1. What is the role of IAM roles and policies?

IAM roles are used to grant temporary permissions to AWS resources, applications, or users. Unlike IAM users, roles do not have long-term credentials and are assumed by entities needing access. Policies define permissions and are attached to IAM users, groups, or roles.

---

2. Can you explain the Terraform plan and its purpose?

terraform plan is a command that previews the changes Terraform will apply before execution. It helps identify modifications, additions, or deletions in infrastructure, reducing the risk of unintended changes.

---

3. What is AWS Lambda, and how does it work?

AWS Lambda is a serverless compute service that runs code in response to events without managing servers. It executes code triggered by AWS services (e.g., S3, DynamoDB, API Gateway) or external sources.

---

4. How do you invoke a Lambda function, and where do you configure it?

Lambda functions can be invoked via:

- AWS Console
- AWS CLI (aws lambda invoke)
- SDKs (Boto3, Node.js, etc.)
- Event triggers (S3, SNS, SQS, DynamoDB, API Gateway)

You configure the function in the AWS Lambda Console, using environment variables, permissions (IAM roles), and triggers.

---

5. How does Lambda handle scaling and event-based invocations?

Lambda scales automatically by running multiple instances of the function in parallel. It can handle thousands of concurrent executions by provisioning more instances as requests increase.

---

6. What is Amazon CloudWatch, and have you configured any custom metrics?

Amazon CloudWatch is a monitoring service that collects and analyzes logs and metrics. Yes, custom metrics can be configured using the PutMetricData API or via CloudWatch Agent.

---

7. What metrics are available on your CloudWatch dashboard?

Some key metrics include:

- **CPU Utilization**
- **Memory Usage**
- **Network Traffic**
- **Disk I/O**
- **Lambda Invocations & Errors**

---

8. How do you configure CPU utilization on your CloudWatch dashboard?

1. Go to CloudWatch → Metrics
2. Select **EC2 Metrics** → Choose **CPUUtilization**
3. Click **Create Alarm** and set thresholds

---

9. How do you attach an SSL certificate to an S3 bucket?

S3 itself doesn't support SSL directly. You must use:

1. **CloudFront** (CDN) with an SSL certificate from AWS Certificate Manager (ACM).
2. **Application Load Balancer (ALB)** if routing traffic via EC2.

---

10. What type of encryption have you implemented in your project?

Encryption methods:

- **S3 Encryption:** SSE-S3, SSE-KMS
- **EBS Encryption:** KMS-managed keys
- **RDS Encryption:** Transparent Data Encryption (TDE)
- **IAM Credentials Encryption:** KMS

---

11. If an S3 bucket has a read-only policy, can you modify objects in the bucket?

No, unless you have additional permissions such as s3:PutObject or s3:DeleteObject.

---

12. Why did you choose Terraform over Boto3 for infrastructure provisioning?

- **Terraform** is declarative and manages infrastructure as code (IaC).
- **Boto3** is imperative and better for scripting AWS API calls.
- Terraform ensures state management and idempotency.

---

13. What is a Content Delivery Network (CDN), and how does it work?

A CDN, like **Amazon CloudFront**, caches content in edge locations worldwide to improve latency and load times.

---

14. Have you created a Jenkins pipeline for your project?

Yes, using **Jenkinsfile**, I implemented CI/CD pipelines for automated builds, testing, and deployments.

---

15. How do you attach policies to IAM users, either individually or by group?

- **Individually:** Attach a policy directly to the user.
- **By Group:** Create a group with the required policy and add users to it.

---

16. What type of deployment strategies are you using in your project?

- **Blue-Green Deployment**
- **Canary Deployment**
- **Rolling Updates**
- **Feature Flags**

---

17. Have you used any tools to create customized Amazon Machine Images (AMIs)?

Yes, using **Packer** to automate AMI creation.

---

18. What is connection draining, and how does it work?

It allows active connections to finish before terminating an instance from an **ELB**. Prevents connection drops during scaling or maintenance.

---

19. How does an Elastic Load Balancer (ELB) distribute traffic?

- **Round Robin** (default for ALB)
- **Least Outstanding Requests** (for NLB)
- **IP Hash**

---

20. What is auto-scaling, and how does it work?

Auto-scaling adjusts compute resources dynamically based on load by scaling up/down EC2 instances.

---

21. Can you describe the different types of Load Balancers and provide examples?

1. **Application Load Balancer (ALB)** - HTTP/HTTPS, Layer 7
2. **Network Load Balancer (NLB)** - TCP/UDP, Layer 4
3. **Classic Load Balancer (CLB)** - Deprecated

---

22. What is the maximum runtime for a Lambda function?

15 minutes (900 seconds).

---

23. What is the maximum memory size for a Lambda function?

10 GB.

---

24. How can you increase the runtime for a Lambda function?

Modify the function's **timeout setting** in AWS Lambda configuration.

---

25. What automations have you performed using Lambda in your project?

- **Automated S3 data processing**
- **Event-driven EC2 management**
- **Auto-remediation scripts**

---

26. Why did you choose Terraform over Boto3 for infrastructure provisioning?

(Same as Q12) Terraform provides better infrastructure state management and declarative syntax.

---

27. What modules have you used in your Lambda function?

- **Boto3** (AWS SDK for Python)
- **Requests** (HTTP requests)
- **Pandas** (Data processing)

---

28. Have you created an SNS topic for your project?

Yes, for sending notifications to Lambda, SQS, and email subscribers.

---

29. If you've exhausted IP addresses in your VPC, how would you provision new resources?

1. Add **secondary CIDR blocks** to the VPC.
2. Create additional **subnets** in another region.

---

30. What is Groovy, and how is it used in Jenkins?

Groovy is a scripting language used in **Jenkins Pipelines** for automation.

---

31. Why do you use Groovy in Jenkins, and where do you save Jenkins files?

- It provides flexibility in defining CI/CD workflows.

- Jenkinsfile is stored in the **Git repository**.

---

32. What is Ansible, and what is its purpose?

Ansible is an **IaC tool** for **configuration management and automation**.

---

33. What language do you use in Ansible?

YAML (Playbooks).

---

34. Where do you run Terraform code, remotely or locally?

Both:

- **Locally** (for testing)
- **Remotely** (using Terraform Cloud or CI/CD pipelines).

---

35. What is the purpose of access keys and secret keys in AWS?

They authenticate AWS CLI/API requests.

---

36. What are Terraform modules, and have you used any in your project?

Reusable Terraform configurations; yes, I've used **VPC, EC2, IAM modules**.

---

37. What environments have you set up for your project?

Development, SIT, UAT, and Production.

---

38. Do you use the same AWS account for all environments?

No, we use **separate AWS accounts** for security.

---

39. Do you have separate Jenkins servers for each environment?

Yes, dedicated Jenkins servers for different environments.

---

40. Where do you write and save your Lambda function code?

In AWS Console, S3, or Git repository with CI/CD pipelines.

50 Questions & Answers

1. What is DevOps, and why is it important?
DevOps is a software development methodology that combines development (Dev) and operations (Ops) to enhance collaboration, automate processes, and improve software delivery speed and reliability. It is important because it helps organizations deliver high-quality software faster, improve deployment frequency, and reduce failures.

2. How does DevOps differ from Agile and traditional IT operations?

- Agile focuses on iterative development and collaboration within development teams, while DevOps extends Agile principles to operations.
- Traditional IT operations emphasize stability and control, often leading to slower releases, whereas DevOps aims for speed, automation, and continuous delivery.

3. What are the key benefits of DevOps?

- Faster software delivery
- Improved collaboration
- Higher deployment frequency
- Reduced failures and downtime
- Enhanced security and compliance

4. What are some key DevOps tools you have worked with?

- CI/CD: Jenkins, GitLab CI/CD, Azure DevOps
- Configuration Management: Ansible, Chef, Puppet
- Containerization: Docker, Kubernetes
- Monitoring: Prometheus, Grafana, ELK Stack
- Infrastructure as Code: Terraform, CloudFormation
- Version Control: Git, Bitbucket

5. Explain the DevOps lifecycle.

1. **Plan** – Requirements and planning.

2. **Develop** – Code development and version control.
3. **Build** – Continuous integration and build automation.
4. **Test** – Automated testing.
5. **Release** – Deployment planning.
6. **Deploy** – Automated deployment.
7. **Operate** – Monitoring and management.
8. **Monitor** – Performance and security monitoring.

6. What are some common Linux commands used in DevOps?

- ls, cd, pwd, cp, mv, rm – File operations
- ps, top, htop – Process management
- grep, sed, awk – Text processing
- chmod, chown – Permissions
- tar, zip, scp, rsync – Archiving and transfer
- netstat, curl, ping – Networking

7. How do you schedule a cron job in Linux?
Use crontab -e to edit the cron jobs. Example: 0 2 * * * /path/to/script.sh (Runs at 2 AM daily)

8. What is the difference between grep, sed, and awk?

- grep – Searches for patterns in text.
- sed – Stream editor for modifying text.
- awk – Text processing and reporting.

9. How do you check system performance in Linux?

- top, htop – Monitor CPU and memory usage.
- vmstat, iostat – Check system statistics.
- df -h, du -sh – Disk usage.
- free -m – Memory usage.

10. How would you write a Bash script to automate a task?

```
#!/bin/bash
echo "Starting backup"
tar -czf backup.tar.gz /path/to/files
echo "Backup completed"
```

11. What is Git, and why is it used in DevOps?
Git is a distributed version control system used for tracking code changes, collaboration, and integration with CI/CD.

12. Explain the difference between Git merge and Git rebase.

- merge – Combines branches while keeping commit history.

- rebase – Moves commits to a new base, making history linear.

13. What is a Git branching strategy?
Branching strategies like Git Flow, GitHub Flow, and Trunk-based development define how teams manage branches in Git.

14. How do you resolve merge conflicts in Git?
Use git status to check conflicts, manually edit files, then git add and git commit to resolve.

15. Explain the purpose of a Git commit, push, pull, and fetch.

- commit – Saves changes locally.
- push – Uploads commits to a remote repository.
- pull – Fetches and merges changes from the remote.
- fetch – Downloads changes without merging.

16. What is Continuous Integration (CI), and why is it important?
CI automates code integration, detects errors early, and ensures a stable codebase.

17. What is Continuous Deployment (CD), and how does it differ from Continuous Delivery?

- **Continuous Deployment** – Deploys automatically after successful tests.
- **Continuous Delivery** – Requires manual approval before deployment.

18. How would you set up a CI/CD pipeline?
Use Jenkins, GitLab CI/CD, or Azure DevOps to automate build, test, and deployment stages.

19. What are some common CI/CD tools, and how do they compare?

- Jenkins – Highly customizable.
- GitLab CI/CD – Built into GitLab.
- Azure DevOps – Integrates well with Microsoft tools.

20. How do you handle rollback in a CI/CD pipeline?
Use versioned deployments, feature flags, or rollback scripts.

21. What are the benefits of cloud computing in DevOps?

- Scalability
- Cost efficiency
- Disaster recovery

22. What are the main services provided by AWS for DevOps?

- CodePipeline, CodeBuild, CodeDeploy, CloudFormation, ECS, EKS.

23. What is the difference between EC2, ECS, and EKS in AWS?

- **EC2** – Virtual machines.
- **ECS** – Container management.
- **EKS** – Managed Kubernetes.

24. What is IAM, and how does it help in security?
IAM (Identity and Access Management) controls access permissions.

25. How would you deploy an application in the cloud?
Use Kubernetes, Terraform, or AWS services like Elastic Beanstalk.

26. What is Docker, and how does it work?
Docker is a containerization platform that packages applications with dependencies.

27. Explain the difference between a Docker image and a Docker container.

- **Image** – A template.
- **Container** – A running instance of an image.

28. What is Kubernetes, and why is it used?
Kubernetes orchestrates containerized applications.

29. What is a Kubernetes Pod, and how does it work?
A Pod is the smallest deployable unit in Kubernetes.

30. What is the difference between a Deployment and a StatefulSet in Kubernetes?
Deployments manage stateless applications; StatefulSets manage stateful applications.

31. What is Infrastructure as Code (IaC), and why is it important?
IaC automates infrastructure provisioning using code, ensuring consistency and reducing manual errors.

32. How does Terraform differ from Ansible?
Terraform is declarative and manages infrastructure, while Ansible is procedural and used for configuration management.

33. What are Terraform modules, and how do they work?
Terraform modules are reusable configurations that simplify infrastructure management.

34. Explain the difference between terraform plan and terraform apply.

- terraform plan – Shows changes before applying.
- terraform apply – Executes the changes.

35. How would you manage secrets in Terraform?
Use Vault, AWS Secrets Manager, or environment variables to store secrets securely.

36. What is monitoring in DevOps, and why is it necessary?
Monitoring tracks system health, ensuring performance and availability.

37. What is the difference between Prometheus and Grafana?

- **Prometheus** collects metrics.
- **Grafana** visualizes metrics.

38. How would you set up an alert in Prometheus?
Define alert rules in Prometheus and configure Alertmanager for notifications.

39. What is the ELK Stack, and what is it used for?
ELK (Elasticsearch, Logstash, Kibana) is used for log management and analysis.

40. How do you debug performance issues in a production system?
Analyze logs, monitor metrics, and identify bottlenecks.

41. What are some best practices for securing CI/CD pipelines?
Use role-based access control (RBAC), secret management, and automated security scans.

42. What tools do you use for security scanning in DevOps?
Snyk, SonarQube, Aqua Security.

43. How would you handle secrets management in DevOps?
Use tools like HashiCorp Vault, AWS Secrets Manager, or Kubernetes Secrets.

44. How do you ensure compliance with security policies in DevOps workflows?
Implement automated compliance checks and audits.

45. How would you handle a failing deployment?
Rollback to the last stable version or fix and redeploy.

46. What steps would you take if a Kubernetes pod is stuck in a crash loop?
Check logs (kubectl logs), describe the pod (kubectl describe pod), and fix the issue.

47. How do you troubleshoot high CPU usage in a cloud environment?
Use top, htop, or cloud monitoring tools to identify and optimize resource consumption.

48. How would you optimize a slow-running CI/CD pipeline?
Parallelize builds, use caching, and optimize test execution.

49. Tell me about a time when you solved a major DevOps challenge.
Provide a scenario where you improved performance, security, or deployment efficiency.

50. What is HPA (Horizontal Pod Autoscaler)?
HPA automatically scales Kubernetes pods based on CPU or memory usage.