CI/CD Pipeline - 50 Mock Interview Questions & Answers

1. Q1: What is a CI/CD pipeline?

A: It’s an automated process that builds, tests, and deploys applications, ensuring faster and reliable software delivery.

2. Q2: What is the difference between CI and CD?

A: CI (Continuous Integration) automates code builds and tests. CD (Continuous Delivery/Deployment) automates application releases to environments.

3. Q3: Continuous Delivery vs Continuous Deployment?

A: Continuous Delivery prepares the application for release but requires manual approval. Continuous Deployment automatically releases to production without manual intervention.

4. Q4: What are the main stages of a CI/CD pipeline?

A: Source → Build → Test → Deploy → Monitor.

5. Q5: What are the benefits of CI/CD?

A: Faster delivery, reduced errors, consistent deployments, quick feedback, and higher developer productivity.

6. Q6: What is Blue-Green deployment?

A: Two identical environments (Blue & Green). Traffic switches to the new environment after deployment to minimize downtime.

7. Q7: What is Canary deployment?

A: Releasing software to a small percentage of users first, then gradually rolling out to everyone.

8. Q8: Rolling update vs Recreate deployment?

A: Rolling updates replace pods gradually, while recreate terminates all old pods before starting new ones.

9. Q9: What are feature flags in CI/CD?

A: Configurations to enable/disable features without redeploying code.

10. Q10: How do pipelines improve developer collaboration?

A: They provide shared visibility, immediate feedback, and reduce integration conflicts.

11. Q11: Explain Jenkins architecture.

A: Jenkins follows a master-agent model where the master schedules jobs, and agents execute builds.

12. Q12: Jenkins pipeline vs freestyle job?

A: Freestyle is simple UI-based, pipeline is code-based (Jenkinsfile) and supports complex workflows.

13. Q13: Role of .gitlab-ci.yml in GitLab?

A: Defines the pipeline structure, jobs, and stages.

14. Q14: What is GitHub Actions?

A: A CI/CD tool that runs workflows using YAML configuration within GitHub.

15. Q15: What is ArgoCD?

A: A GitOps tool for continuous delivery in Kubernetes.

16. Q16: What is Spinnaker?

A: A multi-cloud continuous delivery platform.

17. Q17: What are Tekton pipelines?

A: Kubernetes-native CI/CD pipelines built using CRDs.

18. Q18: Bamboo vs Jenkins?

A: Bamboo is commercial with Atlassian integration, Jenkins is open-source and highly customizable.

19. Q19: Declarative vs Scripted Jenkinsfile?

A: Declarative is structured and simple; scripted provides flexibility with Groovy scripting.

20. Q20: How to implement parallel stages in Jenkins?

A: Using parallel directive inside pipeline stages.

21. Q21: How do Git hooks help CI/CD?

A: They trigger scripts on events (e.g., pre-commit, post-push) for automation.

22. Q22: GitFlow vs Trunk-based branching in CI/CD?

A: GitFlow uses long-lived branches, Trunk-based prefers small, frequent commits on main.

23. Q23: Role of pull requests in CI/CD?

A: They enforce code review and automated testing before merging.

24. Q24: How to handle merge conflicts in pipelines?

A: Resolve conflicts locally, rebase or merge, and re-run pipeline.

25. Q25: What are webhooks in CI/CD?

A: HTTP callbacks that trigger pipelines when events occur (e.g., code push).

26. Q26: How do you build apps with Maven/Gradle in CI/CD?

A: By integrating mvn install or gradle build into pipeline stages.

27. Q27: Types of tests in CI/CD?

A: Unit, Integration, End-to-End, Performance, Security.

28. Q28: Role of SonarQube?

A: Analyzes code quality, bugs, vulnerabilities, and technical debt.

29. Q29: What are quality gates?

A: Criteria like code coverage and security checks that must pass before proceeding.

30. Q30: What is SAST and DAST?

A: Static Application Security Testing (code analysis) and Dynamic Application Security Testing (runtime testing).

31. Q31: How do you build & push Docker images in pipelines?

A: Using docker build & docker push integrated in pipeline jobs.

32. Q32: Kubernetes deployment automation in CI/CD?

A: By applying manifests/Helm charts via kubectl or GitOps tools.

33. Q33: Role of Helm in CI/CD?

A: Helm charts manage Kubernetes application packaging and deployment.

34. Q34: How do you integrate Terraform in CI/CD?

A: By running terraform plan and terraform apply as pipeline stages.

35. Q35: How to achieve zero-downtime deployment?

A: Using rolling updates, Blue-Green, or Canary strategies.

36. Q36: How do you detect pipeline failures?

A: By monitoring logs, test reports, and failure notifications.

37. Q37: How do you monitor pipeline metrics?

A: Using tools like Prometheus, Grafana, ELK stack.

38. Q38: Role of Prometheus in CI/CD?

A: Collects metrics and monitors system performance.

39. Q39: How to set up alerts for pipeline failures?

A: Using integrations with Slack, PagerDuty, or email alerts.

40. Q40: How to rollback failed deployments?

A: Using previous stable builds, Git revert, or Kubernetes kubectl rollout undo.

41. Q41: What if Jenkins master goes down?

A: Pipelines stop; use HA setup or backup masters to recover.

42. Q42: How do you handle secrets in pipelines?

A: Using Vault, AWS KMS, Kubernetes Secrets, or environment variables.

43. Q43: How to build multi-cloud CI/CD?

A: Using Terraform + multi-cloud tools like Spinnaker or GitHub Actions.

44. Q44: Difference in CI/CD for monolith vs microservices?

A: Monolith uses a single pipeline, microservices need multiple pipelines per service.

45. Q45: How do you set manual approvals in CI/CD?

A: Using pipeline gates (Jenkins input, GitLab manual jobs, etc.).

46. Q46: How do you optimize pipeline speed?

A: By caching dependencies, parallel execution, and using ephemeral environments.

47. Q47: What are ephemeral environments?

A: Temporary test environments created per pull request and destroyed after use.

48. Q48: How do you ensure compliance in pipelines?

A: By enforcing security scans, audit logging, and approval gates.

49. Q49: How to automate DR (Disaster Recovery) testing in CI/CD?

A: By simulating outages and validating recovery scripts via pipeline jobs.

50. Q50: What's the best pipeline you've designed?

A: (Personal answer expected: mention a real-world CI/CD design, challenges, and improvements).