1. General DevOps Questions

1. What is DevOps and why is it needed?

DevOps is a culture and set of practices that combine development (Dev) and operations (Ops) to enable faster, more reliable software delivery. It aims to automate workflows, shorten feedback loops, and improve collaboration. The goal is continuous integration, continuous delivery, and faster innovation cycles.

2. How is DevOps different from Agile?

Agile focuses on iterative software development and customer feedback, while DevOps extends Agile principles to deployment and operations. Agile manages *what* to build; DevOps manages *how* to deploy and run it efficiently. DevOps bridges development, QA, and operations teams for continuous delivery.

3. What are the key benefits of DevOps?

DevOps offers faster release cycles, improved collaboration, reduced failure rates, and quicker recovery times. It leads to enhanced product quality, efficient resource utilization, and better alignment between business goals and technology outcomes.

4. What are common DevOps tools?

Popular tools include Git (VCS), Jenkins (CI/CD), Docker (containers), Kubernetes (orchestration), Ansible/Puppet (configuration), and Prometheus/Grafana (monitoring). Each tool automates a stage in the DevOps lifecycle to ensure smooth delivery.

5. Explain the DevOps lifecycle.

The DevOps lifecycle includes phases like Code, Build, Test, Release, Deploy, Operate, and Monitor. It forms a continuous loop enabling automation and feedback across the pipeline for faster and more reliable releases.

6. What is CI/CD?

CI/CD stands for Continuous Integration and Continuous Delivery/Deployment. CI automates code integration and testing, while CD automates deployment to production. Together, they enable faster, more reliable, and error-free software delivery.

7. What is version control and why is Git popular?

Version control tracks code changes, enabling rollback, collaboration, and history tracking. Git is distributed, meaning every developer has a local copy of the full repository, ensuring faster collaboration and offline work.

8. What is Infrastructure as Code (IaC)?

IaC is the practice of managing infrastructure using code and automation tools. It enables reproducibility, consistency, and scalability in infrastructure provisioning using tools like Terraform, Ansible, or CloudFormation.

9. What are containers?

Containers package an application with its dependencies and configurations into isolated units. They ensure consistency across environments and improve scalability and resource efficiency compared to traditional VMs.

10. Explain blue-green deployment.

Blue-Green deployment maintains two environments — Blue (current) and Green (new). Traffic is switched to Green after successful testing, allowing instant rollback if issues occur. It minimizes downtime during releases.

🔧2. Scenario-Based & Real-World Questions

11. A Jenkins build fails after a code commit — how will you troubleshoot?

First, review the Jenkins console output and check logs for syntax or dependency errors. Validate environment variables and credentials. Ensure the SCM webhook triggered the job correctly. Fix errors locally, re-run the job, and document the fix.

12. How would you handle a failed deployment in Kubernetes?

Check the pod status using `kubectl get pods`, describe failed pods, and review logs. If configuration or image issues are found, roll back using `kubectl rollout undo`. Implement readiness probes and use Helm for versioned releases.

13. How do you automate infrastructure provisioning?

Use IaC tools like Terraform or CloudFormation to define cloud resources as code. Store configurations in version control and apply CI/CD for environment consistency and scalability.

14. How to secure credentials in Jenkins?

Use Jenkins Credentials Manager to store secrets, tokens, and SSH keys securely. Reference them with environment variables or credential IDs instead of hardcoding them in scripts.

15. How do you integrate testing in a CI/CD pipeline?

Include unit and integration test stages in Jenkins or GitHub Actions. Use tools like JUnit, Selenium, or PyTest, and configure automated triggers after each code commit.

16. How would you scale Docker containers automatically?

Use orchestration tools like Kubernetes or Docker Swarm with auto-scaling policies. Scale pods based on CPU/memory thresholds defined in the Horizontal Pod Autoscaler.

17. How do you manage secrets in a containerized environment?

Use tools like AWS Secrets Manager, HashiCorp Vault, or Kubernetes Secrets. Ensure secrets are not baked into Docker images or version control.

18. How do you ensure high availability in production?

Use load balancers (ALB/Nginx), auto-scaling groups, and multi-AZ deployments. Implement health checks, failover strategies, and redundant components to prevent downtime.

19. How do you handle rolling updates in Kubernetes?

Use `kubectl rollout` or Helm upgrade commands with zero-downtime settings. Monitor rollout status, define readiness probes, and enable rollback in case of failure.

20. How do you perform zero-downtime deployments?

Use blue-green or rolling deployment strategies with health checks. Route traffic dynamically using load balancers while gradually replacing old instances.

21. How do you back up Jenkins configuration?

Backup the `JENKINS_HOME` directory regularly, which contains job configs, plugins, and credentials. Use version control or automation for scheduled backups.

22. How do you handle pipeline rollback in Jenkins?

Use versioned artifacts and maintain rollback scripts in Jenkins pipelines. In case of failure, redeploy the previous stable build automatically using tags or build history.

23. How do you implement security scanning in a CI/CD pipeline?

Integrate tools like Trivy or SonarQube for image and code scanning. Automate vulnerability detection before deployment and fail builds if critical issues are found.

24. How do you optimize Jenkins jobs for performance?

Use pipeline stages in parallel, enable agent reusability, cache dependencies, and run builds in Docker. Monitor Jenkins master resource usage and distribute load via agents.

25. How would you monitor API performance?

Use API monitoring tools like Postman monitors, Datadog, or New Relic. Track response time, error rates, and throughput; set alerts for performance degradation.

26. How do you implement cost optimization in AWS?

Use AWS Trusted Advisor, right-size instances, enable auto-scaling, and implement S3 lifecycle policies. Schedule non-production resources to shut down during off-hours.

27. How do you ensure compliance in DevOps pipelines?

Use policy-as-code tools like Open Policy Agent or Checkov to enforce standards. Automate compliance checks during CI/CD builds to meet security benchmarks.

28. How do you integrate monitoring with Prometheus and Grafana?

Deploy Prometheus to collect metrics and Grafana to visualize them. Use exporters for system, container, and app metrics, and set up alerts for performance anomalies.

29. How do you manage logs across distributed systems?

Use centralized logging solutions like ELK Stack (Elasticsearch, Logstash, Kibana) or AWS CloudWatch Logs. Configure log rotation and alerts for critical patterns.

30. How do you detect configuration drift?

Use tools like Terraform plan, Ansible diff, or AWS Config to compare actual infrastructure with code. Automate periodic audits to ensure consistency.

🗜 3. Advanced & Tool-Specific Questions

31. Explain Git rebase vs merge.

`git merge` combines branches preserving history, while `git rebase` rewrites commits to create a linear history. Rebase is cleaner but should be avoided on shared branches to prevent conflicts.

32. What is Jenkins agent and how does it work?

Jenkins agents (nodes) execute jobs distributed from the master (controller). They connect via SSH or JNLP and enable parallel builds across multiple machines, improving scalability.

33. What is Docker Compose?

Docker Compose is a YAML-based tool to define and run multi-container applications. It simplifies orchestration by defining services, networks, and volumes in a single file.

34. Explain Dockerfile and its purpose.

A Dockerfile contains instructions to build a Docker image — defining the base image, dependencies, and commands. It ensures consistent environments and automated builds.

35. What is Kubernetes Ingress?

Ingress is an API object that manages external HTTP/S access to services in Kubernetes. It provides load balancing, SSL termination, and routing via YAML-defined rules.

36. What are Helm charts?

Helm is Kubernetes' package manager. Charts are pre-configured templates that define a complete Kubernetes application — simplifying deployment, rollback, and version control.

37. How do you secure Kubernetes clusters?

Enable RBAC, use namespaces, network policies, and secrets encryption. Regularly scan container images and use tools like kube-bench for compliance checks.

38. What is Ansible and why is it agentless?

Ansible is an open-source automation tool that manages configurations and deployments using SSH. It's agentless, meaning no software installation is needed on client machines.

39. What is Terraform and how does it differ from Ansible?

Terraform focuses on provisioning infrastructure, while Ansible manages configurations. Terraform uses a declarative syntax and maintains a state file for tracking resources.

40. Explain Prometheus architecture.

Prometheus collects metrics via HTTP pull from targets. It stores time-series data in its database and uses Alertmanager for notifications. Grafana is often used for visualization.

41. What is a Service Mesh?

A Service Mesh (e.g., Istio or Linkerd) manages communication between microservices. It provides observability, security, and traffic control without modifying application code.

42. How does Blue-Green differ from Canary deployment?

Blue-Green switches traffic between two environments at once, while Canary gradually releases to a subset of users. Canary provides better real-time risk control.

43. How do you build lightweight Docker images?

Use minimal base images like Alpine, multi-stage builds to remove intermediate layers, and `.dockerignore` to exclude unnecessary files. This improves performance and security.

44. What are ephemeral environments?

Ephemeral environments are short-lived test setups created per pull request or branch. They enable isolated testing using IaC and get destroyed after use.

45. What is Terraform state file and why is it important?

The Terraform state file stores metadata about deployed infrastructure. It helps Terraform know what resources exist and manage updates safely. Always secure and back up the state file.

46. How does Ansible manage inventory?

Ansible uses inventory files to list target hosts and groups. It supports static (INI/YAML) and dynamic inventories from cloud providers, simplifying environment management.

47. How do you integrate AWS with Kubernetes (EKS)?

Use IAM roles for service accounts, EBS CSI drivers for persistent volumes, and ALB ingress controllers. This integration simplifies scalability and secure workload management.

48. How do you implement alerting in Prometheus?

Define alert rules in configuration files, set thresholds, and connect Alertmanager to send notifications to email, Slack, or PagerDuty when alerts trigger.

49. How do you ensure observability in microservices?

Implement structured logging, distributed tracing (Jaeger/Zipkin), and metrics collection (Prometheus). Observability provides insights into application health and performance.

50. What is a Service Discovery mechanism?

Service Discovery allows microservices to find each other dynamically without hardcoding endpoints. Tools like Consul or Kubernetes DNS automate registration and communication.

🈸**Prepared by:** CDEC Trainer *Abhipray Dhoble — Kothrud, Pune*