**Coupling and cohesion**

**DevOps Interview Questions and Answers**

**1. What is DevOps?**

**Answer:** The DevOps is a continuous process that emphasizes collaboration, automation, and integration between development (Dev) and operations (Ops) teams to deliver high-quality software quickly and efficiently.

key phases of the DevOps lifecycle:

1. Plan

Involves defining requirements, project goals, and workflows.

Tools: Jira, Trello, Confluence, GitLab (issues/planning boards).

2. Develop

Writing, reviewing, and managing code.

Continuous integration (CI) ensures code changes are merged into the main branch and tested frequently.

Tools: Git, GitHub, GitLab, Bitbucket, VS Code.

3. Build

Compiling source code into executable artifacts.

Automated builds and dependency management ensure consistent results.

Tools: Jenkins, Maven, Gradle, Azure DevOps.

4. Test

Automated and manual testing to ensure functionality, performance, and security.

Continuous testing catches bugs early in the development cycle.

Tools: Selenium, JUnit, TestNG, Postman.

5. Release

Preparing the software for deployment by packaging it into containers or executables.

Tools for managing version control, tagging releases, and change approval processes are used.

Tools: GitLab CI/CD, AWS CodePipeline, Azure DevOps, Jenkins.

6. Deploy

Automatically or manually deploying the application to production or staging environments.

Infrastructure-as-Code (IaC) tools and container orchestration streamline deployments.

Tools: Kubernetes, Docker, AWS CodeDeploy, Ansible.

7. Operate

Monitoring and managing application performance, infrastructure, and uptime.

Logging, alerting, and proactive issue detection ensure reliability.

Tools: Prometheus, Grafana, Splunk, Nagios.

8. Monitor

Continuous feedback loops using monitoring, analytics, and user feedback.

Observability provides actionable insights for improvement.

Tools: New Relic, Datadog, ELK Stack (Elasticsearch, Logstash, Kibana).

**2. What are the key components of DevOps?**

**Answer:** The key components of DevOps include:

* Continuous Integration (CI)
* Continuous Deployment (CD)
* Configuration Management
* Infrastructure as Code (IaC)
* Monitoring and Logging
* Collaboration and Communication

**3. What are the benefits of DevOps?**

**Answer:** DevOps offers several benefits, including:

* Faster software releases
* Improved collaboration between development and operations teams
* Increased efficiency and reduced deployment failures
* Enhanced scalability and reliability
* Better monitoring and feedback loops

**4. What is CI/CD?**

**Answer:**

* **Continuous Integration (CI)** involves automatically integrating code changes from multiple contributors into a shared repository multiple times a day.
* **Continuous Deployment (CD)** ensures that code changes are automatically tested and deployed to production without manual intervention.

**5. What tools are commonly used in DevOps?**

**Answer:** Common DevOps tools include:

* **Version Control:** Git, GitHub, GitLab, Bitbucket
* **CI/CD:** Jenkins, GitHub Actions, GitLab CI/CD, AWS CodePipeline
* **Configuration Management:** Ansible, Puppet, Chef
* **Containerization:** Docker, Kubernetes
* **Monitoring & Logging:** Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana)
* **Infrastructure as Code:** Terraform, AWS CloudFormation

**6. What is Infrastructure as Code (IaC)?**

**Answer:** Infrastructure as Code (IaC) is the **practice of managing and provisioning** infrastructure using machine-readable script files instead of manual processes. Tools like Terraform and AWS CloudFormation help automate infrastructure management.

**7. What is Docker, and why is it used in DevOps?**

**Answer:** Docker is a containerization platform that enables developers to package applications and their dependencies into lightweight, portable containers. It helps in:

* Consistent development environments
* Rapid deployment and scaling
* Resource efficiency and isolation

**8. What is Kubernetes?**

**Answer:** Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It ensures high availability and efficient resource utilization.

**9. What is a Microservices Architecture?**

**Answer:** Microservices architecture is an approach to software development where an application is built as a collection of small, loosely coupled services that can be independently deployed and scaled.

**10. What are the differences between Monolithic and Microservices Architectures?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Monolithic Architecture** | **Microservices Architecture** |
| Scalability | Limited | Highly Scalable |
| Deployment | Slow, needs full redeployment | Fast, deploy services independently |
| Maintenance | Harder to maintain | Easier to maintain |
| Technology Flexibility | Limited | Can use different technologies per service |

**11. What is a Blue-Green Deployment?**

**Answer:** Blue-Green Deployment is a release management strategy where two identical environments (Blue and Green) exist. The live environment (Blue) serves traffic while updates are deployed to the inactive environment (Green). Once the new version is verified, traffic is switched from Blue to Green, ensuring minimal downtime.

**12. What is the role of monitoring in DevOps?**

**Answer:** Monitoring helps in:

* Identifying and resolving performance issues
* Ensuring high availability
* Proactive alerting for failures
* Gathering insights for continuous improvement

**13. What are the differences between DevOps and Agile?**

|  |  |  |
| --- | --- | --- |
| Aspect | DevOps | Agile |
| Focus | Collaboration between Dev & Ops | Iterative development & customer feedback |
| Scope | Covers development & operations | Focuses on development process |
| Automation | Strong emphasis on automation | Less emphasis on automation |
| Delivery | Continuous integration & deployment | Incremental releases |

**14. What is a Load Balancer?**

**Answer:** A Load Balancer is a network component that distributes incoming traffic across multiple servers to ensure reliability and performance.

**15. What is the difference between Stateful and Stateless applications?**

**Answer:**

* **Stateful Applications** store client data across sessions (e.g., databases, chat applications).
* **Stateless Applications** do not store client data and each request is independent (e.g., REST APIs).

**16. What is Ansible?**

**Answer:** Ansible is an open-source automation tool used for configuration management, application deployment, and task automation using YAML playbooks.

**17. What is Terraform?**

**Answer:** Terraform is an Infrastructure as Code (IaC) tool that allows users to define and provision infrastructure using a declarative configuration language.

**18. What are the different deployment strategies?**

**Answer:**

* **Rolling Deployment** – Gradually updates instances with minimal downtime.
* **Canary Deployment** – Deploys a small percentage of traffic to new versions before full rollout.
* **Blue-Green Deployment** – Uses two environments for seamless deployment.
* **Recreate Deployment** – Shuts down existing versions before deploying new ones.

**19. What is Helm in Kubernetes?**

**Answer:** Helm is a package manager for Kubernetes that simplifies the deployment and management of applications using Helm charts.

**20. How do you secure a DevOps pipeline?**

**Answer:**

* Implement role-based access control (RBAC)
* Use secrets management tools like AWS Secrets Manager or HashiCorp Vault
* Enable logging and monitoring
* Automate security scans in the pipeline
* Use immutable infrastructure principles

### 1. How will you approach a project that needs to implement DevOps?

**Stage 1:** Assess the existing process and implementation for 2-3 weeks to identify areas for improvement and create a roadmap.

**Stage 2:** Create a proof of concept (PoC). Once approved, start implementing and rolling out the project plan.

**Stage 3:** Implement DevOps by setting up version control, integration, testing, deployment, delivery, and monitoring.

### 2. What is the difference between Continuous Delivery and Continuous Deployment?

|  |  |
| --- | --- |
| **Continuous Delivery** | **Continuous Deployment** |
| Ensures code can be safely deployed to production | Every change that passes automated tests is deployed to production automatically |
| Ensures business applications function as expected | Enables faster, more robust software development and release |
| Requires manual approval for production deployment | No manual approval is needed; relies on monitoring |

### 3. What is the role of Configuration Management in DevOps?

* Manages and standardizes IT infrastructure.
* Administers multiple servers and maintains system integrity.
* Ensures consistency across environments.

### 4. How does Continuous Monitoring help maintain system architecture?

* Detects, identifies, and reports faults or threats.
* Ensures services and applications run correctly.
* Provides continuous auditing, transaction inspection, and control monitoring.

### 5. What is the role of AWS in DevOps?

* **Flexible services**: Ready-to-use, no setup needed.
* **Scalability**: Manage single or thousands of instances.
* **Automation**: Automates tasks and processes.
* **Security**: AWS IAM for access control.
* **Large partner ecosystem**: Supports multiple integrations.

### 6. Name three important DevOps KPIs. Key Performance Indicators

1. **Mean Time to Recovery (MTTR):** Average time to recover from a failure.
2. **Deployment Frequency:** Number of deployments in a given period.
3. **Percentage of Failed Deployments:** Measures stability of releases.

### 7. Explain "Infrastructure as Code" (IaC) in Configuration Management.

* Uses code to manage configurations, deployments, and provisioning.
* Defines infrastructure with machine-readable files (e.g., YAML, JSON).
* Ensures consistency across servers and environments.

### 8. How is IaC implemented using AWS?

* Replaces traditional manual configurations with code.
* Uses formats like JSON/YAML for infrastructure provisioning.
* Enables fast and automated deployment of infrastructure changes.

### 9. Why has DevOps gained prominence in recent years?

* Used by tech giants like Netflix and Facebook for automation.
* Enhances deployment speed and reliability.
* Ensures higher success rates for releases and reduces lead time.

### 10. What are the key differences between DevOps and Agile?

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Agile** | **DevOps** |
| Work Scope | Focuses on agility | Focuses on automation and agility |
| Feedback | Customer feedback | Tool-based feedback (monitoring) |
| Processes | Uses Scrum, Kanban | Uses CI/CD, automated testing |
| Release Cycle | Smaller releases | Immediate feedback & releases |

### 11. What are anti-patterns in DevOps?

* Assuming DevOps is a process or tool, not a culture.
* Creating a separate DevOps team rather than integrating it.
* Believing DevOps solves all organizational problems.
* Assuming DevOps equals Agile.

### 12. What are the benefits of using Version Control?

* Enables team collaboration without conflicts.
* Tracks changes with history logs.
* Provides a backup and restore mechanism.
* Supports branching and merging strategies.

### 13. Describe branching strategies used in DevOps.

1. **Release Branching**: Branches are created when a release is ready.
2. **Feature Branching**: Each feature is developed in its own branch.
3. **Task Branching**: Each task has its dedicated branch.

### 14. What is "Shift Left" in DevOps?

* Moving security and testing earlier in the development process.
* Detects issues early, reducing failures and costs.
* Integrates security into development rather than post-deployment.

### 15. What is the Blue/Green Deployment pattern?

* **Blue environment**: Runs the old version.
* **Green environment**: Hosts the new version.
* Traffic is switched from blue to green after validation, ensuring zero downtime.

### 16. What is Continuous Testing?

* Automates testing in the software delivery pipeline.
* Provides immediate feedback on business risks.
* Prevents errors from propagating through the development cycle.

### 17. What is Automation Testing?

* Automates manual testing procedures.
* Improves speed and efficiency of software validation.
* Ensures consistent testing results.

**DevOps Interview Preparation: Git Questions**

**1. Introduction to Git**

Git is a distributed version control system widely used in DevOps for managing source code. It allows multiple developers to collaborate efficiently, track changes, and manage project versions.

**2. Basic Git Commands**

Here are some fundamental Git commands and their explanations:

* git init - Initializes a new Git repository.
* git clone <repository\_url> - Clones an existing repository.
* git add <file> - Stages a file for commit.
* git commit -m "message" - Commits the staged changes.
* git push origin <branch> - Pushes changes to a remote repository.
* git pull origin <branch> - Pulls changes from a remote repository.
* git status - Shows the status of the working directory.
* git log - Displays the commit history.
* git checkout <branch> - Switches to another branch.
* git branch <branch\_name> - Creates a new branch.
* git merge <branch> - Merges changes from another branch.

**3. Git Branching and Merging**

Branching in Git allows developers to work on different features simultaneously.

* **Feature Branching**: Create a separate branch for each feature.
* **Merging Strategies**:
  + Fast-forward merge (git merge --ff)
  + Recursive merge (default)
  + Rebase (git rebase <branch>)

**4. Git Conflict Resolution**

Conflicts occur when multiple changes affect the same part of a file. To resolve:

1. Use git status to check conflicting files.
2. Open the files and manually edit conflicts.
3. Use git add <file> to stage resolved files.
4. Commit the resolution using git commit.

**5. Git Stash**

The git stash command temporarily saves changes without committing them.

* git stash save "message" - Stashes changes.
* git stash pop - Applies stashed changes and removes them.
* git stash list - Lists all stashed changes.
* git stash drop - Removes a specific stash.

**6. Git Rebase vs. Merge**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Merge** | **Rebase** |
| Preserves commit history | Yes | No |
| Creates a new commit | Yes | No |
| Used for integrating branches | Yes | Yes |
| Avoids unnecessary merge commits | No | Yes |

**7. Git Hooks**

Git hooks are scripts that run before or after events such as commits, pushes, and merges.

* pre-commit - Runs before a commit.
* post-commit - Runs after a commit.
* pre-receive - Runs on the server before updating refs.
* post-receive - Runs on the server after updating refs.

**8. Git Tags**

Tags mark specific points in the commit history, often for releases.

* git tag -a v1.0 -m "Version 1.0" - Creates an annotated tag.
* git tag - Lists all tags.
* git push origin --tags - Pushes tags to the remote repository.

**9. Git Best Practices**

* Use meaningful commit messages.
* Keep commits small and focused.
* Use branches effectively (feature, bugfix, hotfix).
* Regularly pull from the remote repository to stay updated.
* Avoid pushing broken code.

**10. Conclusion**

Mastering Git is essential for DevOps engineers as it facilitates efficient source code management and collaboration. Understanding Git commands, workflows, and best practices will help in both development and deployment processes.

1. **Explain the master-slave architecture of Jenkins.**

* The Jenkins master pulls the code from the remote GitHub repository every time there is a code commit.
* It distributes the workload to all the Jenkins slaves.
* On request from the Jenkins master, the slaves carry out builds, tests, and produce test reports.

1. **What is a Jenkinsfile?**

* A Jenkinsfile contains the definition of a Jenkins pipeline and is checked into the source control repository. It is a text file.
* It allows code review and iteration on the pipeline.
* It permits an audit trail for the pipeline.
* It provides a single source of truth for the pipeline, which can be viewed and edited.

1. **Which of the following commands runs Jenkins from the command line?**

* java –jar Jenkins.war
* java –war Jenkins.jar
* java –jar Jenkins.jar
* java –war Jenkins.war
* **Correct Answer:** A) java –jar Jenkins.war

1. **What concepts are key aspects of the Jenkins pipeline?**

* **Pipeline:** A user-defined model of a CI/CD pipeline. The pipeline code defines the entire build process, including building, testing, and delivering an application.
* **Node:** A machine that is part of the Jenkins environment and capable of executing a pipeline.
* **Step:** A single task that tells Jenkins what to do at a particular point in time.
* **Stage:** Defines a conceptually distinct subset of tasks performed throughout the pipeline (e.g., build, test, deploy stages).

1. **Which file is used to define dependencies in Maven?**

* build.xml
* pom.xml
* dependency.xml
* version.xml
* **Correct Answer:** B) pom.xml

1. **Explain the two types of pipelines in Jenkins, along with their syntax.**

* **Scripted Pipeline:**
  + Based on Groovy script as a Domain Specific Language (DSL).
  + Uses node blocks to define stages and steps.
  + Syntax:

groovy

node {

stage('Build') {

steps {

// Build stage commands

}

}

stage('Test') {

steps {

// Test stage commands

}

}

stage('Deploy') {

steps {

// Deploy stage commands

}

}

}

* **Declarative Pipeline:**
  + Provides a simpler syntax for defining pipelines.
  + Uses pipeline block to define the workflow.
  + Syntax:

groovy

pipeline {

agent any

stages {

stage('Build') {

steps {

// Build steps

}

}

stage('Test') {

steps {

// Test steps

}

}

stage('Deploy') {

steps {

// Deploy steps

}

}

}

}

1. **How do you create a backup and copy files in Jenkins?**

* Periodically back up the JENKINS\_HOME directory.
* To create a backup, copy the JENKINS\_HOME directory.
* To clone or replicate a job, copy the job directory.

1. **How can you copy Jenkins from one server to another?**

* Move a job from one Jenkins installation to another by copying the corresponding job directory.
* Clone an existing job by duplicating its job directory under a different name.
* Rename an existing job by renaming its directory.

1. **Name three security mechanisms Jenkins uses to authenticate users.**

* Jenkins uses an internal database to store user data and credentials.
* Jenkins can authenticate users via a Lightweight Directory Access Protocol (LDAP) server.
* Jenkins can integrate with the authentication mechanism of the deployed application server.

1. **How is a custom build of a core plugin deployed?**

* Copy the .hpi file to $JENKINS\_HOME/plugins.
* Remove the plugin's development directory.
* Create an empty file named <plugin>.hpi.pinned.
* Restart Jenkins to use the custom build of the plugin.

1. **How can you temporarily turn off Jenkins security if the administrative users have locked themselves out of the admin console?**

* Modify the Jenkins configuration file (config.xml) and locate the <useSecurity> element.
* Change its value from true to false.
* Restart Jenkins to disable security temporarily.

1. **What are the ways in which a build can be scheduled/run in Jenkins?**

* Triggered by source code management (SCM) commits.
* After the completion of another build.
* Scheduled to run at a specified time.
* Manually triggered by a user.

1. **What are the commands to restart Jenkins manually?**

* (Jenkins\_url)/restart → Forces an immediate restart without waiting for builds to complete.
* (Jenkins\_url)/safeRestart → Allows running builds to complete before restarting Jenkins.

1. **Explain how you can set up a Jenkins job.**

* Navigate to the Jenkins homepage and select **New Job**.
* Choose **Build a free-style software project**.
* Configure the job with:
  + **Triggers** for when to build.
  + **Steps** for gathering data, testing, and archiving artifacts.
  + **Build scripts** (e.g., Maven, shell script).
  + **Source code management (SCM)** configuration (e.g., Git, SVN).

### 5. Mention Some of the Core Benefits of DevOps

The core benefits of DevOps are as follows:

#### **Technical Benefits**

* Continuous software delivery
* Less complex problems to manage
* Early detection and faster correction of defects

#### **Business Benefits**

* Faster delivery of features
* Stable operating environments
* Improved communication and collaboration between teams

### 6. How Will You Approach a Project That Needs to Implement DevOps?

The following standard approaches can be used to implement DevOps in a project:

#### **Stage 1**

* Assess the existing process and implementation for about two to three weeks.
* Identify areas of improvement to create a roadmap for implementation.

#### **Stage 2**

* Create a Proof of Concept (PoC).
* Once approved, start implementing the project plan.

#### **Stage 3**

* Implement DevOps using a step-by-step process for version control, integration, testing, deployment, delivery, and monitoring.

### 7. What Is the Difference Between Continuous Delivery and Continuous Deployment?

| **Continuous Delivery** | **Continuous Deployment** |
| --- | --- |
| Ensures code can be safely deployed onto production | Every change that passes automated tests is deployed to production automatically |
| Ensures business applications and services function as expected | Makes software development and the release process faster and more robust |
| Delivers every change to a production-like environment through rigorous automated testing | No explicit approval from a developer is needed; requires a strong monitoring culture |

### 8. What Is the Role of Configuration Management in DevOps?

* Enables management of and changes to multiple systems.
* Standardizes resource configurations, which in turn manage IT infrastructure.
* Helps with administration and management of multiple servers while maintaining infrastructure integrity.

### 9. How Does Continuous Monitoring Help Maintain the Entire Architecture of the System?

Continuous monitoring in DevOps detects, identifies, and reports faults or threats in the system’s infrastructure.

* Ensures all services, applications, and resources run correctly on servers.
* Monitors server status and determines application health.
* Enables continuous auditing, transaction inspection, and control monitoring.

### 10. What Is the Role of AWS in DevOps?

AWS supports DevOps with the following features:

* **Flexible Services** – Provides ready-to-use services without installation/setup.
* **Scalability** – Manages a single instance or scales to thousands.
* **Automation** – Automates tasks and processes for efficiency.
* **Security** – Allows setting user permissions via AWS IAM.
* **Large Partner Ecosystem** – Supports a broad range of third-party integrations.

### 11. Name Three Important DevOps KPIs.

* **Mean Time to Failure Recovery (MTTR):** The average time to recover from failure.
* **Deployment Frequency:** How often deployments occur.
* **Percentage of Failed Deployments:** Number of deployment failures compared to successful ones.

### 12. Explain the Term "Infrastructure as Code" (IaC) in Configuration Management.

* Writing code to manage configuration, deployment, and provisioning automatically.
* Managing data centers with machine-readable definition files instead of manual configuration.
* Ensuring consistent provisioning of servers and infrastructure.
* Administering cloud environments through Infrastructure as a Service (IaaS).

### 13. How Is IaC Implemented Using AWS?

IaC replaces traditional script-based automation by allowing developers to write, test, and maintain infrastructure using JSON or YAML formats. AWS services like CloudFormation and Terraform help automate infrastructure deployment.

### 14. Why Has DevOps Gained Prominence Over the Last Few Years?

DevOps is widely adopted due to its success in companies like Facebook and Netflix:

* **Facebook** uses continuous deployment and code ownership models to scale efficiently.
* **Netflix** follows an automated DevOps approach to serve millions of users globally.
* **Benefits:** Increased release success rates, faster bug fixes, automated deployments, and reduced operational costs.

### 15. What Are the Fundamental Differences Between DevOps and Agile?

| **Characteristic** | **Agile** | **DevOps** |
| --- | --- | --- |
| **Work Scope** | Focuses only on agility | Requires both automation and agility |
| **Focus Area** | Prioritizes time and deadlines | Balances quality and time management |
| **Feedback Source** | Feedback from customers | Feedback from monitoring tools |
| **Practices Followed** | Uses Agile Kanban, Scrum, etc. | Uses CI/CD, Continuous Testing, etc. |
| **Release Cycles** | Shorter releases | Shorter releases with immediate feedback |
| **Agility** | Present in development | Present in both development and operations |

### 16. What Are the Anti-Patterns of DevOps?

Anti-patterns occur when organizations adopt ineffective practices blindly:

* **Cannot perform DevOps → Blame wrong people**
* **Developers do DevOps → Misinterpretation of roles**
* **DevOps solves all problems → Unrealistic expectations**
* **DevOps == Process** (Misconception)
* **DevOps == Agile** (Incorrect understanding)
* **Cannot perform DevOps → Organization is unique** (False assumption)
* **A separate DevOps team is required** (Not always necessary)

### 17. What Are the Benefits of Using Version Control?

* Enables multiple team members to work on files simultaneously.
* Tracks changes and maintains version history.
* Allows easy rollback to previous versions if necessary.
* Supports distributed repositories (e.g., Git), preventing data loss if the main server fails.

### 18. Describe the Branching Strategies You Have Used.

* **Release Branching:** A separate branch is created when a release is ready.
* **Feature Branching:** Each new feature is developed in a separate branch and merged after testing.
* **Task Branching:** Each task is developed in its own branch, with task keys in branch names for tracking.

### 19. Can You Explain the "Shift Left to Reduce Failure" Concept in DevOps?

* Shift left refers to integrating security, testing, and performance earlier in the development process.
* Instead of testing security only before deployment, security measures should be incorporated in earlier phases.
* This helps in early detection of vulnerabilities, reducing security risks.

### 20. What Is the Blue/Green Deployment Pattern?

* **Blue/Green deployment** is a technique to minimize downtime during releases.
* The old version (Blue) remains live while the new version (Green) is deployed.
* Once tested, traffic is switched from Blue to Green, ensuring a smooth transition.

### 21. What Is Continuous Testing?

Continuous Testing involves automated testing as part of the software delivery pipeline:

* Provides immediate feedback on software quality.
* Helps prevent bottlenecks between development and production.
* Ensures smooth CI/CD processes by integrating with automation tools.

### **Git Basics**

**Add or stage your changes.**

**Commit your changes with a comment.**

Now, you can merge the branches on the command line or push your changes to your remote repository on GitHub and merge them in a pull request.

### **Git Interview Questions**

#### **39. What is Git bisect? How can you use it to determine the source of a (regression) bug?**

Git bisect is a tool that uses binary search to locate the commit that triggered a bug.

##### **Git bisect command:**

sh

git bisect <subcommand> <options>

The git bisect command helps find the commit that introduced a bug using a binary search algorithm.

* The commit where the bug appears is called the **"bad" commit**.
* The commit before the bug occurs is called the **"good" commit**.

Git bisect picks a random commit between these two points and asks the user whether it is **good** or **bad**. This process continues until the exact commit causing the bug is found.

#### **40. Explain some basic Git commands.**

Below is a table summarizing some essential Git commands:

| **Command** | **Purpose** |
| --- | --- |
| git init | Initializes a new repository. |
| git config –global user.name “[name]”  git config –global user.email “[email address]” | Sets the username and email for commits. |
| git clone <repository\_path> | Creates a local copy of a repository. |
| git add <file\_name>  git add . | Stages one or more files. |
| git commit -m “<commit\_message>” | Saves changes to the repository. |
| git diff [branch1] [branch2] | Shows differences between two branches. |
| git status | Displays the status of changes in the working directory. |
| git rm <file\_name> | Removes files from the repository and stages the deletion. |
| git show <commit> | Displays details about a commit. |
| git branch | Lists all branches and highlights the current branch. |
| git branch [branch\_name] | Creates a new branch. |
| git branch -d [branch\_name] | Deletes a specified branch. |

### **DevOps Interview Questions for Continuous Integration: Jenkins**

#### **41. Explain the master-slave architecture of Jenkins.**

* **Jenkins Master:** Pulls code from the remote repository whenever there is a commit and distributes workloads to slave nodes.
* **Jenkins Slave:** Executes builds, tests, and generates reports based on requests from the master.

#### **42. What is a Jenkinsfile?**

A **Jenkinsfile** is a text file that contains the definition of a Jenkins pipeline and is stored in version control.

**Benefits of Jenkinsfile:**

* Enables code review and iteration on pipelines.
* Provides an audit trail for pipeline execution.
* Serves as a single source of truth for the pipeline.

#### **43. Which command runs Jenkins from the command line?**

**Correct Answer:**

sh

java –jar Jenkins.war

#### **44. Key concepts of Jenkins pipeline:**

* **Pipeline:** Defines the complete CI/CD workflow, including build, test, and deployment stages.
* **Node:** A machine within the Jenkins environment that executes a pipeline.
* **Step:** A single task in the pipeline execution.
* **Stage:** A logically distinct part of the pipeline (e.g., build, test, deploy).

#### **45. Which file is used to define dependencies in Maven?**

**Correct Answer:** pom.xml

### **Jenkins Pipelines**

#### **46. Two types of Jenkins pipelines**

| **Pipeline Type** | **Description** |
| --- | --- |
| **Scripted Pipeline** | Uses Groovy scripting and node blocks for execution. |
| **Declarative Pipeline** | Uses a structured, easy-to-read syntax to define pipeline stages. |

**Example Syntax for Declarative Pipeline:**

groovy

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

}

stage('Test') {

steps {

echo 'Testing...'

}

}

stage('Deploy') {

steps {

echo 'Deploying...'

}

}

}

}

#### **47. How to create a backup and copy files in Jenkins?**

* Backup the **JENKINS\_HOME** directory regularly.
* Copy a job directory to clone or replicate a job.

#### **48. How to migrate Jenkins from one server to another?**

* Copy the **job directory** from one installation to another.
* Clone an existing job by duplicating the directory.
* Rename an existing job by renaming its directory.

#### **49. Security mechanisms in Jenkins**

1. **Internal database:** Stores user credentials.
2. **LDAP integration:** Authenticates users via an LDAP server.
3. **Application server authentication:** Uses external authentication mechanisms.

#### **50. How to deploy a custom core plugin in Jenkins?**

1. Copy the .hpi file to $JENKINS\_HOME/plugins.
2. Remove the plugin's development directory.
3. Create an empty <plugin>.hpi.pinned file.
4. Restart Jenkins.

### **Selenium Interview Questions for Continuous Testing**

#### **55. What are the different Selenium components?**

1. **Selenium IDE:** A Firefox plugin for record-and-playback testing.
2. **Selenium RC:** Supports multiple languages for writing tests.
3. **Selenium WebDriver:** A more advanced tool that directly interacts with web browsers.
4. **Selenium Grid:** Enables parallel execution on multiple browsers.

#### **56. Different exceptions in Selenium WebDriver**

| **Exception** | **Description** |
| --- | --- |
| **TimeoutException** | Thrown when an operation takes too long. |
| **NoSuchElementException** | Occurs when an element is not found. |
| **ElementNotVisibleException** | Happens when an element is in the DOM but not visible. |
| **SessionNotFoundException** | Occurs when WebDriver is used after quitting the browser. |

#### **57. Can Selenium test an application on an Android browser?**

Yes, Selenium can test an application using **Selendroid** or **Appium**.

#### **58. Test types supported by Selenium**

1. **Functional Testing** – Verifies software functionality.
2. **Regression Testing** – Detects new defects in existing functionality.
3. **Load Testing** – Analyzes system performance under load.

#### **59. How to retrieve text from a web element?**

java

String text = driver.findElement(By.id("text")).getText();

#### **60. How to handle keyboard and mouse actions in Selenium?**

The **Actions** class in Selenium WebDriver is used for this:

java

Actions action = new Actions(driver);

action.clickAndHold(element).moveToElement(target).release().perform();

#### **61.** findElement() **vs.** findElements()

| **Method** | **Description** |
| --- | --- |
| findElement(By.xpath()) | Returns the first matching element. |
| findElements(By.xpath()) | Returns a list of all matching elements. |

#### **62. Difference between** driver.close() **and** driver.quit()

* driver.close(): Closes the current window.
* driver.quit(): Closes all browser windows and ends the WebDriver session.

#### **63. How to submit a form in Selenium?**

java

driver.findElement(By.id("ElementID")).submit();

### **Puppet Questions**

#### **74. Which open-source or community tools do you use to make Puppet more powerful?**

* Configuration changes are tracked using **Jira**, and further maintenance follows internal procedures.
* **Git** and **Puppet's code manager app** are used for version control.
* Changes pass through **Jenkins' continuous integration pipeline** before being deployed.

#### **75. What are the resources in Puppet?**

* **Resources** are the basic units in any configuration management tool.
* They define a node’s features, such as software packages or services.
* A **resource declaration** describes the action to be performed.
* When executed in a **catalog**, it ensures the node reaches its desired state.

#### **76. What is a class in Puppet?**

* **Classes** are named blocks in a manifest used to configure node functionalities (services, files, packages).
* Classes must be **explicitly invoked** in a node’s catalog.
* Example of a Puppet class:

puppet

class apache (String $version = ‘latest’) {

package { ‘httpd’:

ensure => $version,

before => File[‘/etc/httpd.conf’],

}

}

### **Ansible Questions**

#### **77. What is an Ansible role?**

* A **role** is an independent block of tasks, variables, files, and templates inside a playbook.
* Example: A playbook that installs **Tomcat on node1**.

#### **78. When should I use** {{ }} **in Ansible?**

* Use **{{ }}** for variables unless using **conditional statements** like when: ....
* Example:

yaml

echo “This prints the value of {{foo}}”

when: foo is defined

* This prevents Ansible from misinterpreting it as a dictionary declaration.

#### **79. What is the best way to make content reusable in Ansible?**

1. **Roles** – Manage tasks in a playbook, easily shareable via **Ansible Galaxy**.
2. **Include** – Adds a submodule/file to a playbook for reuse.
3. **Import** – A refined version of include, ensuring files are added only once.

#### **80. How is Ansible different from Puppet?**

| **Feature** | **Ansible** | **Puppet** |
| --- | --- | --- |
| Installation | Agentless | Agent-based |
| Language | Python | Ruby |
| Config Files | YAML | DSL |
| OS Support | No Windows Support | Supports all major OSs |

### **Docker Questions**

#### **81. Explain the architecture of Docker.**

* **Client-Server Model**:
  + **Docker Client** sends commands via REST API.
  + **Docker Daemon** processes requests and interacts with the OS.
  + **Docker Image** is a blueprint for containers.
  + **Docker Container** is an executable application package.
  + **Docker Registry** hosts and distributes images.

#### **82. What are the advantages of Docker over Virtual Machines?**

| **Feature** | **Virtual Machine** | **Docker** |
| --- | --- | --- |
| Memory Usage | High | Low |
| Boot Time | Slow | Fast |
| Performance | Unstable with multiple VMs | Stable |
| Scalability | Hard to scale | Easy to scale |
| Efficiency | Low | High |
| Portability | Limited | High |
| Data Sharing | Not possible | Possible |

#### **83. How do we share Docker containers with different nodes?**

* Use **Docker Swarm** to manage clusters of Docker nodes.
* **Manager Node** – Controls the swarm.
* **Worker Nodes** – Run the containers.

#### **84. What are the commands used to create a Docker Swarm?**

1. **Initialize Swarm on Manager Node:**

sh

docker swarm init --advertise-addr <MANAGER-IP>

1. **Add Worker Node to Swarm:**

sh

docker swarm join --token <TOKEN> <MANAGER-IP>:2377

#### **85. How do you run multiple containers as a single service?**

* Use **Docker Compose**, where containers interact while running in isolation.
* Define services in a **YAML file**.

#### **86. What is a Dockerfile used for?**

* Defines **instructions** for building a **Docker Image**.
* Once built, images are stored in a **Docker Registry**.

#### **87. Differences between Docker Images and Docker Containers**

| **Feature** | **Docker Image** | **Docker Container** |
| --- | --- | --- |
| Purpose | Template for containers | Running instance of an image |
| Creation | Built using Dockerfile | Created from images |
| Storage | Stored in Docker Hub | Stored in Docker Daemon |
| File System | Read-only | Read-Write |

#### **88. Can you use an alternative to YAML for Docker Compose?**

* **Yes, JSON** can be used:

sh

docker-compose -f docker-compose.json up

#### **89. How do you create a Docker container?**

* Pull an image from **Docker Hub** (e.g., MySQL).
* Run the container:

sh

docker run -t -i MySQL

* List running containers:

sh

docker ps

#### **90. Difference between a Registry and a Repository?**

| **Feature** | **Registry** | **Repository** |
| --- | --- | --- |
| Purpose | Hosts and distributes images | Collection of image versions |
| Storage | Docker Hub, Private Registry | Stored in Registry |
| Types | Default: Docker Hub | Public & Private |

#### **91. Cloud platforms supporting Docker**

* **AWS**, **Azure**, **GCP**, **Rackspace**

#### **92. Expose vs. Publish in Docker**

| **Feature** | **Expose** | **Publish** |
| --- | --- | --- |
| Usage | In Dockerfile | In docker run command |
| Purpose | Internal network port mapping | External port mapping |
| Example | EXPOSE 8080 | docker run -d -p 80:80 |

### **Continuous Monitoring - Nagios**

#### **93. How does Nagios help in continuous monitoring?**

* Monitors servers, applications, and services.
* Detects **failures**, **utilization**, and **performance issues**.

#### **94. What is Nagios Remote Plugin Executor (NRPE)?**

* Runs **Nagios plugins** on remote **Linux/Unix** machines.
* Components:
  + **check\_nrpe plugin** (on local machine)
  + **NRPE daemon** (on remote machine)

#### **95. What ports does Nagios use?**

* Default port: **5666**

#### **96. What are Active and Passive Checks in Nagios?**

| **Feature** | **Active Checks** | **Passive Checks** |
| --- | --- | --- |
| Trigger | Nagios itself | External application |
| Scheduling | Regularly scheduled | Event-based |
| Example | Service status check | Logs processing |

#### **97. Main Configuration File in Nagios**

* Located at:

sh

/usr/local/Nagios/etc/resource.cfg

#### **98. What is the Nagios Network Analyzer?**

* Provides network traffic analysis, security threat detection, and bandwidth insights.

#### **99. Benefits of HTTP & SSL Monitoring with Nagios**

| **Feature** | **HTTP Monitoring** | **SSL Monitoring** |
| --- | --- | --- |
| Detects | Outages, Failures | Expiry, Security Issues |
| Improves | Web Transactions | Website Security |

#### **100. What is State Stalking in Nagios?**

* **Logs state changes** of hosts and services.
* Helps in **debugging intermittent issues**.

### **Nagios Interview Questions**

#### **100. What is the Nagios Network Analyzer?**

Nagios Network Analyzer provides a comprehensive view of network traffic sources and security threats. It offers the following benefits:

* Provides an in-depth analysis of network traffic and bandwidth data.
* Allows system administrators to monitor the overall health of the network.
* Helps proactively resolve outages, abnormal behavior, and security threats before they impact critical business operations.

#### **101. What are the benefits of HTTP and SSL certificate monitoring with Nagios?**

##### **HTTP Certificate Monitoring:**

* Increases server, service, and application availability.
* Enables fast detection of network outages and protocol failures.
* Supports web transaction and web server performance monitoring.

##### **SSL Certificate Monitoring:**

* Ensures increased website availability.
* Enhances application availability.
* Provides improved security.

#### **102. Explain virtualization with Nagios.**

Nagios supports various virtualization platforms such as VMware, Microsoft Virtual PC, Xen, and Amazon EC2. It offers:

* Monitoring capabilities across different platforms.
* Quick detection of service and application failures.
* Ability to track key metrics, including:
  + CPU usage
  + Memory utilization
  + Network performance
  + Virtual machine status
* Reduced administrative overhead.

#### **103. Name the three variables that affect recursion and inheritance in Nagios.**

1. **name** – Defines the template name, which can be referenced in other object definitions for inheritance.
2. **use** – Specifies the name of the template object from which properties/variables should be inherited.
3. **register** – Indicates whether the object definition should be registered with Nagios (0 = no, 1 = yes).

**Example:**

yaml

CopyEdit

define someobjecttype {

object-specific variables...

name template\_name

use name\_of\_template

register [0/1]

}

#### **104. Why is Nagios said to be object-oriented?**

Nagios is object-oriented because it allows object definitions to inherit properties from other objects. This enables efficient configuration management.

**Types of Objects in Nagios:**

* Services
* Hosts
* Commands
* Time Periods

#### **105. What is state stalking in Nagios?**

State stalking in Nagios is used for logging purposes:

* When enabled for a host or service, Nagios closely monitors its state.
* Any changes in check results are logged.
* Helps in detailed analysis of log files for troubleshooting.

### **Version Control System (VCS) Interview Questions**

#### **106. What is a version control system (VCS)?**

A VCS is a tool that enables developers to manage changes to source code over time. It allows for:

* Tracking and managing different versions of code.
* Collaboration among developers.
* Reverting to previous versions when needed.

#### **107. What are the benefits of using a VCS?**

* Tracks changes to code over time.
* Facilitates collaboration and code sharing.
* Allows reverting to previous versions if necessary.
* Supports branching and working on different features or fixes simultaneously.
* Enables merging changes from other contributors.
* Improves confidence and control over code deployments.

#### **108. What are the types of VCS?**

1. **Centralized VCS (CVCS):**
   * Stores all versions of code in a central repository.
   * Developers check out files, make changes, and commit them back to the repository.
   * Example: Subversion (SVN).
2. **Distributed VCS (DVCS):**
   * Developers maintain local repositories with full code history.
   * Supports local commits before pushing changes to a remote repository.
   * Example: Git, Mercurial.

#### **109. What is the difference between Git and SVN?**

| **Feature** | **Git** | **SVN** |
| --- | --- | --- |
| Type | Distributed VCS | Centralized VCS |
| Flexibility | More flexible with branching & merging | Less flexible |
| Binary File Handling | Less optimized | Better for large binary files |
| Speed | Generally faster | Can be slower in some operations |

### **Virtualization Interview Questions**

#### **110. What is virtualization?**

Virtualization is a technology that enables multiple operating systems or applications to run on a single physical machine by creating virtual instances of hardware resources (CPU, memory, storage).

#### **111. What are the benefits of virtualization?**

* Reduces hardware costs.
* Increases efficiency and resource utilization.
* Improves scalability and flexibility.
* Enhances reliability and availability of applications.
* Simplifies IT infrastructure management.

#### **112. What are the different types of virtualization?**

1. **Server Virtualization** – Runs multiple OS instances on one physical server.
2. **Network Virtualization** – Creates virtual networks independent of physical infrastructure.
3. **Storage Virtualization** – Combines multiple storage resources into a virtual pool.
4. **Desktop Virtualization** – Runs multiple desktop environments on a single system.

#### **113. What is a hypervisor?**

A hypervisor is software that enables virtualization by allowing multiple virtual machines to share a single physical machine. It:

* Allocates resources (CPU, memory, storage) among virtual machines.
* Ensures isolation between different virtual machines.
* Examples: VMware ESXi, Microsoft Hyper-V, KVM.

#### **114. How does virtualization relate to DevOps?**

Virtualization plays a crucial role in DevOps by:

* Enabling rapid provisioning of development, testing, and production environments.
* Improving efficiency and cost savings.
* Allowing better scalability and flexibility in resource allocation.

#### **115. What are the benefits of using virtualization in DevOps?**

* **Improved efficiency** – Quick creation, deployment, and management of environments.
* **Greater scalability** – Easy scaling without additional hardware.
* **Increased flexibility** – Custom environments for specific needs.
* **Cost reduction** – Lower infrastructure costs through efficient resource utilization.

#### **116. What are common virtualization technologies used in DevOps?**

1. **Virtual Machines (VMs)** – Created using software like VMware or VirtualBox.
2. **Containers** – Lightweight, portable environments created using Docker or Kubernetes.
3. **Cloud Computing** – Virtualized infrastructure services from AWS, Azure, or GCP.