# DevOps Questions and Answers

## Q: What is DevOps and why is it important?

A: DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the development lifecycle and deliver high-quality software continuously.

## Q: How does DevOps differ from traditional software development models?

A: Traditional models often separate development and operations, leading to silos. DevOps promotes collaboration, automation, and continuous feedback, resulting in faster and more reliable releases.

## Q: What are the key principles of DevOps?

A: - Continuous Integration and Delivery - Automation - Collaboration - Monitoring and Feedback - Infrastructure as Code

## Q: Can you explain the DevOps lifecycle?

A: The lifecycle includes: Plan → Develop → Build → Test → Release → Deploy → Operate → Monitor → Feedback.

## Q: What tools are commonly used in DevOps pipelines?

A: - Git (version control) - Jenkins/GitLab CI (CI/CD) - Docker (containerization) - Kubernetes (orchestration) - Ansible/Terraform (IaC) - Prometheus/Grafana (monitoring)

## Q: What is Infrastructure as Code (IaC)?

A: IaC is managing and provisioning infrastructure through code instead of manual processes. Tools: Terraform, Ansible, CloudFormation.

## Q: How do DevOps practices improve software quality and delivery speed?

A: Through automation, continuous testing, and rapid feedback loops, DevOps reduces errors and accelerates delivery.

## Q: What are some challenges in implementing DevOps?

A: - Cultural resistance - Tool integration - Skill gaps - Legacy systems

## Q: What is the role of automation in DevOps?

A: Automation is key for CI/CD, testing, deployment, and infrastructure provisioning, reducing manual errors and speeding up processes.

## Q: How do monitoring and logging fit into the DevOps process?

A: They provide visibility into system performance and help detect issues early, enabling proactive maintenance and faster incident response.

## Q: What is the role of feedback loops in DevOps?

A: Feedback loops help teams identify issues early, improve continuously, and align development with user needs.

## Q: What is the difference between Agile and DevOps?

A: Agile focuses on iterative development; DevOps extends Agile by including operations and deployment.

## Q: What is a DevOps toolchain?

A: A set of integrated tools that support development, integration, testing, delivery, deployment, and monitoring.

## Q: What is shift-left testing?

A: Moving testing earlier in the development cycle to catch bugs sooner.

## Q: What is the role of containers in DevOps?

A: Containers ensure consistency across environments, making deployments faster and more reliable.

## Q: What is continuous monitoring?

A: The practice of constantly observing systems and applications to detect issues and performance bottlenecks.

## Q: How does DevOps support microservices architecture?

A: DevOps enables independent deployment, scaling, and monitoring of microservices.

## Q: What is the CALMS model in DevOps?

A: Culture, Automation, Lean, Measurement, Sharing – a framework for assessing DevOps maturity.

## Q: What is the difference between push and pull deployment models?

A: - Push: Central server initiates deployment. - Pull: Nodes pull updates from a central repository.

## Q: What is the role of version control in DevOps?

A: It tracks changes, supports collaboration, and integrates with CI/CD pipelines for automation.

# CI/CD Questions and Answers

## Q: What is CI/CD and how does it relate to DevOps?

A: CI/CD automates the integration, testing, and deployment of code. It’s a core practice in DevOps for delivering software quickly and reliably.

## Q: Differences between CI, CD (Delivery), and CD (Deployment)?

A: - CI: Automatically integrates and tests code. - CD (Delivery): Prepares code for release. - CD (Deployment): Automatically deploys to production.

## Q: Benefits of CI/CD pipelines?

A: - Faster releases - Reduced bugs - Improved collaboration - Consistent deployments

## Q: Typical stages in a CI/CD pipeline?

A: - Code commit - Build - Test - Package - Deploy - Monitor

## Q: Common CI/CD tools?

A: Jenkins, GitLab CI, GitHub Actions, CircleCI, Travis CI, Bamboo.

## Q: How do you handle failures in a CI/CD pipeline?

A: - Automated rollback - Notifications - Logs and metrics for debugging - Manual intervention if needed

## Q: What is a build artifact?

A: A file or set of files generated after a build (e.g., JAR, Docker image) that can be deployed.

## Q: How do you ensure security in CI/CD?

A: - Secrets management - Code scanning - Dependency checks - Role-based access

## Q: Role of version control in CI/CD?

A: It tracks changes, supports collaboration, and ensures traceability and integration with CI/CD pipelines.

## Q: Blue-green deployment vs. Canary release?

A: - Blue-green: Two environments; switch traffic to the new one. - Canary: Gradually roll out to a subset of users.

## Q: What is a pipeline as code?

A: Defining CI/CD workflows in code (e.g., YAML files) for versioning and automation.

## Q: What is artifact repository management?

A: Tools like Nexus or Artifactory store and manage build artifacts for reuse and deployment.

## Q: What is a rollback strategy in CI/CD?

A: Techniques to revert to a previous stable version in case of deployment failure.

## Q: What is test automation in CI/CD?

A: Running automated tests (unit, integration, etc.) as part of the pipeline to ensure code quality.

## Q: What is a staging environment?

A: A pre-production environment that mimics production for final testing.

## Q: What is pipeline orchestration?

A: Managing the sequence and dependencies of tasks in a CI/CD pipeline.

## Q: What is a self-hosted runner in CI/CD?

A: A custom machine that executes pipeline jobs instead of using cloud-hosted runners.

## Q: What is a build matrix?

A: Running builds across multiple environments (e.g., OS, language versions) to ensure compatibility.

## Q: What is the difference between declarative and scripted pipelines?

A: - Declarative: YAML/DSL-based, easier to read. - Scripted: More flexible, written in code (e.g., Groovy in Jenkins).

## Q: What is pipeline caching?

A: Storing dependencies or build outputs to speed up subsequent pipeline runs.

# Docker Questions and Answers

## Q: What is Docker?

A: Docker is a platform for developing, shipping, and running applications in lightweight, portable containers.

## Q: Container vs. Virtual Machine?

A: Containers share the host OS kernel, making them lightweight. VMs have full OS instances, making them heavier.

## Q: What is a Dockerfile?

A: A script with instructions to build a Docker image. Includes base image, commands, environment variables, etc.

## Q: How to build and run a Docker container?

A: docker build -t myapp . docker run -d -p 8080:80 myapp

## Q: What is Docker Compose?

A: A tool to define and run multi-container Docker applications using a docker-compose.yml file.

## Q: Image vs. Container?

A: - Image: Blueprint of the application. - Container: Running instance of an image.

## Q: Managing data in Docker?

A: - Volumes: Managed by Docker. - Bind mounts: Link host directory to container.

## Q: Docker networks?

A: Allow containers to communicate. Types: bridge, host, overlay.

## Q: Optimizing Docker images?

A: - Use smaller base images - Minimize layers - Clean up unnecessary files

## Q: Common Docker commands?

A: - docker build, docker run, docker ps, docker exec, docker stop, docker rm, docker images

## Q: What is the Docker daemon?

A: The background service that manages Docker containers and images.

## Q: What is the purpose of the .dockerignore file?

A: It excludes files from being copied into the Docker image, reducing size and build time.

## Q: What is a multi-stage build in Docker?

A: A technique to use multiple FROM statements to optimize image size by separating build and runtime environments.

## Q: How do you update a running container?

A: You typically stop and remove the old container, then run a new one with the updated image.

## Q: What is the difference between ENTRYPOINT and CMD in Dockerfile?

A: - ENTRYPOINT: Defines the main command. - CMD: Provides default arguments to ENTRYPOINT.

## Q: How do you secure Docker containers?

A: - Use minimal base images - Run as non-root - Use Docker Bench for Security - Regularly scan images

## Q: What is Docker Swarm?

A: Docker’s native clustering and orchestration tool for managing a cluster of Docker nodes.

## Q: How do you share data between containers?

A: Using volumes or shared networks.

## Q: What is the difference between docker exec and docker run?

A: - docker exec: Runs a command in a running container. - docker run: Starts a new container.

## Q: How do you troubleshoot a failing container?

A: Use docker logs, docker inspect, and docker exec to diagnose issues.

# Kubernetes Questions and Answers

## Q: What is Kubernetes?

A: An open-source container orchestration platform for automating deployment, scaling, and management of containerized apps.

## Q: Main components of Kubernetes architecture?

A: - Master Node: API Server, Scheduler, Controller Manager - Worker Node: kubelet, kube-proxy, container runtime

## Q: What is a Pod?

A: The smallest deployable unit in Kubernetes. A Pod can contain one or more containers.

## Q: Deployment vs. ReplicaSet vs. StatefulSet?

A: - Deployment: Manages stateless apps. - ReplicaSet: Ensures a specified number of pod replicas. - StatefulSet: Manages stateful apps with stable identities.

## Q: Service discovery and load balancing?

A: Kubernetes Services expose Pods and provide DNS names and load balancing across them.

## Q: ConfigMap vs. Secret?

A: - ConfigMap: Stores non-sensitive config data. - Secret: Stores sensitive data like passwords, tokens.

## Q: Scaling in Kubernetes?

A: - Manual: kubectl scale - Auto: Horizontal Pod Autoscaler (HPA)

## Q: Role of kubelet and kube-proxy?

A: - kubelet: Manages Pods on a node. - kube-proxy: Handles networking and load balancing.

## Q: Persistent Volumes and Claims?

A: - PV: Storage resource in the cluster. - PVC: Request for storage by a user.

## Q: What is Helm?

A: A package manager for Kubernetes that simplifies deployment using charts (pre-configured resources).

## Q: What is a Namespace in Kubernetes?

A: A way to divide cluster resources between multiple users or teams.

## Q: What is a DaemonSet?

A: Ensures a copy of a Pod runs on all (or some) nodes in the cluster.

## Q: What is a Job and CronJob in Kubernetes?

A: - Job: Runs a task to completion. - CronJob: Schedules Jobs at specified times.

## Q: What is a ServiceAccount in Kubernetes?

A: Provides an identity for processes running in Pods to interact with the API server.

## Q: What is a NodePort service?

A: Exposes a service on a static port on each node’s IP.

## Q: What is a Horizontal Pod Autoscaler (HPA)?

A: Automatically scales Pods based on CPU/memory usage or custom metrics.

## Q: What is a readiness probe vs. liveness probe?

A: - Readiness: Checks if the app is ready to serve traffic. - Liveness: Checks if the app is alive and should be restarted if not.

## Q: What is a sidecar container?

A: A helper container in the same Pod that supports the main container (e.g., logging, proxy).

## Q: What is the role of etcd in Kubernetes?

A: A distributed key-value store that holds cluster state and configuration.

## Q: What is a Custom Resource Definition (CRD)?

A: Allows users to define their own resources and extend Kubernetes functionality.