Viva Questions and Answers: DevOps and Related Topics

1. DevOps and Agile

What is DevOps, and how does it differ from traditional software development models?

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle and provide continuous delivery with high quality. It emphasizes collaboration between developers and operations teams, unlike traditional models where these functions were siloed.

Explain the core principles of DevOps.

The core principles of DevOps include: Continuous Integration/Continuous Deployment (CI/CD), automation, collaboration, monitoring, and feedback loops to improve efficiency and reliability.

What is the purpose of Continuous Integration (CI) and Continuous Deployment (CD) in DevOps?

CI ensures that code changes are frequently integrated into a shared repository, enabling early detection of issues. CD automates the deployment process, ensuring that software changes can be reliably and continuously pushed to production environments.

How does the Agile model support DevOps practices?

Agile focuses on iterative development and customer collaboration. DevOps complements Agile by ensuring that the software developed iteratively is also deployed quickly and reliably.

What are some popular tools used in a DevOps pipeline?

Some popular tools include Jenkins, Git, Docker, Kubernetes, Ansible, Puppet, Maven, and Selenium.

Can you explain the concept of ShiftLeft Testing?

ShiftLeft Testing involves moving testing earlier in the software development lifecycle. It helps in identifying and addressing issues early, thereby reducing costs and improving quality.

How do DevOps practices improve the software delivery lifecycle?

DevOps practices improve delivery by fostering collaboration, automating processes, reducing deployment time, and ensuring rapid and reliable software releases.

2. Git, GitHub, and Git Cheat Sheet

What is Git, and how is it different from GitHub?

Git is a version control system that helps track code changes. GitHub is a cloudbased platform that hosts Git repositories and facilitates collaboration.

Explain the purpose of the 'git clone' and 'git pull' commands.

'git clone' creates a copy of a remote repository on your local machine. 'git pull' updates your local repository with the latest changes from the remote repository.

What does 'git stash' do, and when would you use it?

'git stash' temporarily saves uncommitted changes so you can switch branches or work on something else without committing them.

What is a pull request, and how is it used in GitHub?

A pull request is a request to merge changes from one branch into another. It is used in GitHub for code reviews and collaboration.

Explain the difference between 'git merge' and 'git rebase'.

'git merge' integrates changes from one branch into another, creating a new commit. 'git rebase' moves or combines commits from one branch onto another, making the history linear.

How do you resolve conflicts in Git?

Conflicts occur when changes in two branches overlap. They can be resolved by manually editing the conflicting files and committing the changes.

What is the difference between 'git fetch' and 'git pull'?

'git fetch' downloads changes from the remote repository but doesn’t merge them. 'git pull' downloads and merges changes automatically.

Can you explain the significance of the .gitignore file?

The .gitignore file specifies which files or directories Git should ignore, preventing them from being tracked in the repository.

3. Jenkins and Maven

What is Jenkins, and how does it support CI/CD?

Jenkins is an opensource automation server that helps automate software development processes, including building, testing, and deploying code.

How do you set up a Jenkins pipeline?

A Jenkins pipeline can be set up using either a declarative or scripted approach, defining steps such as build, test, and deploy.

What are the key differences between a declarative and scripted Jenkins pipeline?

A declarative pipeline uses a simpler, predefined structure, while a scripted pipeline offers more flexibility but requires Groovy scripting knowledge.

What is Maven, and how is it used in Jenkins pipelines?

Maven is a build automation tool primarily used for Java projects. It manages dependencies and builds processes within Jenkins pipelines.

Can you explain the purpose of 'pom.xml' in a Maven project?

'pom.xml' is the configuration file in Maven projects that defines dependencies, plugins, and project information.

What are the different stages of a Jenkins pipeline?

The stages typically include build, test, and deploy, though they can vary depending on the project.

Here’s a detailed comparison between DevOps Developer and Software Developer roles:

Aspect	DevOps Developer	Software Developer
Primary Focus	Focuses on automating software development pipelines and deploying code efficiently.	Focuses on designing, coding, testing, and building software applications.
Responsibilities	Manages CI/CD pipelines, automates deployments, and monitors production environments.	Develops software features, writes code, and fixes bugs.
Key Skills	Automation, infrastructure as code (IaC), CI/CD tools (Jenkins, GitHub Actions, etc.), cloud platforms (AWS, Azure).	Programming languages (Java, Python, JavaScript, etc.), software design principles, and debugging.
Tools Used	Docker, Kubernetes, Jenkins, Ansible, Terraform, Git, Prometheus.	IDEs (Visual Studio Code, IntelliJ, etc.), Git, frameworks like React or Django.

| Collaboration | Works closely with operations teams to streamline deployment and delivery. | Collaborates with other developers and designers to build software. |

| Deployment | Focuses on deploying and managing the application in production and nonproduction environments. | Focuses on the development phase, writing code and meeting feature requirements. |

| Workflow Approach | Uses practices like CI/CD to ensure smooth, automated deployments. | Follows Agile, Scrum, or other development methodologies for feature building. |

| System Knowledge | Deep knowledge of cloud infrastructure, networking, and system operations. | Strong understanding of software architecture, data structures, and algorithms. |

| Role in Software Lifecycle | Ensures the code is properly integrated, tested, and deployed. Monitors and improves performance postdeployment. | Responsible for building features and ensuring code quality through testing. |

| Key Goal | Ensure seamless delivery and operational efficiency of software. | Build highquality software applications that meet customer requirements. |

| Challenges | Handling deployment errors, maintaining infrastructure stability, and monitoring system health. | Managing code complexity, meeting deadlines, and ensuring bugfree releases. |

Summary

Software Developers are primarily focused on writing code to build applications, while DevOps Developers are concerned with automating processes to ensure that code moves efficiently from development to production.

DevOps bridges the gap between development and operations, ensuring faster releases with high stability, while software developers focus on building the core functionality of applications.

In many modern teams, collaboration between these roles ensures smooth endtoend delivery of software.

Here are viva questions with concise answers for topics in DevOps and related tools:

1. DevOps and Agile

1. What is DevOps, and how does it differ from traditional software development models?

DevOps emphasizes collaboration between development and operations teams to automate software delivery. Traditional models focus on isolated phases (Dev → Ops).

2. Explain the core principles of DevOps.

Automation, collaboration, CI/CD, monitoring, and continuous feedback are key principles.

3. What is the purpose of Continuous Integration (CI) and Continuous Deployment (CD) in DevOps?

CI ensures code changes are integrated and tested frequently, while CD automates deployment to production.

4. How does the Agile model support DevOps practices?

Agile promotes frequent iterations and quick feedback, which align with DevOps goals of continuous delivery and automation.

5. What are some popular tools used in a DevOps pipeline?

Jenkins, Docker, Kubernetes, Git, Ansible, and Maven are common.

6. Can you explain the concept of ShiftLeft Testing?

It means testing earlier in the software lifecycle to identify defects early and reduce cost/time.

7. How do DevOps practices improve the software delivery lifecycle?

They automate repetitive tasks, reduce manual errors, and enable faster releases with better quality.

2. Git, GitHub, and Git Cheat Sheet

1. What is Git, and how is it different from GitHub?

Git is a version control system. GitHub is a platform for hosting Git repositories online.

2. Explain the purpose of the 'git clone' and 'git pull' commands.

`git clone`: Copies a remote repository locally.

`git pull`: Fetches changes and merges them into the local branch.

3. What does 'git stash' do, and when would you use it?

Saves uncommitted changes temporarily. Useful if you need to switch branches without committing.

4. What is a pull request, and how is it used in GitHub?

A pull request is a request to review and merge changes from one branch to another.

5. Explain the difference between 'git merge' and 'git rebase'.

Merge combines changes from two branches with a new merge commit.

Rebase integrates changes without a merge commit, keeping history linear.

6. How do you resolve conflicts in Git?

Manually edit conflicting files, stage the changes, and commit.

7. What is the difference between 'git fetch' and 'git pull'?

`git fetch`: Retrieves changes from the remote branch without merging.

`git pull`: Fetches and merges changes into the current branch.

8. Can you explain the significance of the .gitignore file?

Specifies files or directories that Git should ignore, like `node_modules/`.

3. Jenkins and Maven

1. What is Jenkins, and how does it support CI/CD?

Jenkins automates builds, tests, and deployments in a CI/CD pipeline.

2. How do you set up a Jenkins pipeline?

Define the pipeline stages in Declarative or Scripted syntax within a `Jenkinsfile`.

3. What are the key differences between a declarative and scripted Jenkins pipeline?

Declarative pipelines are easier to use with predefined structure. Scripted pipelines are more flexible.

4. What is Maven, and how is it used in Jenkins pipelines?

Maven is a build automation tool that manages dependencies and builds projects. Jenkins can invoke Maven in build steps.

5. Can you explain the purpose of 'pom.xml' in a Maven project?

It defines project dependencies, plugins, and configurations for Maven.

6. What are the different stages of a Jenkins pipeline?

Build, Test, Deploy, and Release stages.

4. CI/CD Pipeline and MasterSlave Architecture

1. What is a CI/CD pipeline, and why is it essential in DevOps?

Automates software delivery, ensuring quick and reliable builds, tests, and deployments.

2. Explain the stages of a typical CI/CD pipeline.

Source Code Management → Build → Test → Deploy.

3. What is the masterslave architecture in Jenkins?

Master manages jobs, and slaves (agents) run build tasks on multiple nodes.

4. How does load distribution work in a masterslave setup?

The master assigns jobs to slaves, distributing workloads across nodes.

5. What is the role of a build agent in a CI/CD pipeline?

It executes tasks like building or testing on the assigned node.

6. Can you describe how to secure a Jenkins pipeline?

Use rolebased access control (RBAC), credentials encryption, and restrict access to pipelines.

5. Selenium Basics

1. What is Selenium, and how is it used for testing?

   Selenium automates browser testing to validate web applications.

2. Explain the difference between Selenium WebDriver and Selenium IDE.

   WebDriver: Scriptbased testing across multiple browsers.

   IDE: A GUI tool for recording and running tests.

3. What are some challenges of automating tests with Selenium?

   Handling dynamic elements, browser compatibility, and synchronization issues.

4. How do you handle dynamic elements in Selenium tests?

   Use XPath, CSS selectors, and waits (implicit/explicit).

5. What is a test suite in Selenium, and how do you create one?

   A collection of tests grouped together. You define a suite in a testng.xml or JUnit class.

6. Docker and Docker Architecture

1. What is Docker, and how does it differ from virtual machines?

   Docker containers share the host OS while VMs have separate OS instances.

2. Explain Docker architecture and its key components.

Client, Docker Daemon, Registry, and Containers.

3. What are the basic Docker commands to run, stop, and remove containers?

`docker run`, `docker stop`, `docker rm`.

4. How do you create a Docker image from a Dockerfile?

Use `docker build t <image_name> .`.

5. What is the role of Docker Compose?

Manages multicontainer applications using a `dockercompose.yml` file.

6. Explain the concept of Docker networking.

Containers communicate over networks like bridge, host, and overlay.

7. Ansible and Puppet Basics

1. What is Ansible, and how is it different from Puppet?

Ansible is agentless, while Puppet uses agents on nodes.

2. What are playbooks in Ansible, and how are they structured?

Playbooks are YAML files defining tasks and roles.

3. Explain the concept of infrastructure as code (IaC) and how Ansible supports it.

IaC uses code to define infrastructure, and Ansible applies these configurations automatically.

4. What are Puppet manifests, and how are they used?

Manifests are Puppet's configuration files written in DSL.

5. How do you manage configuration drift with Puppet?

Puppet periodically checks and enforces configurations on nodes.

These viva questions cover essential topics, providing a balanced mix of theoretical knowledge and practical understanding.