Did You Know?

- The global DevOps market will hit $24.71 billion by 2027.
- DevOps will generate over $35 billion annually by 2030.
- Puppet's State of DevOps report shows that 8 out of 10 organizations have DevOps implementation ongoing.
- In its Global DevOps Survey for 2023, GitLab found that most DevOps engineers work in five roles: software engineer or developer, development/engineering manager/director, technology executive, DevOps manager or director, and DevOps engineer.
- The industry-standard tool for continuous integration and delivery, Jenkins, was originally called "Hudson" when it was first released in 2007.
- GitLab's survey found that 1 out of 4 organizations that practice DevOps are in the computer hardware or software services (including SaaS) industry.

Basic DevOps Interview Questions for Freshers

1. What is DevOps?

Characteristics	DevOps
Basic premise	A collaboration of development and operations teams. It is more of a cultural shift.
Related to	Agile methodology
Priorities	Resource management, communication, and teamwork
Benefits	Speed, functionality, stability, and innovation

2. How do you define the role of a DevOps engineer?

A DevOps engineer is responsible for bridging the gap between the development and operations teams by facilitating the delivery of high-quality software products. They use automation tools and techniques to streamline the software development lifecycle, monitor and optimize system performance, and ensure continuous deployment and delivery.

Moreover, they ensure that everything in the development and operations process runs smoothly.

3. How does DevOps differ from traditional software development methodologies?

DevOps differs from traditional software development methodologies in its collaboration, automation, and continuous delivery. Instead of having separate teams for development and operations, DevOps promotes cross-functional teams that work together to streamline the entire software development process.

DevOps also relies heavily on automation tools and techniques to accelerate development and ensure consistency, quality, and reliability in every lifecycle stage.

4. How does HTTP work?

HTTP or Hypertext Transfer Protocol works in a client–server model like most other protocols. HTTP provides a way to interact with web resources by transmitting hypertext messages between clients and servers.

5. In terms of development and infrastructure, mention the core operations of DevOps.

The core operations of DevOps include:

- Development
- Version Control
- Testing
- Integration
- Deployment
- Delivery
- Configuration
- Monitoring
- Feedback

6. What are some technical and business benefits of DevOps?

Technical benefits:

- Continuous software delivery
- Less complex problems to fix
- Faster bug resolution

Business benefits:

- Faster delivery of features for customer satisfaction
- More stable operating environments
- More time available to add product value

7. What is CI? What is its purpose?

CI stands for continuous integration. It is a software development process where developers frequently merge code changes into a central repository.

CI aims to detect issues early and ensure the codebase always works. By integrating code changes often, developers can identify and fix problems faster. This makes building, testing, and deploying the software easier.

8. Name three important DevOps KPIs.

- Three of the most common DevOps KPIs are:
    - Meantime to failure recovery
    - Deployment frequency
    - Percentage of failed deployments

Intermediate DevOps Interview Questions

9. Name some of the most important DevOps tools.

This is a list of the most important DevOps Tools being used:

- Git
- Maven
- Selenium
- Jenkins
- Docker
- Puppet
- Chef
- Ansible
- Nagios

10. What are the core principles of DevOps?

The core principles of DevOps include continuous integration, continuous delivery, automation, collaboration, and monitoring. DevOps teams work together to automate as much of the software development process as possible, making it faster, more reliable, and more efficient.

They also collaborate closely to ensure everyone is on the same page and continuously working to improve the software through continuous monitoring and feedback loops.

11. How do you measure the success of DevOps implementation?

To evaluate the success of a DevOps implementation, we can use key indicators such as the frequency of changes, the speed of implementation, error recovery time, and the incidents of issues arising from changes. These metrics enable us to assess the efficiency and effectiveness of our software development process. We can also ask for feedback from team members and clients to measure the satisfaction level with the software and its functionality.

12. What is continuous delivery?

Continuous delivery is a software development practice that automates the release process. Its primary objective is to ensure that software is consistently prepared for deployment to production. This means that code changes are frequently integrated, built, tested, and deployed in a repeatable and automated manner. It helps the teams to reduce the time and effort required to release software while maintaining high quality and reliability. With continuous delivery, teams can release software more frequently and confidently.

13. What is the difference between continuous deployment and continuous delivery?

Continuous deployment is a software development practice where code changes are automatically deployed to production as soon as they pass the necessary tests and checks. Continuous delivery, on the other hand, focuses on automating the software release process but allows for manual intervention to determine when and how code changes are deployed to production.

 Untitled

While continuous deployment is fully automated, continuous delivery provides the flexibility to choose when and how to release code changes, even though the process is automated.

14. What is infrastructure as code?

Infrastructure as code (IaC) is a practice where infrastructure is defined and managed using code. Instead of manually configuring servers and infrastructure components, IaC allows teams to automate the provisioning, configuration, and deployment of infrastructure using code.

This approach helps teams achieve consistency, reduce errors, and increase speed and efficiency. IaC also enables teams to version control their infrastructure code, making it easier to track changes and collaborate.

15. Is DevOps a tool?

- DevOps can’t be referred to as a tool; it is a collaborative work culture that combines development and operations teams for continuous development, testing, integration, deployment, and monito

16. What is a DevOps pipeline?

A DevOps pipeline is a series of automated procedures that enable software development teams to quickly and efficiently develop, test, and deliver applications. A pipeline often has numerous steps, including code analysis, building, testing, packaging, and deployment. The pipeline’s stages are all automated, and changes to the code are always carried over from one stage to the next.

17. How would you ensure your DevOps pipeline is scalable and can handle increased demand?

You can take the following actions to ensure a DevOps pipeline is scalable and can cope with rising demand:

- Utilize cloud-based resources and tools to aid scalability
- Improve the scalability of the pipeline’s architecture and design
- Use orchestration and automation to increase efficiency
- Track pipeline performance and find areas for improvement using performance monitoring and analytics
- Test and validate the pipeline frequently to ensure it can handle rising demand and workloads.

Advanced DevOps Interview Questions for Experienced

18. What is the role of collaboration in DevOps?

DevOps requires collaboration since it encourages teamwork and communication between the development, operations, and other teams. DevOps teams can break down the barriers to information and improve the software development process by cooperating and exchanging information. The DevOps culture places immense value on collaboration, ensuring everyone works towards the same goals.

19. What are the core operations of DevOps in terms of development and infrastructure?

The core operations of DevOps are application development, version control, unit testing, deployment with infrastructure, monitoring, configuration, and orchestration.

20. How do you integrate security testing into your DevOps pipeline?

It is essential to use security tools that can automatically scan the code and infrastructure for vulnerabilities to integrate security testing into a DevOps pipeline. This can include devices such as static analysis security testing and dynamic application security testing.

These tools can be integrated into the build process to test for security issues and provide feedback to developers automatically. Additionally, it is essential to involve security experts early in the development process to identify potential security risks and ensure that security is integrated into every pipeline stage.

21. What role does automation play in ensuring security in a DevOps environment?

Automation plays a critical role in ensuring security in a DevOps environment. Automating security checks and testing makes detecting and fixing potential vulnerabilities and threats easier. Automation can also help ensure the consistent application of security policies across the entire pipeline, reducing the risk of human error.

Additionally, automation can enable continuous monitoring and threat detection, allowing for a rapid response to any possible security incidents. Overall, automation helps to create a more secure and resilient DevOps environment.

DevOps Interview Questions for 2 Years Experience

22. What is the importance of continuous feedback in DevOps?

Continuous feedback is essential in DevOps as it helps teams identify and address issues early, improving the quality and speed of software delivery. By collecting feedback throughout the software development process, teams can continuously monitor the performance of their applications and infrastructure and make improvements as needed.

It also helps to ensure that the software meets the needs of its users, resulting in better customer satisfaction and higher business value. Continuous feedback is a key element of the DevOps culture and promotes a mindset of continuous learning and improvement.

23. How do you ensure that both development and operations teams adopt DevOps practices?

You can practice the following to ensure development and operations teams adopt DevOps practices:

- Establish clear goals and objectives for the DevOps initiative
- Encourage collaboration and communication between teams
- Provide training and resources to help teams learn and adopt DevOps practices
- Use metrics and KPIs to measure progress and identify areas for improvement
- Foster a culture of continuous learning and improvement

- Promote ownership and accountability among team members
- Celebrate successes and share lessons learnt to build momentum and support for DevOps practices

24. How do you ensure your infrastructure is secure in a DevOps environment?

To ensure that infrastructure is secure in a DevOps environment, you can take the following steps:

- Implement DevOps best security practices in the development process
- Use automated security testing tools to identify vulnerabilities
- Monitor infrastructure continuously to detect and respond to security threats promptly
- Employ role-based access control to restrict access to sensitive information
- Regularly update and patch systems and software to address known vulnerabilities
- Establish incident responses and disaster recovery plans to minimize the impact of security incidents.

25. Can you describe the role of security in a DevOps environment?

Security is a key element integrated into the entire software development lifecycle in a DevOps context. Security ensures the software is created to comply with rules and defend against any security risks.

DevOps teams should include professionals who are knowledgeable about the most current security standards, can spot security threats, and can put the best security practices into action. This guarantees that the program is secure from creation and constantly watched throughout the deployment and maintenance phases.

26. How do you ensure your DevOps tools and technologies are up-to-date and secure?

- The success of every DevOps project depends on maintaining the most recent and secure versions of DevOps tools and technology. Regularly checking for updates and security patches for all the tools is crucial to ensuring this. You can accomplish this by joining the vendor’s email lists or following their social media accounts.It is also advised to employ security testing tools like vulnerability scanners and penetration testing programs to find and fix any security problems.

27. What are the key security considerations when building a DevOps pipeline?

Security should be considered throughout the development lifecycle when building a DevOps pipeline. Some key security considerations include ensuring that code is written securely, using secure and up-to-date software dependencies, and implementing security testing and vulnerability scanning tools.

Access control and authentication measures should also be implemented, and security incidents should be quickly identified and addressed.

28. Can you describe the process for securing application containers in a DevOps environment?

In a DevOps environment, securing application containers is critical to preventing cyber threats. The process typically starts with selecting a secure container registry and implementing a continuous integration and continuous deployment (CI/CD) pipeline, including vulnerability scanning and image signing.

Once the container images are deployed, access control and network segmentation are used to limit their exposure to potential threats. Regular security scanning, patching, monitoring, and logging are essential to

maintaining container security.

29. Why are monitoring and logging important in DevOps?

In a DevOps environment, monitoring and logging are vital because they give insight into the system's functionality, effectiveness, and security. While logging enables analysis of system events and actions, monitoring assists in identifying any issues before they become critical.

Identifying the root cause of problems simplifies the process of resolution and helps prevent their recurrence. Additionally, monitoring and logging offer insights into user behavior and usage patterns, facilitating better optimization and decision-making.

30. How does DevOps help in reducing the time-to-market for software products?

DevOps combines software development and IT operations to deliver software products quickly and efficiently. DevOps reduces time-to-market by promoting automation, collaboration, Agile development, and infrastructure as code. Automation tools reduce manual testing and deployment, and collaboration helps identify and resolve issues quickly. Agile development allows for rapid feedback and infrastructure as code automates infrastructure management.

31. What is the difference between a container and a virtual machine?

Virtual machines (VMs) and containers are two different approaches to running software. Applications can execute in a consistent environment across several systems due to containers, which are portable, lightweight environments that share the host system's resources.

A virtual machine (VM) is an operating system that operates entirely independently on top of a host machine while utilizing its resources, such as memory and CPU.

32. How does automation help in DevOps?

Automation plays a crucial role in DevOps by streamlining software development. It eliminates repetitive and time-consuming tasks like testing and deployment, leading to faster time-to-market, enhanced software quality, and reduced chances of human error. Automation tools are instrumental in achieving these benefits by automating various aspects of the software development process.

Automation also increases efficiency, scalability, and reliability, enabling teams to focus on innovation and value creation.

DevOps Interview Questions for 3 Years Experience

33. Can you discuss the importance of identity and access management in a DevOps environment?

Identity and access management is critical in a DevOps environment to ensure that only authorized personnel can access the systems and resources. It helps manage users' identities, access to resources, and permissions.

Identity and access management ensures that users have access to the resources required for their roles and responsibilities. It also helps detect and prevent unauthorized access and ensures access requests are verified and authenticated.

34. How do you approach testing in a DevOps environment?

In a DevOps environment, testing is integrated into the software development process. Automated testing tools ensure that code changes are thoroughly tested before deployment and that feedback is continuously provided to developers.

Testing is performed continuously throughout the software development lifecycle, allowing for early detection of issues and faster delivery of high-quality software.

35. Can you tell me what continuous testing and automation testing are?

Automation testing, as the name suggests, is a process of automating the manual testing process. It involves using separate testing tools that let developers create test scripts that can be executed repeatedly without manual intervention.

Continuous testing is the process of executing automated tests as part of the software delivery pipeline in DevOps. Each build is tested continuously in this process, allowing the development team to get fast business feedback to prevent the problems from progressing to the next stage of the software delivery lifecycle. This will dramatically speed up a developer's workflow. They no longer need to manually rebuild the project and re-run all tests after making changes.

36. What are the benefits of automation testing?

Automation testing has several advantages, including quicker and more effective testing, expanded coverage, and higher test accuracy. It can save time and money in the long run because automated testing can be repeated without human intervention.

37. Can one consider DevOps as an Agile methodology?

DevOps can be considered complementary to the Agile methodology but not entirely similar.

DevOps Interview Questions for 5 Years Experience

38. What is the role of continuous testing in DevOps?

Continuous testing is a critical component of DevOps that involves testing early and often throughout the software development process. It provides continuous feedback to developers, ensuring that code changes are tested thoroughly and defects are detected early. Continuous testing helps improve software quality, reduce costs and time-to-market, and increase customer satisfaction.

39. What is the role of cloud computing in DevOps?

Cloud computing plays a vital role within the realm of DevOps, as it offers a versatile and scalable infrastructure for software development and deployment. Its provision of computing resources on demand,

which are easily manageable and provisional, is instrumental in empowering DevOps teams. By leveraging cloud services, these teams are able to automate the deployment process, collaborate effectively, and seamlessly integrate various DevOps practices.

With an extensive range of services such as virtual machines, containers, serverless computing, and managed databases, cloud computing empowers teams to expedite software delivery, enhance scalability, and reduce infrastructure expenses. It significantly augments the overall DevOps lifecycle by facilitating faster development and deployment cycles.

40. How can DevOps help improve customer satisfaction and drive business growth?

DevOps can increase customer satisfaction and drive business growth by providing better software faster. DevOps teams can offer features that satisfy customers' expectations quickly by concentrating on collaboration, continuous improvement, and customer feedback. It can result in more loyal consumers and, ultimately, the company's growth.

41. How can DevOps help organizations reduce the risk of software failures?

DevOps reduces the risk of software failures by promoting collaboration between teams, continuous testing, and monitoring. It also encourages a culture of learning from mistakes to address issues quickly and prevent them from happening again.

42. What are some common misconceptions about DevOps?

One common misconception about DevOps is that it is solely focused on tools and automation. In reality, it is a cultural and organizational shift that involves collaboration between teams and breaking down barriers.

Another misconception is that DevOps is only for startups or tech companies, when it can be beneficial for any organization looking to improve its software development and delivery processes.

People consider DevOps the responsibility of the IT department, but it requires buy-in and involvement from all levels of the organization.

43. How can DevOps affect culture change to improve the business?

DevOps can positively affect business culture by encouraging team collaboration, enhancing communication, and promoting a culture of continuous learning and improvement. This cultural change can result in delivering software products and services that are quicker and more effective, with greater customer satisfaction. Adopting DevOps practices enables organizations to develop a culture that values adaptability, creativity, and teamwork, ultimately improving business results.

44. Our team has some ideas and wants to turn those ideas into a software application. Now, as a manager, I am confused about whether I should follow the Agile work culture or DevOps. Can you tell me why I should follow DevOps over Agile?

According to the current market trend, instead of releasing big sets of features in an application, companies are launching small features for software with better product quality and quick customer feedback, for high customer satisfaction. To keep up with this, we have to:

- Increase the deployment frequency with utmost safety and reliability

- Lower the failure rate of new releases
- Shorten the bug resolution time

DevOps fulfills all these requirements for fast and reliable development and deployment of software. Companies like Amazon and Google have adopted DevOps and are launching thousands of code deployments per day. But Agile, on the other hand, only focuses on software development.

45. How is DevOps different from Agile methodology?

DevOps and Agile are two methodologies that aim to improve software development processes. Agile focuses on delivering features iteratively and incrementally, while DevOps focuses on creating a collaborative culture between development and operations teams to achieve faster delivery and more reliable software.

DevOps also emphasizes the use of automation, continuous integration and delivery, and continuous feedback to improve the software development process. While Agile is primarily focused on the development phase, DevOps is focused on the entire software development lifecycle, from planning and coding to testing and deployment.

46. What is the role of configuration management in DevOps?

The major advantages of configuration management are given below:

- It enables us to manage the configurations on multiple systems.
- It allows us to standardize the configurations on all systems in a cluster.
- It helps us in the administration and management of multiple servers in the architecture.

47. What is the role of AWS in DevOps?

AWS in DevOps works as a cloud provider, and it has the following roles:

- Flexible services: AWS provides us with ready-to-use resources for implementation.
- Scaling purpose: We can deploy thousands of machines on AWS, depending on the requirement.
- Automation: AWS helps us automate tasks using various services.
- Security: Using its security options (IAM), we can secure our deployments and builds.

48. Can you list certain key performance indicators (KPIs) that are used to gauge the success of DevOps?

- Deployment Frequency: It is the number of times an application is deployed in a specific period.
- Mean Time to Recovery (MTTR): It is the average time taken to restore a failed system.
- Lead Time: It is the time taken from code commit to production release.
- Change Failure Rate: It is the percentage of changes that cause issues or failures.
- Time to Detect and Respond (TTDR): It is the average time taken to detect and respond to incidents.

49. What can be a preparatory approach for developing a project using the DevOps methodology?

Understanding the project’s objectives, specifications, and deadlines is essential to developing a project using the DevOps process. All stakeholders should be included in the planning process, and means for

communication should be established.

Teams can also create a culture of collaboration, automate crucial procedures, and use a continuous improvement strategy. On the basis of the particular requirements and objectives of the project, it is also crucial to choose the right tools and technologies.

50. What are the various branching strategies used in version control systems?

Several branching strategies are used in version control systems, including trunk-based development, feature branching, release branching, and git-flow. Trunk-based development involves committing changes directly to the main branch, while feature branching involves creating a new branch for each new feature.

Release branching involves creating a separate branch for each release, and git-flow combines feature and release branches to create a more structured branching strategy. Each strategy has its advantages and disadvantages, and the choice of strategy depends on the specific needs of the project and the team.

51. What are the benefits of using version control?

- It helps improve the collaborative work culture: Here, team members are allowed to work freely on any file at any time. The version control system allows us to merge all changes into a common version.
- It keeps different versions of code files securely: All the previous versions and variants of code files are neatly packed up inside the version control system.
- It understands what happened: Every time we save a new version of our project, the version control system asks us to provide a short description of what was changed. More than that it allows us to see what changes were made in the file's content, as well as who has made those changes.
- It keeps a backup: A distributed version control system like Git allows all team members to access the complete history of the project file so that in case there is a breakdown in the central server, they can use any of their teammate's local Git repository.

52. What are some common integration challenges that can arise when using multiple DevOps tools?

When using multiple DevOps tools, integration challenges can arise, such as data incompatibility, conflicting configurations, and a lack of communication between tools. For example, a deployment tool may not be compatible with a monitoring tool, or a configuration management tool may have different settings from the testing tool.

Organizations can use integration frameworks, middleware, and APIs to facilitate communication between tools and ensure data consistency to overcome these challenges. It is also important to establish clear communication channels between teams using different tools to ensure effective collaboration.

DevOps Interview Questions for Source Code Management: Git

53. Can you tell me the advantages of using Git?

- Data redundancy and replication

- High availability
- Only one Git directory per repository
- Superior disk utilization and network performance
- Collaboration friendly
- Can be used for any sort of project

54. Are git fetch and git pull the same?

The command 'git pull' pulls any new commits from a branch from the central repository and then updates the target branch in the local repository.

But, 'git fetch' is a slightly different form of 'git pull'. Unlike 'git pull', it pulls all new commits from the desired branch and then stores them in a new branch in the local repository.

In order to reflect these changes in your target branch, 'git fetch' must be followed with a 'git merge'. The target branch will only be updated after merging with the fetched branch (where we performed 'git fetch'). We can also interpret the whole thing with an equation like this:

55. How do you handle merge conflicts in Git?

In order to resolve merge conflicts in Git, we need to follow three steps:

- Understand what happened: It could be because of the same line being edited on the same file; it could be because of deleting some files, or it could also be because of files with the same file names. You can check everything by using 'git status'.
- Mark and clean up the conflict: When we open the files using the merge tool, Git marks the conflicted area like this '<<<<< HEAD' and ' >>>>> [other/branch/name]'.
- Perform the commit again and then merge the current branch with the master branch.

56. Can you tell me some advantages of the forking workflow over other Git workflows?

- The forking workflow is fundamentally different from other Git workflows. Instead of using a single server-side repository to act as the 'central' codebase, the forking workflow gives every developer their server-side repositories. This workflow is most often seen in public open-source projects.The main advantage is that contributions can be integrated without the need for everyone to push to a single central repository to maintain a clean project history. Developers can push to their server-side repositories, and only the one who maintains the project will push it to the official repository.As soon as developers are ready to publish their local commits, they will push their commits to their public repositories. Then, they will perform a pull request from the main repository which notifies the project manager that an update is ready to be integrated.

57. When do you use 'git rebase' instead of 'git merge'?

Both 'git rebases' and 'git merge' commands are designed to integrate changes from one branch into another branch: just that they just do it in different ways.

When we perform rebase of a feature branch onto the master branch, we move the base of the feature branch to the master branch's ending point.

By performing a merge, we take the contents of the feature branch and integrate them with the master branch. As a result, only the master branch is changed, but the feature branch history remains the same. Merging adds a new commit to your history.

Rebasing will create inconsistent repositories. For individuals, rebasing makes a lot of sense. Now, in order to see the history completely, the same way as it has happened, we should use merge. Merge preserves history, whereas rebase rewrites it.

58. I just made a bad git commit and made it public, and I need to revert the commit. Can you suggest how to do that?

Here we need to use the Git command:

git revert

This command can revert any command just by adding the commit ID.

59. Can you tell me how to squash the last n commits into a single commit? Is it even possible?

To squash the last n commits into a single commit, we can use:

DevOps Interview Questions for Continuous Integration: Jenkins

60. I want to move or copy Jenkins from one server to another. Is it possible? If yes, how?

I would suggest copying the Jenkins jobs directory from the old server to the new one. We can just move a job from one installation of Jenkins to another by copying the corresponding job directory.

Or, we can also make a copy of an existing Jenkins job by making a clone of that job directory in a different name.

Another way is that we can rename an existing job by renaming the directory. But, if you change the name of a job, you will need to change any other job that tries to call the renamed job.

61. I have 40 jobs in the Jenkins dashboard and I need to build them all at once. Is it possible?

Yes, it is. With the help of a Jenkins plugin, we can build DevOps projects one after the other. If one parent job is carried out, then automatically other jobs are also run. We also have the option to use Jenkins Pipeline jobs for the same.

62. How will you secure Jenkins?

The way to secure Jenkins is as follows:

- Ensure that global security is on

- Check whether Jenkins is integrated with the company's user directory with an appropriate plugin
- Make sure that the Project matrix is enabled to fine-tune access
- Automate the process of setting rights or privileges in Jenkins with a custom version-controlled script
- Limit physical access to Jenkins data or folders
- Periodically run security audits

63. Can you please tell me how to create a backup and copy files in Jenkins?

To create a backup, all we need to do is periodically back up our JENKINS_HOME directory. This contains all of the build configurations of our job, our slave node configurations, and our build history. To create a backup of our Jenkins setup, just copy this directory. We can also copy a job directory to clone or replicate a job or rename the directory.

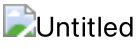
64. What are Jenkins Pipeline and CI/CD Pipeline?

Jenkins Pipeline can be defined as a suite of plugins supporting both the implementation and integration of Jenkins' continuous delivery pipeline.

Continuous integration or continuous delivery pipeline consists of build, deploy, test, and release. The pipeline feature is very time-saving. In other words, a pipeline is a group of build jobs that are chained and integrated in a sequence.

DevOps Interview Questions for Continuous Testing: Selenium

65. How to launch a browser using WebDriver?



66. Explain the different Selenium components.

Following are the different components of Selenium:

- Selenium Integrated Development Environment (IDE) – The Selenium IDE consists of a simple framework and comes with a Firefox plug-in that can be easily installed. This Selenium component should be used for prototyping.
- Selenium Remote Control (RC) – It is a testing framework for developers and QA that supports coding in any programming language like Java, PHP, C#, Perl, etc. This helps automate the UI testing process of web applications against any HTTP website.
- Selenium WebDriver – It has a better approach to automating the testing process of web-based applications and does not rely on JavaScript. This web framework allows cross-browser tests to be performed.
- Selenium Grid – This proxy server works with Selenium RC, and with the help of browsers, it is able to run parallel tests on different nodes or machines.

67. What are the testing types supported by Selenium?

Selenium supports regression testing and functional testing.

68. Are there any technical challenges with Selenium?

- It supports only web-based applications.
- It does not support the Bitmap comparison.
- No vendor support is available for Selenium compared to commercial tools like HP UFT.
- As there is no object repository concept, the maintainability of objects becomes very complex.

69. When should I use Selenium Grid?

Selenium Grid can be used to execute the same or different test scripts on multiple platforms and browsers, concurrently, in order to achieve distributed test execution. It allows testing under different environments, remarkably saving execution time.

70. Describe the difference between driver.close() and driver.quit().

The driver.close command closes the focused browser window. But, the driver.quit command calls the driver.dispose method which closes all browser windows and also ends the WebDriver session.

DevOps Chef Interview Questions

71. Why are SSL certificates used in Chef?

SSL certificates are used in Chef to establish secure and encrypted communication channels between Chef components and nodes. These certificates verify the authenticity of Chef servers and nodes, ensuring secure data transmission. By encrypting communication, SSL certificates protect sensitive information, such as authentication credentials and configuration data, from unauthorized access or tampering. This enhances the overall security of the Chef infrastructure and helps maintain the integrity and confidentiality of the data being exchanged.

72. How does Chef differ from other configuration management tools like Puppet and Ansible?

Chef differs from other configuration management tools like Puppet and Ansible in its approach to infrastructure automation. While Puppet and Ansible rely on a procedural approach, Chef uses a declarative approach, which means that users define the desired state of their infrastructure, and Chef ensures that it remains in that state. Additionally, Chef has a strong focus on testing and compliance, making it a popular choice in enterprise environments with strict security and compliance requirements.

73. What are the key components of a Chef deployment?

The key components of a Chef deployment include the Chef Server, which acts as the central hub for storing configuration data and Chef code; Chef Client, which runs on each node and applies the configurations defined by the Chef code; and the Chef Workstation, which is used by developers and administrators to write and test the Chef code before pushing it to the Chef Server for deployment. Other important components include cookbooks, recipes, and resources, which define the desired state of the infrastructure and the actions needed to achieve it.

DevOps Puppet Interview Questions

74. What are Puppet Manifests?

Every Puppet Node or Puppet Agent has got its configuration details in Puppet Master, written in the native Puppet language. These details are written in a language that Puppet can understand and are termed as Puppet Manifests. These manifests are composed of Puppet codes, and their filenames use the .pp extension.

For instance, we can write a manifest in Puppet Master that creates a file and installs Apache on all Puppet Agents or slaves that are connected to the Puppet Master.

75. How can I configure systems with Puppet?

In order to configure systems in a client or server architecture with Puppet, we need to use the Puppet Agent and the Puppet Master applications. In stand-alone architecture, we use the Puppet apply application.

76. What is a Puppet Module? How is it different from the Puppet Manifest?

A Puppet Module is nothing but a collection of manifests and data (e.g., facts, files, and templates). Puppet Modules have a specific directory structure. They are useful for organizing the Puppet code because, with Puppet Modules, we can split the Puppet code into multiple manifests. It is considered best practice to use Puppet Modules to organize almost all of your Puppet Manifests.

Puppet Modules are different from Puppet Manifests. Manifests are nothing but Puppet programs, composed of the Puppet code. File names of Puppet Manifests use the .pp extension.

77. Can you tell me what a Puppet codedir is?

It is the main directory for code and data in Puppet. It consists of environments (containing manifests and modules), a global modules directory for all the environments, and your Hiera data.

78. Where do you find codedir in Puppet?

It is found at one of the following locations:

DevOps Ansible Interview Questions

79. How does Ansible work?

Ansible is an open-source automation tool which is categorized into two types of servers:

- Controlling machines
- Nodes

Ansible will be installed on the controlling machine and using that, machine nodes are managed with the help of SSH. Nodes' locations are specified by inventories in that controlling machine.

Since Ansible is an agentless tool, it doesn't require any mandatory installations on remote nodes. So, there is no need for background programs to be executed while it is managing any nodes.

Ansible can handle a lot of nodes from a single system over an SSH connection with the help of Ansible Playbooks. Playbooks are capable of performing multiple tasks, and they are in the YAML file format.

80. Sometimes, we use ad-hoc commands instead of Playbooks in Ansible. Can you tell me what's the difference between Ansible Playbook and an ad-hoc command? Also, cite when to use them.

Ad-hoc commands are used to do something quickly, and they are mostly for one-time use. Whereas Ansible Playbook is used to perform repeated actions. There are scenarios where we want to use ad-hoc commands to perform a non-repetitive activity.

81. Why should I use Ansible?

Ansible can help in:

- Configuration Management
- Application Deployment
- Task Automation

82. What are handlers in Ansible?

Handlers in Ansible are just like regular tasks inside an Ansible Playbook but they are only run if the task contains a 'notify' directive. Handlers are triggered when it is called by another task.

83. Have you heard about Ansible Galaxy? What does it do?

Yes, I have. Ansible Galaxy refers to the 'Galaxy website' by Ansible, where users share Ansible roles. It is used to install, create, and manage Ansible roles.

84. What are the benefits of Ansible?

Ansible is an open-source configuration management tool that helps us in the following:

- Automating tasks
- Managing configurations
- Deploying applications
- Efficiency

85. What are the prerequisites to install Ansible 2.8 on Linux?

To install Ansible 2.8 on Linux, Security-Enhanced Linux (SELinux) has to be enabled and Python 3 has to be installed on remote nodes.

DevOps Scenario-Based Interview Questions

86. Can you write the syntax for building a docker image?

To build a docker image, we use the following command:

 Untitled

87. How can Docker containers be shared with different nodes?

Docker containers can be shared on different nodes with the help of the Docker Swarm. IT developers and administrators use this tool for the creation and management of a cluster of swarm nodes within the Docker platform. A swarm consists of a worker node and a manager node.

88. What are the advantages of Docker over virtual machines?

Below are the differences in multiple criteria that show why Docker has advantages over virtual machines.

Memory Space – In terms of memory, Docker occupies less space than a virtual machine.

Boot-up Time – Docker has a shorter boot-up time than a virtual machine.

Performance – Docker containers show better performance as they are hosted in a single Docker engine, whereas performance is unstable if multiple virtual machines are run.

Scaling – Docker is easy to scale up compared to virtual machines.

Efficiency – The efficiency of docker is higher, which is an advantage over virtual machines.

Portability – Docker doesn't have the same cross-platform compatibility issues with porting as virtual machines do.

Space Allocation – Data volumes can be shared and used repeatedly across multiple containers in Docker, unlike virtual machines that cannot share data volumes.

89. What is the concept of sudo in Linux?

Sudo is a program for Unix/Linux-based systems that provides the ability to allow specific users to use specific system commands at the system's root level. It is an abbreviation of 'superuser do', where 'super user' means the 'root user'.

90. Can you tell me the purpose of SSH?

SSH is nothing but a secure shell that allows users to log in with a secure and encrypted mechanism into remote computers. It is used for encrypted communications between two hosts on an unsafe network. It supports tunneling, forwarding TCP, and also transferring files.

91. What is NRPE in Nagios?

NRPE stands for 'Nagios Remote Plugin Executor'. As the name suggests, it allows you to execute Nagios plugins remotely on other Linux or Unix machines.

Nagios

It can help monitor remote machine performance metrics such as disk usage, CPU load, etc. It can communicate with some of the Windows agent add-ons. We can execute scripts and check metrics on

remote Windows machines as well.

92. Can you tell me why I should use Nagios?

- To plan for infrastructure upgrades before outdated systems fail
- To respond to issues quickly
- To fix problems automatically when detected
- To coordinate with the responses from the technical team
- To ensure that the organization's service-level agreements with the clients are being met
- To make sure that the IT infrastructure outages have only a minimal effect on the organization's net income
- To monitor the entire infrastructure and business processes

93. What is Nagios Log Server?

Nagios Log Server simplifies the process of searching the log data. Nagios Log Server is the best choice to perform tasks such as setting up alerts, notifying when potential threats arise, simply querying the log data, and quickly auditing any system. With Nagios Log Server, we can get all of our log data in one location.

94. Can you tell me why I should use Nagios for HTTP monitoring?

Nagios can provide us with a complete monitoring service for our HTTP servers and protocols. Here are a few benefits of implementing effective HTTP monitoring with Nagios:

- Server, services, and application availability can be increased.
- Network outages and protocol failures can be detected quickly.
- User experience can be monitored.
- Web server performance can be monitored.
- Web transactions can be monitored.
- URLs can be monitored.

95. What is a namespace in Kubernetes?

Namespaces are a way to divide cluster resources between multiple users in Kubernetes. In other words, it is useful when multiple teams or users are using the same cluster which can lead to potential name collision.

96. What is kubectl?

By definition, kubectl is a command-line interface for running commands against Kubernetes clusters. Here, 'ctl' stands for 'control'. This 'kubectl' command-line interface can be used to deploy applications, inspect and manage cluster resources, and view logs.

97. How does Kubernetes help in container orchestration and deployment in a DevOps environment?

Kubernetes helps in container orchestration and deployment in a DevOps environment by providing a platform for managing, scaling, and automating the deployment of containerized applications. It allows developers to deploy and manage applications consistently across different environments and infrastructures, ensuring they are reliable, scalable, and resilient.

Kubernetes automates many tasks related to the deployment and scaling of containers, such as load balancing, scaling, and self-healing, which helps to reduce manual effort and errors and ensure that applications are always available and performing.

98. What are some common challenges in managing Kubernetes clusters in a DevOps environment, and how can they be addressed?

In a DevOps environment, complicated configuration management, monitoring, and troubleshooting, as well as maintaining security and compliance, are common problems in operating Kubernetes clusters.

To address these problems, several strategies can be implemented. Firstly, adopting best practices like infrastructure-as-code and declarative configuration for cluster management can streamline operations. Secondly, automating tasks through CI/CD pipelines can save time and effort. Furthermore, implementing monitoring and logging solutions enables better visibility into cluster performance and security, aiding in problem-solving efforts.

99. Which file is used to define dependency in Maven?

In Maven, we define all dependencies inside pom.xml so that all the dependencies will be downloaded and can be used within the project.

100. What are the benefits of using Maven in a DevOps environment?

Maven provides several benefits for DevOps teams, including centralized management of project dependencies, simplified build configuration using a declarative approach, and automated build and deployment processes through integration with CI/CD pipelines.

Maven also supports modular project structures, allowing teams to develop and test individual components separately, and provides consistent and reproducible builds across development, testing, and production environments. These benefits enable faster and more efficient delivery of software applications in a DevOps environment.

101. Your company plans to move some applications to the cloud. How would you approach the migration plan while keeping security, compliance, and cost optimization in mind?

Let’s imagine our company has a few internal applications that we want to move to the public cloud – like AWS or Azure. How should we approach this?

First – I would make a plan looking at things like:

1. Security

– We need to make sure our data and apps are secure when they move to the cloud. Adding firewall rules, network security groups, role based access etc. Making sure only authorized people can access them.

1. Compliance

– Based on our industry, there may be rules and policies we need to follow even in the cloud – like HIPAA healthcare policies or PCI data standards. We need to check what compliance needs we have and implement those controls in the cloud.

1. Cost

– Clouds provide different instance types, reserved instances etc. So we should optimize and right size our resources to save costs. Only paying for what we use. Also think about auto-scaling resources up and down based on usage and demand.

The migration plan would cover setting up the right cloud accounts and access controls, configuring networks and firewalls, deploying servers and resources as needed while making sure they comply with all our policies. Testing things out first with non-production data. And figuring out how to best monitor, scale and optimize all these applications over time as needed.

102. A new feature released to production starts causing increased CPU load, latency issues and some failures. The team is asking you why. How would you approach debugging this issue? What techniques and tools would you use?

Let's say our app has a new feature that started causing problems – things are running slow, some requests are failing, and our servers are working harder. The developers want to know why.

Some ways I'd start investigating as a DevOps engineer:

1. Look at metrics and monitoring – Many DevOps tools can show metrics for things like CPU usage, memory, network traffic. I'd check graphs around the time issues started happening. If CPU spiked, that's a clue.
2. Review logs – Application and web server logs often record errors or warnings about issues. Search logs from that timeframe for any common errors or repeating patterns.
3. Trace requests – Tools like distributed tracing can follow a specific request through microservices. This can help pinpoint high response times and failures to a specific service.
4. Compare configurations – I'd check if any infrastructure or application changes were made right before issues started – new cloud settings, firewall rules, dependency versions etc. Roll back changes to test.
5. Performance test – Simulate user traffic against different app versions to reproduce issues in a test environment. Helps narrow down root cause.

The goal is finding patterns and reducing variables until the specific line of code, resource limit or configuration causing problems is uncovered. Using monitoring, logs, system data and tests to incrementally eliminate possibilities.

103. Developers complain applications have become very slow after moving to Kubernetes and containers. What strategies would you use for troubleshooting performance issues in a container environment?

Let's imagine our developers just moved their applications to Kubernetes and docker containers. Everything used to run pretty fast, but now it has suddenly become really slow! Developers are complaining to us DevOps engineers about this. How can we figure out why?

First, I would check some basics on the Kubernetes nodes:

- Are CPU/memory resources being overloaded on nodes? Using too many resources could slow things down.
- Is network traffic between nodes and pods unusually high? That could indicate performance issues.

Then I'd look at the applications running:

- Are the container images much larger than before? Bigger images means slower start up times.
- Are there errors or throttling happening at the app level inside containers? App logs and metrics would provide clues.

Next is checking how Kubernetes manages the application:

- Are there issues with how Kubernetes is scheduling pods on nodes? We need to make sure spreading is even.
- Is Kubernetes auto-scaling not working quickly enough during traffic spikes? Causing temporary resource bottlenecks.

There are also tools that can help profile Kubernetes as it manages containerized apps to check for performance bottlenecks.

DevOps Engineer Salary Trends

The current job trends point out that the DevOps market share is growing, with skilled professionals in high global demand.

In India, the average salary is ₹8,45,000, with senior roles reaching ₹45,00,000 for the top 1% in jobs offered by companies like IBM, Accenture, and Oracle, as observed on websites such as LinkedIn and Naukri. Even in the US, average base salaries fluctuate around $104,441 and vary based on experience, reaching $149,168 for 5-7 years and beyond.

Here's a quick overview of current salary trends for DevOps engineers at top companies like IBM, Oracle, Amazon, and others that have been observed on websites like LinkedIn Salary Tool, Naukri, and Glassdoor.

Entry-Level Jobs	**Intermediate-Level Jobs**	**Senior-Level Jobs**
Average Annual Salary	₹4.0 Lakhs – ₹6.5 Lakhs per year	₹8.0 Lakhs – ₹14.0 Lakhs per year
Range	₹3.5 Lakhs – ₹8.0 Lakhs per year	₹6.5 Lakhs – ₹16.0 Lakhs per year
Key Skills	DevOps principles and automation tools (Git, Jenkins, Ansible), scripting languages (Python, Bash), and basic knowledge of cloud platforms and containerization technologies like Docker	Advanced DevOps practices (CI/CD pipelines, infrastructure as code), experience with cloud platforms (AWS, Azure, GCP), and expertise in monitoring and troubleshooting tools

DevOps Engineer Job Trends

The global DevOps market is scheduled for a growth boom in 2024. According to a recent market survey by DevOps.com, the DevOps market is estimated to grow to a **$20 billion industry share by**

2026, growing at a **CAGR of 24.7% from 2019 to 2026.**

According to a Forbes report, DevOps has enabled a quick and reliable software development ecosystem, leading to better quality and customer satisfaction. Gartner anticipates that by 2026, 80% of software engineering organizations will create deployment teams to facilitate a quick and seamless application delivery process.

The growing adaptation of cloud solutions in every stratum of development and deployment, as pointed out in a Forbes report, is driving the demand for DevOps engineers. At the same time, proficiency in cloud platforms like AWS, Azure, and GCP is crucial for these professionals.

Based on the latest number of jobs listed on LinkedIn, there are currently **25,000+ DevOps job openings available in India**. Additionally, Naukri reports indicate that as of **January 2024**, **there are 19,909** vacancies for DevOps jobs. **In the United States**, LinkedIn reports that there are currently **17,000 job** openings for DevOps engineers.

DevOps Engineer Roles and Responsibilities

DevOps engineers at tech giants like IBM and Oracle are the ones holding development and operations together. They overcome obstacles, automate deployments, and solve problems across development. This streamlines software delivery, making apps roll out faster and smoother, keeping those companies at the top of the game.

Here's an overview of the responsibilities expected from DevOps engineers and related roles posted by companies like IBM and Oracle.

DevOps Engineer Responsibilities:

- Collaborating with development and operations teams to build better workflows.
- Design and implement CI/CD pipelines for automated build, testing, and deployment.
- Manage infrastructure provisioning, configuration, and maintenance.
- Set up systems to track app performance and flag potential issues.
- Integrate security practices throughout the software lifecycle.
- Identify and address bottlenecks in the DevOps workflow.
- Secure infrastructure configurations and access controls.
- Develop and implement automated processes for incident response and recovery.
- Ensure compliance with relevant security regulations and standards.

Technical Skills:

- Thorough understanding of DevGit, Jenkins, and Ansible
- CI/CD pipelines, infrastructure as code
- Intermediate knowledge of AWS, Azure, and GCP
- Expertise in security and automation, and contribution to open-source projects

Soft Skills:

- Excellent written and verbal communication
- Work independently and in a team

- Meet deadlines