Instances**.
- Use **CloudWatch metrics and alarms** to trigger scaling.
- Consider **Elastic Load Balancer (ALB)** to balance traffic across instances.
- Enable **instance hibernation or termination protection** for critical instances.

Question 6:

You need to deploy a legacy application that requires a **specific OS and network setup**. How would you ensure **repeatable deployments**?

Answer:

- Create a **custom AMI** with the required OS and application pre-installed.
- Use **CloudFormation or Terraform** to automate instance creation and network setup.
- Use **User Data scripts** to perform initialization on boot.
- Integrate with **Auto Scaling** for scalability if needed.

Question 7:

Your EC2 instance needs to access other AWS services like S3 and DynamoDB securely. How would you **manage credentials safely**?

Answer:

- Assign an **IAM Role** to the EC2 instance with the required permissions.
- Avoid storing **AWS access keys on the instance**.
- Use **temporary credentials provided by the IAM Role**.
- Use **Instance Metadata Service (IMDSv2)** for secure retrieval of credentials.

Question 8:

You notice your EC2 instance is frequently hitting **EBS IOPS limits**. How would you optimize storage performance?

Answer:

- Switch to **provisioned IOPS SSD (io1/io2)** for high-performance workloads.
- Optimize **EBS volume size and type** based on throughput requirements.
- Enable **EBS optimization** on the EC2 instance.
- Consider **RAID 0 across multiple volumes** for increased throughput.
- Monitor using **CloudWatch metrics** for EBS read/write performance.

Question 9:

Your EC2 instance needs to be **reachable from only certain IPs**. How would you configure **network security**?

Answer:

- Use **Security Groups** to allow inbound traffic only from specific IP addresses or ranges.
- Optionally, use **Network ACLs** for an additional layer of subnet-level control.
- Consider **VPN or Direct Connect** for private connectivity.
- Monitor and log traffic using **VPC Flow Logs** for auditing.

Question 10:

You need to run a **highly critical application** on EC2 with **zero downtime during maintenance or patching**. How would you design it?

Answer:

- Deploy EC2 instances across **multiple AZs** for redundancy.
- Use **Auto Scaling Groups** with **rolling updates** to patch instances gradually.
- Place instances behind an **ALB** to route traffic away from instances being updated.
- Use **Elastic IPs** or **DNS failover** to maintain accessibility.
- Implement **CloudWatch alarms** to monitor health during updates.

Auto Scaling (EC2 Auto Scaling & ASG)

Question 1:

Your web application experiences unpredictable traffic spikes. How would you **configure auto scaling policies** to handle this while optimizing costs?

Answer:

- Use **Dynamic Scaling Policies** based on CloudWatch metrics like **CPU utilization, memory, or custom metrics**.
- Set **min, max, and desired instance counts** in the Auto Scaling Group (ASG).
- Use **step scaling** to scale in/out gradually rather than sudden jumps.
- Combine with **Scheduled Scaling** for predictable traffic patterns (e.g., daily peak hours).
- Monitor with **CloudWatch Alarms** to adjust policies as needed.

Question 2:

Explain a scenario where **scheduled scaling** is more appropriate than **dynamic scaling**.

Answer:

- If traffic patterns are **predictable**, such as:
    - E-commerce site traffic increasing every day at 9 AM.
    - Batch jobs running every night.
- Scheduled scaling ensures instances **scale up/down at specific times**, saving cost without reacting to sudden metrics fluctuations.

Question 3:

Your application stores session data locally on EC2 instances. How do you ensure **stateful applications work correctly** with auto scaling?

Answer:

- Avoid storing state on the instance itself.
- Use **external session storage**, e.g., **ElastiCache (Redis/Memcached)** or **DynamoDB**.
- Configure **sticky sessions** on the load balancer if unavoidable, though stateless design is preferred.
- Ensures **new instances can be added or removed without breaking sessions**.

Question 4:

How can you **prevent Auto Scaling from terminating the wrong EC2 instance** during a scale-in event?

Answer:

- Enable **Instance Protection** for critical instances.
- Configure **termination policies** in ASG (e.g., oldest launch configuration first).
- Use **tags** to identify instances that should not be terminated.
- Monitor **ASG activity history** for verification.

Question 5:

You want to reduce costs by using a mix of instance types. How would you **implement mixed instances in Auto Scaling**?

Answer:

- Use **Auto Scaling Groups with mixed instance policies**.
- Define **instance type flexibility** for the same ASG.
- Combine with **Spot Instances** for cost optimization and **On-Demand for critical workloads**.

- Configure **allocation strategies** to balance availability and cost.

Question 6:

Your Auto Scaling Group is frequently **launching and terminating instances rapidly**. What could be the reason and how do you fix it?

Answer:

- Possible reasons:
    - Improper **CloudWatch metric thresholds** (too sensitive).
    - Instances failing **health checks** repeatedly.
- Fix:
    - Adjust scaling **thresholds and cooldown periods**.
    - Ensure **instances are healthy and boot correctly**.
    - Check **user-data scripts and startup tasks** for failures.

Question 7:

Explain how **health checks** in Auto Scaling Groups work and why they are important.

Answer:

- ASG performs **EC2 and ELB health checks** to determine instance health.
- Unhealthy instances are **terminated and replaced automatically**.
- Ensures **high availability and reliability**.
- Can customize **grace periods** to allow application boot time.

Question 8:

You have a multi-tier application. How would you design **Auto Scaling for both web and application tiers**?

Answer:

- Create separate **ASGs for web and app tiers**.
- Web tier: Scale based on **request count, CPU, or network metrics**.
- App tier: Scale based on **queue length (SQS), CPU, or custom metrics**.
- Use **ALB** to route traffic between tiers.
- Ensure **dependencies and health checks** are properly configured.

Question 9:

How do you integrate **Auto Scaling with ELB** for seamless scaling?

Answer:

- Attach **ASG instances to the ELB**.
- ELB automatically **distributes incoming traffic** among healthy instances.
- Auto Scaling **adds/removes instances**, and ELB **updates the target group automatically**.
- Supports **rolling updates and zero downtime deployments**.

Question 10:

Your application has **predictable daily peaks** but also **unexpected spikes**. How would you combine **scheduled and dynamic scaling**?

Answer:

- Use **scheduled scaling** to handle predictable traffic (e.g., mornings and evenings).
- Enable **dynamic scaling** to handle unexpected spikes beyond scheduled limits.
- Use **cooldown periods** to avoid rapid scale-in/out.
- Monitor using **CloudWatch dashboards** and refine thresholds.

Disaster Recovery (DR) in AWS

Question 1:

Your primary AWS region goes down. How would you **ensure RTO (Recovery Time Objective) and RPO (Recovery Point Objective) targets** are met for a multi-tier application?

Answer:

- Use **multi-region deployment** for critical workloads.
- For RPO, replicate data with **cross-region replication (S3) or read replicas (RDS)**.
- For RTO, deploy **standby environments** in another region (pilot light or warm standby).
- Use **Route 53 with failover routing policy** to redirect traffic automatically.
- Test DR plans regularly to ensure **objectives are met**.

Question 2:

Compare **pilot light, warm standby, and multi-region active-active** strategies for DR. When would you use each?

Answer:

Strategy	Description	Use Case
Pilot Light	Minimal resources always running; scale up during disaster	Cost-effective for low-priority apps
Warm Standby	Scaled-down environment running continuously; can scale up	Medium RTO apps
Multi-Region Active-Active	Full-scale deployments in multiple regions	Zero-downtime/highly critical apps

---

Question 3:

How would you **replicate RDS databases across regions** to ensure DR readiness?

Answer:

- Enable **cross-region read replicas** for RDS.
- Optionally use **multi-AZ deployments** for HA within each region.
- Configure **automated backups** to S3 with **cross-region replication**.
- Use **Route 53 failover** to point to the standby region in case of outage.

Question 4:

Your S3 bucket storing critical backups becomes unavailable. How would you **recover your data quickly**?

Answer:

- Enable **S3 versioning** and **cross-region replication (CRR)**.
- Restore from a **replicated bucket in another region**.
- Use **Glacier or Glacier Deep Archive** for long-term backups, retrieving data based on urgency.
- Automate recovery testing to ensure **RTO targets** are met.

Question 5:

You need to **design a DR strategy for a multi-tier web application**. What AWS services and architecture would you use?

Answer:

- **Web Tier:** ALB + EC2 across multiple AZs, optionally multi-region.

- **App Tier:** ECS/EKS containers across AZs.
- **Database Tier:** RDS Multi-AZ + cross-region read replica.
- **Storage:** S3 + cross-region replication.
- **DNS Failover:** Route 53 with health checks.
- Automate **CloudFormation or Terraform** for rapid environment rebuild.

Question 6:

How would you **simulate a DR scenario** to test your recovery plan?

Answer:

- Stop services in the primary region or AZ.
- Failover DNS using **Route 53 failover policies**.
- Validate **application functionality** in standby region/AZ.
- Monitor **RTO and RPO metrics** to ensure objectives are met.
- Document lessons learned and **update DR plan** accordingly.

Question 7:

Your RDS database is mission-critical and cannot tolerate downtime. Which DR strategy would you choose and why?

Answer:

- **Multi-Region Active-Active** or **Multi-AZ with cross-region replicas**.
- Ensures **high availability and fault tolerance**.
- Allows **automatic failover** to another AZ or region.
- Minimal RTO and near-zero data loss (RPO).

Question 8:

You want to **reduce DR costs** for a non-critical application while maintaining acceptable RTO/RPO. What strategy would you use?

Answer:

- **Pilot Light**: Keep minimal infrastructure in standby region.
- Store backups in **S3 with cross-region replication**.
- Scale up resources only during a disaster event.
- Use **automated scripts** (CloudFormation/Terraform) to launch environment when needed.

Question 9:

How would you handle **DR for serverless applications** using Lambda and API Gateway?

Answer:

- Deploy **Lambda functions and API Gateway** in multiple regions.
- Use **S3 replication** for input data and DynamoDB global tables for database replication.
- Configure **Route 53 latency-based or failover routing** to redirect traffic.
- Test failover to ensure **stateless serverless apps continue functioning**.

Question 10:

Explain how you would **monitor and ensure DR readiness continuously** in AWS.

Answer:

- Enable **CloudWatch alarms and dashboards** for health metrics.
- Use **AWS Config** and **CloudTrail** for auditing DR configurations.
- Schedule **regular DR drills** to test RTO/RPO.
- Automate reporting to track **readiness status** of all components.
- Review **AWS Trusted Advisor** recommendations for fault tolerance.

IAM (Identity and Access Management)

Question 1:

You have a team of developers working on multiple projects. How would you **implement least privilege access** for them?

Answer:

- Create **IAM groups** based on project or role.
- Assign **policies with only necessary permissions** (least privilege).
- Use **IAM roles** for temporary access instead of long-term credentials.
- Monitor and **audit permissions regularly** using IAM Access Analyzer and AWS CloudTrail.
- Encourage **MFA** for all users for added security.

Question 2:

A Lambda function needs access to S3 and DynamoDB. How would you **securely assign permissions**?

Answer:

- Create an **IAM role specifically for the Lambda function**.
- Attach a **policy granting only the necessary S3 and DynamoDB permissions**.
- Avoid embedding AWS keys in Lambda code.
- Enable **CloudWatch logs** to monitor access and errors.

Question 3:

You want to **audit all IAM users and roles** for compliance. How would you do this?

Answer:

- Enable **CloudTrail** to log all IAM activity.
- Use **IAM Access Analyzer** to identify over-permissioned roles.
- Generate **IAM credential reports** to check active/inactive users, keys, and password usage.
- Remediate by **removing unused credentials and policies**.

Question 4:

A security breach occurs due to an old access key. How would you **mitigate and prevent future incidents**?

Answer:

- Immediately **rotate or delete compromised access keys**.
- Enable **MFA for sensitive users**.
- Implement **key rotation policies** automatically.
- Use **IAM roles for applications instead of long-term keys**.
- Monitor unusual activity using **CloudTrail and CloudWatch logs**.

Question 5:

Your organization needs to **grant cross-account access** to a partner. How would you implement it securely?

Answer:

- Create an **IAM role in your account** and allow the partner account to assume it.

- Attach **least-privilege policies** to the role.
- Use **STS (Security Token Service)** for temporary credentials.
- Monitor usage via **CloudTrail**.
- Avoid sharing root account credentials.

Question 6:

You want to **allow EC2 instances to access S3 securely** without embedding credentials. How would you do it?

Answer:

- Assign an **IAM role to the EC2 instance**.
- Attach a policy granting **specific S3 bucket access**.
- EC2 automatically retrieves **temporary credentials** via Instance Metadata Service (IMDSv2).
- Ensure **security groups and VPC endpoint policies** are configured correctly.

Question 7:

How would you **secure a multi-account AWS environment** using IAM?

Answer:

- Use **AWS Organizations** with **Service Control Policies (SCPs)**.
- Create **separate accounts for dev, test, and prod**.
- Use **IAM roles for cross-account access** instead of sharing credentials.
- Enforce **MFA and least privilege** policies across accounts.
- Monitor centralized logs via **CloudTrail and CloudWatch**.

Question 8:

How would you **handle temporary access for contractors**?

Answer:

- Create **IAM roles with temporary credentials** using STS.
- Set **expiration for the role** to limit access duration.
- Restrict access to **specific resources only**.
- Monitor access via **CloudTrail logs** and revoke immediately if needed.

Question 9:

Your application requires **fine-grained access control** for multiple S3 buckets. How would you implement it?

Answer:

- Use **IAM policies with bucket/resource-level permissions**.
- Consider **S3 bucket policies** to restrict access to specific prefixes.
- Use **conditions based on VPC, IP, or MFA** to enhance security.
- Enable **S3 access logging** for monitoring.

Question 10:

A developer accidentally escalates their IAM permissions. How would you **detect and mitigate privilege escalation**?

Answer:

- Enable **CloudTrail to monitor IAM API calls**.
- Use **AWS Config rules** to detect overly permissive policies.
- Immediately revoke or adjust **policies and roles**.
- Implement **preventive measures**: MFA, least privilege, and approval workflows for IAM changes.

S3 (Simple Storage Service)

Question 1:

You are storing large files (>5TB) in S3. How would you **optimize storage costs** while ensuring accessibility?

Answer:

- Use **S3 Intelligent-Tiering** to automatically move objects between **frequent and infrequent access tiers**.
- Enable **S3 Lifecycle Policies** to move old objects to **Glacier or Deep Archive**.
- Compress large files where possible.
- Enable **S3 Requester Pays** if multiple parties access the data.
- Monitor storage with **CloudWatch metrics** and **S3 Storage Lens**.

Question 2:

Explain a scenario where **S3 versioning, replication, and lifecycle policies** are crucial.

Answer:

- Versioning helps **recover accidentally deleted or overwritten files**.
- Cross-region replication ensures **DR readiness and high availability**.
- Lifecycle policies automatically **archive old data**, reducing storage costs.
- Example: Financial or healthcare data requiring **compliance and retention policies**.

Question 3:

How would you **secure sensitive files in S3** from public access while allowing specific applications to read/write?

Answer:

- Block **public access at bucket and account level**.
- Use **IAM roles or policies** granting only required permissions to applications.
- Enable **S3 encryption (SSE-S3, SSE-KMS)** for data at rest.
- Optionally use **VPC endpoints** to restrict access from outside your VPC.
- Monitor access with **S3 access logs and CloudTrail**.

Question 4:

You need to serve a static website from S3 with **high traffic spikes**. How would you handle it?

Answer:

- Enable **S3 static website hosting**.
- Use **CloudFront CDN** for caching and global delivery.
- Enable **Origin Shield** for protection against spikes.
- Configure **WAF (Web Application Firewall)** to prevent DDoS attacks.
- Monitor traffic using **CloudWatch and CloudFront logs**.

Question 5:

How would you **enable DR for S3 data** in another region?

Answer:

- Use **S3 Cross-Region Replication (CRR)** to replicate objects automatically.
- Enable **versioning** to preserve historical data.

- Optionally use **S3 Replication Time Control (RTC)** for SLA guarantees.
- Monitor replication status via **S3 metrics and notifications**.

Question 6:

Your application requires **real-time processing** of S3 object uploads. How would you design it?

Answer:

- Enable **S3 Event Notifications** for object creation events.
- Trigger **Lambda functions, SQS, or SNS** for downstream processing.
- Ensure **idempotent Lambda functions** to avoid duplicate processing.
- Monitor events using **CloudWatch logs and metrics**.

Question 7:

How do you **control access to S3 buckets for multiple teams** with different permissions?

Answer:

- Use **IAM policies and roles** for team-based access.
- Use **bucket policies** for resource-level control.
- Enable **prefix-based permissions** to isolate folders.
- Optionally use **AWS Organizations SCPs** for account-level restrictions.
- Monitor and audit access using **CloudTrail**.

Question 8:

A user accidentally deletes critical S3 objects. How would you **recover them**?

Answer:

- Enable **S3 versioning** to restore previous versions.
- Use **S3 object lock (Governance/Compliance mode)** to prevent deletion.
- Restore from **cross-region replicated bucket** if available.
- If archived in Glacier, initiate **restore requests** to recover data.

Question 9:

How would you **secure S3 data in transit and at rest** for compliance?

Answer:

- Enable **S3 SSE-S3, SSE-KMS, or SSE-C** for encryption at rest.
- Require **HTTPS (TLS)** for all data in transit.
- Use **VPC endpoints** to keep traffic private.
- Enable **bucket policies enforcing encryption**.
- Monitor access and encryption compliance using **AWS Config**.

Question 10:

How would you handle **S3 performance bottlenecks** for extremely high request rates?

Answer:

- Use **prefixing or S3 intelligent partitioning** to avoid hot partitions.
- Use **CloudFront CDN** to cache frequently accessed objects.
- Enable **S3 Transfer Acceleration** for faster uploads from remote locations.
- Monitor request rates and **optimize object naming patterns**.

VPC (Virtual Private Cloud)

Question 1:

You need to design a **multi-tier application VPC** with public and private subnets. How would you configure **routing, NAT, and security groups**?

Answer:

- Create **public subnets** for load balancers and **private subnets** for application and database servers.
- Attach an **Internet Gateway** to public subnets for internet access.
- Deploy a **NAT Gateway** in public subnets to allow private instances to access the internet securely.
- Configure **route tables**: public subnet route to IGW, private subnet route to NAT.
- Use **security groups** to allow only required inbound/outbound traffic.

Question 2:

How would you **connect an on-premises network securely** to your VPC?

Answer:

- Use **AWS VPN (Site-to-Site VPN)** or **AWS Direct Connect** for secure connectivity.

- Configure **Customer Gateway** on-premises and **Virtual Private Gateway** in AWS.
- Use **BGP routing** for dynamic route exchange (optional).
- Ensure **security group and NACL rules** allow required traffic.
- Monitor connectivity using **CloudWatch metrics**.

Question 3:

Explain a scenario where **VPC peering** is more appropriate than **Transit Gateway**.

Answer:

- **VPC Peering:** Best for **small-scale, direct connections** between two VPCs. Low cost, simple to manage.
- **Transit Gateway:** Best for **hub-and-spoke architecture** connecting multiple VPCs and on-premises networks. Scalable but more expensive.
- Example: Two VPCs need to share databases – use VPC peering. Ten+ VPCs and hybrid connections – use Transit Gateway.

Question 4:

Your EC2 instance in one subnet cannot reach another EC2 in a different subnet. How would you **troubleshoot connectivity issues**?

Answer:

1. Check **route tables** for correct subnet-to-subnet routing.
2. Verify **security group rules** allow traffic.
3. Check **network ACLs** for inbound/outbound rules.
4. Confirm **instance OS firewall** is not blocking traffic.
5. Test connectivity using **ping or telnet**.

Question 5:

You want to **isolate critical workloads** in a VPC. How would you design subnets and security?

Answer:

- Place workloads in **private subnets** with no internet access.
- Use **public subnets only for NAT Gateways and load balancers**.
- Apply **security groups per application/service** with least privilege rules.
- Use **VPC endpoints** for private access to S3, DynamoDB, etc.

- Optionally implement **subnet-level NACLs** for additional layer of security.

Question 6:

How would you **handle cross-account VPC connectivity**?

Answer:

- Use **VPC peering** between accounts (if few VPCs).
- Use **Transit Gateway** for multiple VPCs across accounts.
- Ensure **route tables** and **security groups** allow the required traffic.
- Use **IAM roles and resource policies** for secure access control.

Question 7:

How would you **enable internet access for private subnets** without exposing instances publicly?

Answer:

- Deploy a **NAT Gateway or NAT instance** in a public subnet.
- Configure **route tables in private subnets** to route traffic to the NAT.
- Keep private instances without **public IPs**.
- Use **security groups and NACLs** to restrict inbound/outbound traffic.

Question 8:

Explain **how you would design a highly available VPC architecture**.

Answer:

- Deploy **subnets across multiple AZs** for redundancy.
- Place **load balancers in public subnets** and **instances in private subnets**.
- Use **NAT Gateways in multiple AZs** for fault tolerance.
- Use **route tables and health checks** to route traffic only to healthy resources.
- Monitor with **CloudWatch** and implement **automatic failover** strategies.

Question 9:

How would you **secure VPC traffic between EC2 instances** for sensitive applications?

Answer:

- Use **security groups** to allow only required ports between instances.

- Enable **VPC Flow Logs** for monitoring traffic.
- Optionally use **encryption in transit** (TLS/HTTPS).
- Avoid exposing instances to public internet unnecessarily.

Question 10:

Your application needs to **access S3 privately** from your VPC. How would you design this?

Answer:

- Use a **VPC endpoint for S3**.
- Update **route tables** in private subnets to use the endpoint.
- Attach an **S3 bucket policy** to allow access only from the VPC endpoint.
- This ensures **private communication** without using the internet.

RDS (Relational Database Service)

Question 1:

How would you design **highly available and fault-tolerant RDS instances** across multiple AZs?

Answer:

- Enable **Multi-AZ deployment** for automatic failover.
- Use **primary DB in one AZ** and **standby in another AZ**.
- Amazon RDS automatically replicates **synchronous data** to the standby.
- Configure **automatic backups** and **snapshot retention**.
- Monitor failover events using **CloudWatch and RDS events**.

Question 2:

Your workload is **read-heavy**. How would you use **read replicas** effectively?

Answer:

- Create **read replicas** in the same or different region.
- Direct **read queries** from applications to replicas, reducing load on the primary DB.
- For cross-region DR, replicas can serve as **standby databases**.
- Monitor replication lag with **CloudWatch metrics**.
- Use **promotion** of a read replica in case of primary failure.

Question 3:

Explain how you would **migrate an on-premises database to RDS** with minimal downtime.

Answer:

- Use **AWS Database Migration Service (DMS)** for live migration.
- Set up **source and target replication**.
- Perform **initial load** and **ongoing replication** of changes.
- Test migration in **staging environment**.
- Cutover by redirecting applications to the **RDS endpoint**.

Question 4:

How would you **implement encryption at rest and in transit** for sensitive RDS data?

Answer:

- Enable **RDS encryption (SSE) using KMS** for data at rest.
- Use **SSL/TLS certificates** for encrypting data in transit.
- Enable **automatic key rotation** via KMS.
- Ensure **parameter groups** enforce SSL connections for clients.

Question 5:

Your RDS instance must **scale vertically and horizontally**. How would you design it?

Answer:

- Vertical scaling: Modify **instance type or storage type** to increase capacity.
- Horizontal scaling: Use **read replicas** for distributing read workloads.
- For write-heavy workloads, consider **sharding** or moving to **Aurora**.
- Monitor **CPU, memory, IOPS** using CloudWatch to trigger scaling decisions.

Question 6:

How would you **design RDS for cross-region disaster recovery**?

Answer:

- Enable **cross-region read replicas** for standby DB.
- Ensure replication lag is minimal to meet **RPO targets**.

- Use **Route 53 failover routing** to redirect applications to standby region.
- Test failover procedures regularly to ensure **RTO targets**.

Question 7:

Your RDS instance is experiencing **high CPU and slow queries**. How would you troubleshoot?

Answer:

- Enable **Enhanced Monitoring** and check CPU, memory, and disk I/O metrics.
- Review **slow query logs** and **performance insights**.
- Optimize database **indexes and queries**.
- Scale **vertically or horizontally** if needed.
- Consider **Aurora** for automatic scaling and better performance.

Question 8:

How would you **backup and restore RDS data** in case of accidental deletion?

Answer:

- Enable **automated backups** with a retention period.
- Take **manual snapshots** for critical points in time.
- Restore by creating a **new RDS instance** from a snapshot or backup.
- For cross-region DR, replicate snapshots to another region.

Question 9:

How would you **ensure compliance and auditability** for RDS access?

Answer:

- Enable **CloudTrail** to log all API calls.
- Use **IAM roles and policies** for fine-grained access control.
- Enable **RDS Enhanced Monitoring and audit logs**.
- Regularly review **database logs** and **user permissions**.

Question 10:

Your RDS instance must **support high availability and read scalability globally**. Which AWS service would you choose and why?

Answer:

- Use **Amazon Aurora Global Database**.
- Provides **multi-region replication** with low-latency reads.
- Automatic failover in one region does not affect read availability in other regions.
- Reduces RTO and RPO for global applications.

Lambda (Serverless Compute)

Question 1:

You have a Lambda function that processes **high-frequency events from S3**. Occasionally, events are dropped. How would you ensure **reliability and no data loss**?

Answer:

- Enable **S3 event notifications to invoke Lambda**.
- Use **SQS (Standard Queue) or SNS** as a buffer between S3 and Lambda to ensure retry.
- Configure **Lambda retries** and **dead-letter queues (DLQ)** for failed events.
- Monitor **CloudWatch metrics** for errors and throttling.

Question 2:

Your Lambda function requires access to a **private RDS database** in a VPC. How would you configure it?

Answer:

- Attach the Lambda function to the **VPC**.
- Assign the function to **private subnets** with **NAT Gateway** if internet access is needed.
- Configure **security groups** to allow Lambda traffic to the RDS instance.
- Monitor **VPC ENI usage** to avoid reaching limits (Lambda creates ENIs in VPC).

Question 3:

How would you **handle Lambda cold starts** for a latency-sensitive application?

Answer:

- Use **provisioned concurrency** to keep a set number of Lambda instances warm.
- Optimize function **package size** and **dependencies** to reduce initialization time.

- Use **VPC endpoints** to reduce latency if accessing AWS services privately.
- Monitor cold start metrics using **CloudWatch logs**.

Question 4:

Your Lambda function is timing out when processing large datasets. How would you **optimize execution**?

Answer:

- Increase **timeout configuration** (up to 15 minutes).
- Split large datasets into smaller **chunks and process asynchronously**.
- Use **S3, SQS, or DynamoDB streams** for event-driven batch processing.
- Optimize function **memory allocation**; higher memory increases CPU.
- Consider **Step Functions** for orchestrating long-running processes.

Question 5:

How would you **secure Lambda function access to other AWS services**?

Answer:

- Assign **IAM roles** with least privilege to the Lambda function.
- Avoid embedding **access keys** in the code.
- Use **KMS encryption** for sensitive data at rest and environment variables.
- Enable **VPC endpoints** for private service access.
- Monitor API calls via **CloudTrail**.

Question 6:

You need to **trigger multiple Lambda functions sequentially**. How would you design this?

Answer:

- Use **AWS Step Functions** to orchestrate multiple Lambdas in a workflow.
- Step Functions handles retries, error handling, and sequential execution.
- Monitor execution using **CloudWatch logs and Step Functions console**.
- Optionally, use **SNS or SQS** for event-driven asynchronous execution.

Question 7:

How would you **deploy Lambda functions across multiple environments** (dev, test, prod) securely?

Answer:

- Use **Lambda aliases and versions** to separate environments.
- Use **environment variables** for configuration (avoid hardcoding).
- Assign **IAM roles per environment** for secure access.
- Automate deployments using **CloudFormation, SAM, or Terraform**.

Question 8:

How do you **monitor Lambda performance and troubleshoot errors**?

Answer:

- Enable **CloudWatch logs** for each function invocation.
- Use **CloudWatch metrics**: Invocations, Duration, Errors, Throttles, IteratorAge.
- Configure **DLQ** for failed events (SQS or SNS).
- Use **X-Ray tracing** for detailed function execution insights.

Question 9:

Your Lambda function needs to **process millions of events daily**. How would you ensure **scalability**?

Answer:

- Lambda automatically scales horizontally; ensure **concurrent execution limits** are sufficient.
- Increase **service quotas** if needed.
- Use **SQS or Kinesis streams** to buffer and throttle incoming events.
- Monitor **throttles and errors** using CloudWatch.

Question 10:

How would you implement **versioning and deployment rollback** for Lambda functions?

Answer:

- Use **Lambda versions** for each deployment.
- Use **aliases** (e.g., dev, test, prod) pointing to specific versions.

- Rollback by **updating the alias** to point to a previous stable version.
- Automate deployment using **SAM, CloudFormation, or CI/CD pipelines**.

ECS (Elastic Container Service)

Question 1:

You need to **run multiple microservices in ECS** with high availability. How would you design it?

Answer:

- Use **ECS with Fargate** or **EC2 launch type** based on control needs.
- Deploy tasks across **multiple AZs** for fault tolerance.
- Use **Application Load Balancer (ALB)** to route traffic to different services.
- Define **task definitions** with CPU/memory requirements and container images.
- Enable **auto-scaling policies** for tasks based on metrics (CPU, memory, custom).

Question 2:

How would you **secure ECS tasks** that need access to S3 and DynamoDB?

Answer:

- Use **IAM roles for tasks (task roles)** to assign least privilege permissions.
- Avoid embedding credentials in container images.
- Enable **VPC endpoints** for private access to AWS services.
- Use **security groups** to restrict inbound/outbound traffic to tasks.

Question 3:

Your ECS tasks are **crashing frequently**. How would you troubleshoot?

Answer:

- Check **CloudWatch Logs** for container logs and errors.
- Inspect **task events** in ECS console for failures.
- Verify **CPU/memory limits** in task definitions are sufficient.
- Check **IAM permissions** if accessing AWS services.
- Review **Dockerfile or application dependencies** for runtime errors.

Question 4:

How would you **update a running ECS service** without downtime?

Answer:

- Use **blue/green deployment** with **CodeDeploy integration**.
- Update task definition to a new version.
- ECS service performs **rolling updates** based on deployment configuration.
- Monitor health of new tasks before terminating old tasks.
- Optionally, use **Canary deployments** to gradually route traffic.

Question 5:

You want ECS tasks to **scale automatically based on load**. How would you configure it?

Answer:

- Enable **Service Auto Scaling** in ECS.
- Define **target tracking policies** based on CPU, memory, or custom metrics.
- Optionally use **CloudWatch alarms** to trigger scaling.
- Ensure tasks are distributed across **multiple AZs** for availability.

Question 6:

Your ECS cluster is **running on EC2 instances** and needs to scale. How would you handle this?

Answer:

- Enable **Cluster Auto Scaling** to add/remove EC2 instances automatically.
- Ensure ECS agent is installed and running on instances.
- Set **desired capacity, min, and max instances** in Auto Scaling Group.
- Monitor **container instance CPU/memory utilization** to trigger scaling.

Question 7:

How would you **secure network access** for ECS tasks in a VPC?

Answer:

- Place tasks in **private subnets** to restrict public internet access.
- Use **security groups** to allow traffic only from specific sources.

- Use **VPC endpoints** for private access to AWS services.
- Optionally use **service mesh (App Mesh)** for secure intra-service communication.

Question 8:

You need **persistent storage** for ECS tasks. How would you implement it?

Answer:

- Use **EFS (Elastic File System)** for shared persistent storage.
- Mount EFS volumes in ECS task definitions.
- Ensure **security groups allow NFS access**.
- Use **EBS volumes** if storage is local to a single instance (EC2 launch type).

Question 9:

How would you **monitor ECS tasks and services** effectively?

Answer:

- Enable **CloudWatch Logs** for container stdout/stderr.
- Use **CloudWatch metrics**: CPU, memory, running tasks, service health.
- Optionally integrate **X-Ray** for distributed tracing in microservices.
- Configure **alarms and notifications** for failures or resource issues.

Question 10:

Your ECS service needs **zero-downtime deployments across multiple regions**. How would you design it?

Answer:

- Deploy ECS services in **multiple regions** behind **ALB or Global Accelerator**.
- Use **blue/green deployment** or **Canary deployment** for task updates.
- Use **Route 53 with latency-based routing** for global traffic distribution.
- Monitor health with **CloudWatch** and perform automated failover if needed.

EKS (Elastic Kubernetes Service)

Question 1:

You need to deploy a **highly available Kubernetes cluster** in AWS. How would you design EKS?

Answer:

- Create **EKS control plane** across multiple **AZs** (AWS manages control plane HA).
- Deploy **worker nodes (EC2 or Fargate)** in multiple AZs for redundancy.
- Use **private subnets** for worker nodes and **public subnets** for load balancers.
- Enable **Managed Node Groups** for automated scaling and updates.
- Use **IAM roles for service accounts (IRSA)** for secure pod access to AWS services.

Question 2:

Your application needs **auto-scaling based on CPU and memory**. How would you configure this in EKS?

Answer:

- Use **Horizontal Pod Autoscaler (HPA)** to scale pods based on CPU/memory or custom metrics.
- Enable **Cluster Autoscaler** to add/remove EC2 nodes as needed.
- Ensure pods have **resource requests and limits** defined.
- Monitor scaling events using **CloudWatch metrics** or **kubectl describe hpa**.

Question 3:

A pod is failing to start due to insufficient resources. How would you troubleshoot and fix it?

Answer:

- Check pod status: `kubectl describe pod <pod-name>`.
- Inspect events for **resource constraints or scheduling issues**.
- Ensure node group has **enough CPU/memory**.
- Adjust **pod resource requests/limits** or scale the cluster using **Cluster Autoscaler**.
- Monitor with **CloudWatch Container Insights** for node/pod metrics.

Question 4:

How would you **secure pod access to AWS resources** (e.g., S3, DynamoDB) without embedding credentials?

Answer:

- Use **IAM Roles for Service Accounts (IRSA)**.
- Assign IAM roles to specific Kubernetes service accounts.
- Attach policies granting **least privilege** access.
- Pods automatically assume role using **OIDC provider** configured with EKS.
- Monitor access with **CloudTrail**.

Question 5:

You want to **deploy a multi-tier application** in EKS with internal and external services. How would you design it?

Answer:

- Use **Deployments** for web, app, and DB tiers.
- Expose web tier via **ALB Ingress Controller** for internet access.
- Expose app tier using **ClusterIP or internal LoadBalancer**.
- Use **Secrets and ConfigMaps** for sensitive data and configuration.
- Apply **network policies** to restrict traffic between pods.

Question 6:

Your cluster needs **zero-downtime deployments** for critical services. How would you implement it?

Answer:

- Use **Deployment with rolling update strategy**.
- Define **maxUnavailable and maxSurge** parameters for controlled rollout.
- Optionally use **Canary or Blue/Green deployments** via service mesh (Istio/App Mesh).
- Monitor pod readiness with **readiness probes** before sending traffic.

Question 7:

How would you **handle persistent storage** for stateful applications in EKS?

Answer:

- Use **EBS volumes** for persistent storage per pod (stateful apps).
- Use **EFS or FSx** for shared storage across multiple pods.
- Define **PersistentVolume (PV) and PersistentVolumeClaim (PVC)**.

- Ensure proper **StorageClass** for dynamic provisioning.

Question 8:

You want to **monitor EKS cluster and pods**. What tools and metrics would you use?

Answer:

- Enable **CloudWatch Container Insights** for node/pod metrics.
- Use **kubectl top** for real-time resource usage.
- Monitor **pod restarts, CPU/memory usage, latency**.
- Optionally use **Prometheus + Grafana** for advanced monitoring.
- Set **CloudWatch alarms** for critical thresholds.

Question 9:

How would you **secure networking** in an EKS cluster?

Answer:

- Use **VPC CNI plugin** for pod networking within VPC.
- Apply **Security Groups for nodes** and **Network Policies for pods**.
- Use **private subnets** for worker nodes.
- Restrict API access using **AWS IAM and Kubernetes RBAC**.
- Enable **encryption in transit** for sensitive traffic.

Question 10:

Your EKS cluster is **cost-sensitive**. How would you optimize resource usage and cost?

Answer:

- Use **Fargate profiles** to run pods without managing EC2 instances.
- Enable **Cluster Autoscaler** to scale nodes automatically.
- Use **resource requests and limits** to prevent over-provisioning.
- Use **Spot instances** for non-critical workloads.
- Regularly monitor unused resources and scale down idle workloads.