AWS CLOUD & DEVOPS INTERVIEW – OUTLINE WITH NOTES

1. Profile / Introduction Questions

Q1. Tell me about your AWS experience. What services have you worked on?

Answer:
In my current role as a DevOps / Cloud Engineer, I work mainly on AWS infrastructure and automation. I'm hands-on with:

Compute & Networking: EC2, AMIs, Auto Scaling Groups, Load Balancers (ALB/NLB), VPC, Subnets, Internet Gateway, NAT Gateway,

Storage & Databases: EBS, EFS, S3 (lifecycle, versioning, encryption), RDS (MySQL/Postgres/Aurora), DynamoDB, ElastiCache (Redi

Monitoring & Logging: CloudWatch metrics, alarms, dashboards, CloudWatch Logs/Agents, CloudTrail, VPC Flow Logs.

Security & IAM: IAM Users/Groups/Roles, Policies, KMS, Secrets Manager, Parameter Store, ACM, WAF, Shield, Security Hub, GuardD

Containers & Serverless: ECR, ECS, Fargate, EKS basics, Lambda, API Gateway.

Networking & DNS: Route 53 hosted zones, records, routing policies, health checks.

Automation & IaC: Terraform, CloudFormation, user data, ASG launch templates.

DevOps & CI/CD: CodeCommit, CodeBuild, CodeDeploy, CodePipeline, sometimes Jenkins integrated with AWS.

I mainly focus on designing, provisioning, securing, monitoring, and optimizing AWS infrastructure for highly available and cost-effective

Notes (your project examples / scenarios):

Q2. What are your roles and responsibilities in your current AWS/DevOps role?

Answer:
My day-to-day responsibilities include:

Provisioning & Automation
- Creating and managing VPCs, subnets, EC2, RDS, S3, EKS/ECS using Terraform / CloudFormation.
- Writing user data and automation scripts for bootstrapping servers.

Monitoring & Reliability
- Setting up CloudWatch metrics, logs, dashboards, alarms, and integrating notifications via SNS/Email.
- Implementing backup and disaster recovery (EBS/RDS snapshots, cross-region backups).

Security & Compliance
- Managing IAM users/roles/policies (least-privilege).
- Enabling MFA, KMS encryption, ACM certificates, WAF/Shield, VPC flow logs.

Troubleshooting
- Debugging issues with instances, networking, DNS, load balancers, deployments.

Cost Optimization
- Right-sizing instances, using Reserved/Spot instances, S3 lifecycle, and cleaning unused resources.

DevOps / CI-CD
- Setting up CodePipeline/CodeBuild/CodeDeploy or Jenkins pipelines to deploy apps to EC2/ECS/Lambda.

I basically own the end-to-end lifecycle of cloud infrastructure: from design to implementation, monitoring, optimization and support.

Notes (your project examples / scenarios):

---

2. Cloud Computing Basics

Q3. What is cloud computing? Explain IaaS, PaaS, and SaaS with examples.

Answer:
Cloud computing is on-demand access to computing resources (servers, storage, databases, services) over the internet with pay-as-you

IaaS (Infrastructure as a Service)
- Provider offers virtual machines, storage, networking.
- You manage OS, runtime, apps.
- Examples: Amazon EC2, EBS, VPC.

PaaS (Platform as a Service)
- Provider manages infrastructure + runtime/platform; you just deploy code.
- Examples: AWS Elastic Beanstalk, RDS, AWS Lambda (often considered FaaS).

SaaS (Software as a Service)
- Ready-to-use applications delivered over the internet.
- Examples: Gmail, Salesforce, Office 365.

Notes (your project examples / scenarios):

Q4. What are public, private, and hybrid clouds?

Answer:
Public Cloud:
- Cloud resources shared across multiple customers over the public internet.
- Example: AWS, Azure, GCP.

Private Cloud:
- Cloud infrastructure dedicated to a single organization, usually on-prem or hosted.
- Example: VMware private cloud, OpenStack.

Hybrid Cloud:
- Combination of on-prem/private cloud + public cloud.
- Workloads can move or integrate between both using VPN/Direct Connect.

Notes (your project examples / scenarios):
---

3. EC2, EBS and Scaling

Q5. What is an EC2 instance? Explain basic EC2 terminology.

Answer:
Amazon EC2 is a resizable virtual machine in the cloud.

Key terms:
- AMI: Template image used to launch instances (OS + preinstalled software).
- Instance Type: Hardware config (vCPU, memory, network) like t3.medium, m5.large.
- Security Group: Virtual firewall at instance level; controls inbound/outbound.
- Key Pair: SSH/RDP authentication using public/private key.
- EBS Volume: Block storage attached to EC2.
- Region / Availability Zone: Geographic area / isolated data centers.

Notes (your project examples / scenarios):

Q6. How do you increase the size of an EBS volume for a Linux instance?

Answer (high-level steps):
- Stop or keep running the instance (EBS supports online resizing for many OSes).
- Go to EBS > Volumes, select the volume → Modify Volume → increase size.
- Wait for the volume modification to complete.
- Inside the instance:
  - Check disk: lsblk
  - Extend the partition (e.g., using growpart or fdisk).
  - Extend the filesystem (e.g., sudo resize2fs for ext4, xfs_growfs for XFS).
- Verify the new size with df -h.

In production, you take snapshots before resizing.

Notes (your project examples / scenarios):

Q7. What is the difference between vertical scaling and horizontal scaling in AWS?

Answer:
Vertical Scaling (Scale Up / Down):
- Increase/decrease size of a single instance (e.g., m5.large → m5.2xlarge).
- Simple but has hardware limits and may require downtime.

Horizontal Scaling (Scale Out / In):
- Add/remove multiple instances behind a Load Balancer.
- Implemented using Auto Scaling Groups.
- Better for high availability and handling large traffic.

In AWS, we prefer horizontal scaling for production, especially stateless apps.

Notes (your project examples / scenarios):

Q8. How do you backup and restore EC2 instances?

Answer:
Backup:
- Use EBS Snapshots (manual or automated via Snapshot Lifecycle Policy).
- Optionally create AMI from instance (includes root volume + attached EBS).

Restore:
- Create a new volume from snapshot and attach to instance, or
- Launch a new instance using the AMI.

For production, I normally use scheduled snapshot lifecycle policies with retention.

Notes (your project examples / scenarios):
---

4. AWS Global Infrastructure

Q9. Explain AWS regions, availability zones, edge locations, and local zones.

Answer:
Region:
- Geographical area (e.g., ap-south-1 for Mumbai). Contains multiple AZs.

Availability Zone (AZ):
- One or more data centers inside a region, isolated but interconnected.
- We design for multi-AZ high availability.

Edge Location:
- Used by CloudFront and Route 53 to cache content and serve DNS closer to users.

Local Zone / Wavelength Zone:
- Provide compute and storage closer to users/devices for ultra-low latency use cases.

Notes (your project examples / scenarios):
---

5. Storage: EBS, EFS, S3

Q10. Difference between EBS, EFS, and S3? When do you use each?

Answer:
EBS (Elastic Block Store):
- Block storage attached to a single EC2 instance (per AZ).

- Good for OS disks, databases, applications needing low-latency block storage.

EFS (Elastic File System):
- Managed NFS file system, can be mounted by multiple instances (Linux) across AZs.
- Good for shared storage between multiple EC2 instances or containers.

S3 (Simple Storage Service):
- Object storage for files/blobs (objects). Highly durable (11 9's).
- Good for backups, logs, static website hosting, data lakes.

Notes (your project examples / scenarios):

Q11. What are EBS volume types and which one would you choose?

Answer:
Common EBS volume types:
- gp2/gp3: General Purpose SSD, balance of price and performance; default choice.
- io1/io2: Provisioned IOPS SSD for high I/O, latency-sensitive workloads (e.g., critical databases).
- st1: HDD – Throughput optimized for big data, log processing.
- sc1: Cold HDD for infrequent access.

In most cases, I start with gp3 and move to io1/io2 if I need guaranteed IOPS.

Notes (your project examples / scenarios):

Q12. Explain S3 storage classes and lifecycle policies.

Answer:
Storage Classes:
- Standard: Frequent access, low latency.
- Standard-IA (Infrequent Access): Cheaper storage, retrieval fee; for less-frequent access.
- One Zone-IA: Like Standard-IA but in a single AZ.
- Intelligent-Tiering: Automatically moves data between frequent/IA based on access.
- Glacier / Glacier Deep Archive: Long-term backup and archival with higher retrieval time.

Lifecycle Policies:
- Rules to automatically move objects between classes or expire/delete objects.
- Example:
  - After 30 days → move from Standard to Standard-IA.
  - After 180 days → move to Glacier.
  - After 365 days → delete.

This helps optimize cost for S3.

Notes (your project examples / scenarios):

Q13. How can an EC2 instance in a private subnet access an S3 bucket?

Answer:
Two main ways:

VPC Endpoint for S3 (Gateway Endpoint):
- Create an S3 VPC endpoint.
- Update route tables to use the endpoint.
- Use bucket policies to allow access from that VPC/endpoint.
- No internet access required.

NAT Gateway + Internet Gateway:
- Route private subnet traffic to NAT Gateway, which goes out via IGW.

Best practice: Use S3 VPC Endpoint + IAM role on EC2 for access.

Notes (your project examples / scenarios):
---

6. IAM and Security

Q14. What is IAM? Explain Users, Groups, Roles, and Policies.

Answer:
AWS IAM (Identity and Access Management) controls who can do what in AWS.

- User: Represents a person/app that needs long-term access (access key/password).
- Group: Collection of users sharing the same policies/permissions.
- Role: Identity with a set of permissions assumed by AWS services or users. No long-term credentials.
- Policy: JSON document defining allow/deny permissions on AWS resources.

Best practice: Use roles instead of access keys, and follow least privilege.

Notes (your project examples / scenarios):

Q15. What is the difference between Security Groups and NACLs?

Answer:
Security Group (SG):
- Instance-level virtual firewall.
- Stateful: Response traffic is automatically allowed.
- Only allow rules (no explicit deny).

Network ACL (NACL):
- Subnet-level firewall.
- Stateless: Return traffic must be allowed explicitly.
- Supports allow and deny rules.
- Processed in rule order.

Usually, I use Security Groups as primary control and NACLs for extra subnet-level protection.

Notes (your project examples / scenarios):

Q16. How do you secure AWS environment? Mention some best practices.

Answer:
- Enable MFA for root and important IAM users.
- Do not use root account for daily operations.
- Use IAM roles instead of access keys; regularly rotate keys if used.
- Restrict Security Groups (no 0.0.0.0/0 for SSH/RDP in production).
- Encrypt data at rest using KMS (EBS, RDS, S3, etc.).
- Use ACM for SSL/TLS certificates and enforce HTTPS.
- Enable CloudTrail, Config, GuardDuty, Security Hub.
- Implement WAF in front of public web apps (SQL injection/XSS protection).
- Regularly review IAM policies and CloudTrail logs.

Notes (your project examples / scenarios):
---

7. Load Balancers and Auto Scaling

Q17. What are the types of Load Balancers in AWS and when do you use each?

Answer:
Application Load Balancer (ALB):
- Layer 7 (HTTP/HTTPS).
- Supports path-based, host-based routing, header-based rules.
- Integrates with ECS/EKS, Lambda, WAF.
- Used for web and microservices APIs.

Network Load Balancer (NLB):
- Layer 4 (TCP/UDP).
- Static IP, can handle millions of requests, very low latency.
- Used for high-performance or non-HTTP protocols (gRPC, custom TCP).

Gateway Load Balancer:
- For deploying/ scaling third-party network appliances (firewalls, IDS).

Classic Load Balancer (old):
- Legacy; generally replaced by ALB/NLB.

Notes (your project examples / scenarios):

Q18. How do Auto Scaling Groups work? What policies can you use?

Answer:
An Auto Scaling Group (ASG) manages a group of EC2 instances based on scaling policies:

You define:
- Min, Max, and Desired capacity.
- Launch template/config (AMI, instance type, SG, subnets).

It automatically:
- Launches instances when scaling out.
- Terminates instances when scaling in.

Scaling policies:
- Target tracking: Keep a metric (e.g., CPU at 60%).
- Step/simple scaling: If CPU > 80% for X minutes, add N instances.
- Scheduled scaling: Scale at specific times (e.g., office hours).
- Manual scaling: Manually change desired capacity.

Notes (your project examples / scenarios):
---

8. VPC & Networking

Q19. What is a VPC? Why do we create custom VPCs instead of using default?

Answer:
A VPC (Virtual Private Cloud) is your isolated virtual network in AWS.

We create custom VPCs because:
- We control CIDR ranges, IP planning.
- We design public and private subnets across multiple AZs.
- We control route tables, NAT gateways, VPN/direct connect, VPC endpoints.
- Better security and compliance: tightly controlled traffic flows.

Default VPC is okay for simple tests, but production needs a custom VPC design.

Notes (your project examples / scenarios):

Q20. Difference between Internet Gateway and NAT Gateway?

Answer:
Internet Gateway (IGW):
- Attached to a VPC to allow public internet access.
- Instances in public subnets with public IPs access internet via IGW.

NAT Gateway:
- Used by private subnet instances to access the internet (e.g., to download patches) without exposing them to inbound internet traffic.
- Private instances route 0.0.0.0/0 to NAT, NAT to IGW.

Notes (your project examples / scenarios):

Q21. What is VPC Peering?

Answer:
VPC Peering connects two VPCs so instances can communicate privately using private IPs.

- Can be between same or different regions and accounts.
- It's non-transitive – if VPC A peers with B, and B with C, A cannot talk to C through B.
- You must update route tables in both VPCs.

Used to integrate multiple VPCs within an org or between accounts.

Notes (your project examples / scenarios):

Q22. What are VPC Endpoints (Interface vs Gateway)?

Answer:
Gateway Endpoint:
- For S3 and DynamoDB.
- Targets a route in the route table.

Interface Endpoint:
- For many AWS services via PrivateLink.
- Creates an ENI (elastic network interface) with private IP in your subnet.

They allow private communication to AWS services without internet or NAT.

Notes (your project examples / scenarios):
---

9. Route 53 and DNS

Q23. What is Route 53? What are hosted zones and record types?

Answer:
Route 53 is AWS's DNS and domain registration service.

Hosted Zone:
- Container for DNS records for a domain (e.g., example.com).
- Types:
  - Public Hosted Zone: For public internet.
  - Private Hosted Zone: For VPC-internal DNS.

Common record types:
- A / AAAA: Maps name → IPv4/IPv6 address.
- CNAME: Alias one name to another.
- MX: Mail exchange.
- NS, SOA: Name servers and start of authority.
- Alias records: AWS-specific "A" like mapping to ELB, CloudFront, S3 website.

Notes (your project examples / scenarios):

Q24. What are Route 53 routing policies?

Answer:
- Simple: Single record, no special routing.
- Weighted: Split traffic based on weight (A/B testing, migration).
- Latency-based: Route to region with lowest latency.
- Failover: Health-based primary/secondary.
- Geolocation / Geoproximity: Route based on user location.
- Multivalue Answer: Return multiple healthy records.

I often use failover for DR, and weighted for gradual migrations.

Notes (your project examples / scenarios):
---

10. S3 – Extra Interview Points

Q25. How do you host a static website using S3?

Answer:
- Create an S3 bucket (usually same name as domain).
- Enable Static website hosting in bucket properties.
- Upload index.html and error.html.
- Make objects public (or use CloudFront + OAC for secure).
- Optionally use Route 53 to map domain to S3 website endpoint or CloudFront distribution.

In production, I normally put CloudFront in front for caching, HTTPS, and custom domain.

Notes (your project examples / scenarios):

Q26. What is S3 Versioning and Object Lock?

Answer:
Versioning:
- Keeps multiple versions of an object.
- Helps recover from accidental delete/overwrite.

Object Lock:
- WORM (Write Once Read Many) protection.
- Used for compliance – prevents deletion/modification for a set retention period.

Notes (your project examples / scenarios):
---

11. RDS / Databases

Q27. Why use RDS instead of installing database on EC2?

Answer:
RDS is a managed relational database service. Benefits:
- Automated backups, snapshots, retention.
- Multi-AZ for high availability.
- Automatic minor version upgrades, monitoring.
- Easy scale up/down.
- Integrated encryption and read replicas.

This reduces administrative overhead compared to self-managed DB on EC2.

Notes (your project examples / scenarios):
---

12. Monitoring: CloudWatch & CloudTrail

Q28. Difference between CloudWatch and CloudTrail?

Answer:
CloudWatch:
- For metrics, logs, dashboards, alarms.
- Example: CPU, memory (via agent), disk, application logs.

CloudTrail:
- For API call auditing (who did what, from where, and when).
- Helps in security audits and troubleshooting changes.

Notes (your project examples / scenarios):

Q29. What metrics do you usually monitor in CloudWatch?

Answer:
For EC2:
- CPUUtilization
- DiskReadOps / DiskWriteOps
- NetworkIn / NetworkOut
- StatusCheckFailed
- Memory, Disk space (via CloudWatch Agent)

For RDS:
- CPUUtilization, FreeableMemory
- FreeStorageSpace
- Read/WriteLatency
- DatabaseConnections

For ALB/NLB:
- RequestCount
- TargetResponseTime
- HTTPCode_ELB_4XX/5XX
- HealthyHostCount/UnHealthyHostCount

We build dashboards and alarms on these metrics.

Notes (your project examples / scenarios):
---

13. Infrastructure as Code – CloudFormation & Terraform

Q30. What is CloudFormation? How is it different from Terraform?

Answer:
CloudFormation:
- AWS-native Infrastructure as Code service.
- Uses YAML/JSON templates.
- Manages stacks, stack sets, change sets, drift detection.

Terraform:
- Multi-cloud IaC tool by HashiCorp.
- Uses HCL (HashiCorp Configuration Language).
- Can manage AWS + many other providers (GitHub, Datadog, etc).

In practice, I've used Terraform for multi-cloud / complex setups and CloudFormation for AWS-specific stacks or when teams prefer AW

Notes (your project examples / scenarios):
---

14. Security & Encryption Services

Q31. How do you encrypt data in AWS (EBS/RDS/S3)?

Answer:
EBS:
- Enable encryption using KMS key when creating volume or AMI.
- For unencrypted existing volumes, create encrypted snapshot → new encrypted volume.

RDS:
- Enable encryption at instance creation using KMS.
- For existing unencrypted, create snapshot and restore to encrypted instance.

S3:

- Enable default encryption (SSE-S3, SSE-KMS).
- Use bucket policies + KMS key policies for stricter control.

Notes (your project examples / scenarios):

Q32. What are AWS KMS, ACM, and Secrets Manager?

Answer:
KMS (Key Management Service):
- Manages encryption keys for EBS, RDS, S3, etc.

ACM (AWS Certificate Manager):
- Issues and manages SSL/TLS certificates for ELB, CloudFront, API Gateway.

Secrets Manager:
- Stores and rotates secrets (DB passwords, API keys).
- Provides secure retrieval via API/SDK/IAM roles.

Notes (your project examples / scenarios):
---

15. Containers: ECR, ECS, EKS, Fargate

Q33. What is ECR, ECS, and Fargate?

Answer:
ECR (Elastic Container Registry):
- Private Docker image registry.

ECS (Elastic Container Service):
- Container orchestration service.
- Manages tasks, services, clusters.

Fargate:
- Serverless compute engine for containers.
- You run containers without managing EC2 instances.

I typically build images → push to ECR → deploy to ECS (Fargate or EC2) behind an ALB.

Notes (your project examples / scenarios):

Q34. When would you use ECS vs EKS?

Answer:
ECS:
- AWS-native, simpler to manage.
- Good for teams that want managed orchestration without full Kubernetes complexity.

EKS:
- Managed Kubernetes control plane.
- Preferred if the organization standardizes on Kubernetes across environments.

Notes (your project examples / scenarios):
---

16. Serverless: Lambda, API Gateway, Cognito, DynamoDB

Q35. What is AWS Lambda? What are its limits?

Answer:
Lambda is a serverless compute service where you run code without managing servers.

Key points:

- Pay only for compute time used (per ms).
- Supports languages like Python, Node.js, Java, etc.
- Max execution timeout is typically 15 minutes.
- Scales automatically with number of requests.
- Use cases: API backends, event processing (S3, SQS, SNS), cron jobs.

Notes (your project examples / scenarios):

Q36. How do you trigger Lambda functions?

Answer:
Lambda can be triggered by:
- API Gateway (REST/HTTP APIs).
- S3 events (object create/delete).
- SQS messages.
- SNS topics.
- CloudWatch Events / EventBridge (schedules, events).
- DynamoDB Streams, Kinesis Streams, etc.

Notes (your project examples / scenarios):

Q37. What is Cognito and DynamoDB used for in serverless apps?

Answer:
Cognito:
- Manages user sign-up, sign-in, and access control.
- Provides JWT tokens, integrates with API Gateway and Lambda.

DynamoDB:
- Fully managed NoSQL key-value and document database.
- Serverless scaling, high performance; often used in Lambda-based apps.

Notes (your project examples / scenarios):
---

17. DevOps on AWS (CodeCommit, CodeBuild, CodeDeploy, CodePipeline)

Q38. How do you build a CI/CD pipeline in AWS?

Answer:
Using AWS DevOps tools:

CodeCommit / GitHub:
- Source control.

CodeBuild:
- Build and test code, create artifacts/Docker images.

CodeDeploy / ECS / Lambda:
- Deploy builds to EC2, ECS, or Lambda.

CodePipeline:
- Orchestrates stages (Source → Build → Test → Deploy).

Integrations:
- Use SNS/Slack for notifications.
- Store environment configs/secrets in SSM Parameter Store / Secrets Manager.

Alternatively, Jenkins can integrate with AWS (using IAM roles, CLI, SDK).

Notes (your project examples / scenarios):
---

18. Migration & Architecture

Q39. How would you design and deploy a 3-tier web application in AWS?

Answer (high-level):
Network:
- Custom VPC with multiple AZs.
- Public subnets: ALB, NAT gateways.
- Private subnets: App servers and DB.

Compute:
- Auto Scaling group of EC2 instances or ECS/EKS for app tier.

Database:
- RDS (Multi-AZ) for relational DB.

Storage:
- S3 for static content, backups, logs.

Security:
- Security groups per tier (web → app → DB).
- WAF + ACM HTTPS on ALB.

DNS & CDN:
- Route 53 + CloudFront if required.

Monitoring & Logging:
- CloudWatch, CloudTrail, VPC flow logs, alarms.

You can also propose serverless version using S3 + CloudFront + API Gateway + Lambda + DynamoDB.

Notes (your project examples / scenarios):

Q40. Describe a migration from on-prem to AWS.

Answer (template you can adapt):
Assessment:
- Analyze existing VMs (vCPU, RAM, disk, dependencies).

Strategy:
- Decide rehost (lift-and-shift) vs re-platform (move DB to RDS, app to ECS/Lambda).

Tools:
- Use AWS Application Migration Service / Database Migration Service.

Execution:
- Create target VPC, subnets, security, IAM.
- Migrate DB using DMS with continuous replication and cutover.
- Migrate app servers as AMIs/instances or containerize and deploy to ECS/EKS.

Post-migration:
- Switch DNS to AWS.
- Validate app, performance, monitoring, backups.

Notes (your project examples / scenarios):
