Top DevOps Interview Questions

These are the top questions you might face in a DevOps job interview:

General DevOps Interview Questions

This category will include questions that are not related to any particular DevOps stage. Questions here are meant to test your understanding about DevOps rather than focusing on a particular tool or a stage.

Q1. What are the fundamental differences between DevOps & Agile?

The differences between the two are listed down in the table below.

Features	DevOps	Agile
Agility	Agility in both Development & Operations	Agility in only Development
Processes/ Practices	Involves processes such as CI, CD, CT, etc.	Involves practices such as Agile Scrum, Agile Kanban, etc.
Key Focus Area	Timeliness & quality have equal priority	Timeliness is the main priority
Release Cycles/ Development Sprints	Smaller release cycles with immediate feedback	Smaller release cycles
Source of Feedback	Feedback is from self (Monitoring tools)	Feedback is from customers
Scope of Work	Agility & need for Automation	Agility only

DevOps vs Agile

Q2. What is the need for DevOps?

According to me, this answer should start by explaining the general market trend. Instead of releasing big sets of features, companies are trying to see if small features can be transported to their customers through a series of release trains. This has many advantages like quick feedback from customers, better quality of software etc. which in turn leads to high customer satisfaction. To achieve this, companies are required to:

1. Increase deployment frequency
2. Lower failure rate of new releases
3. Shortened lead time between fixes
4. Faster mean time to recovery in the event of new release crashing

DevOps fulfills all these requirements and helps in achieving seamless software delivery. You can give examples of companies like Etsy, Google and Amazon which have adopted DevOps to achieve levels of performance that were unthinkable even five years ago. They are doing tens, hundreds or even thousands of code deployments per day while delivering world class stability, reliability and security.

If I have to test your knowledge on DevOps, you should know the difference between Agile and DevOps. The next question is directed towards that.

Q3. How is DevOps different from Agile / SDLC?

I would advise you to go with the below explanation:

Agile is a set of values and principles about how to produce i.e. develop software. Example: if you have some ideas and you want to turn those ideas into working software, you can use the Agile values and principles as a way to do that. But, that software might only be working on a developer's laptop or in a test environment. You want a way to quickly, easily and repeatably move that software into production infrastructure, in a safe and simple way. To do that you need DevOps tools and techniques.

You can summarize by saying Agile software development methodology focuses on the development of software but DevOps on the other hand is responsible for development as well as deployment of the software in the safest and most reliable way possible. Here's a blog that will give you more information on the evolution of DevOps.

Now remember, you have included DevOps tools in your previous answer so be prepared to answer some questions related to that.

Q4. Which are the top DevOps tools? Which tools have you worked on?

The most popular DevOps tools are mentioned below:

- Git : Version Control System tool

- Jenkins : Continuous Integration tool
- Selenium : Continuous Testing tool
- Puppet, Chef, Ansible : Configuration Management and Deployment tools
- Nagios : Continuous Monitoring tool
- Docker : Containerization tool

You can also mention any other tool if you want, but make sure you include the above tools in your answer.

The second part of the answer has two possibilities:

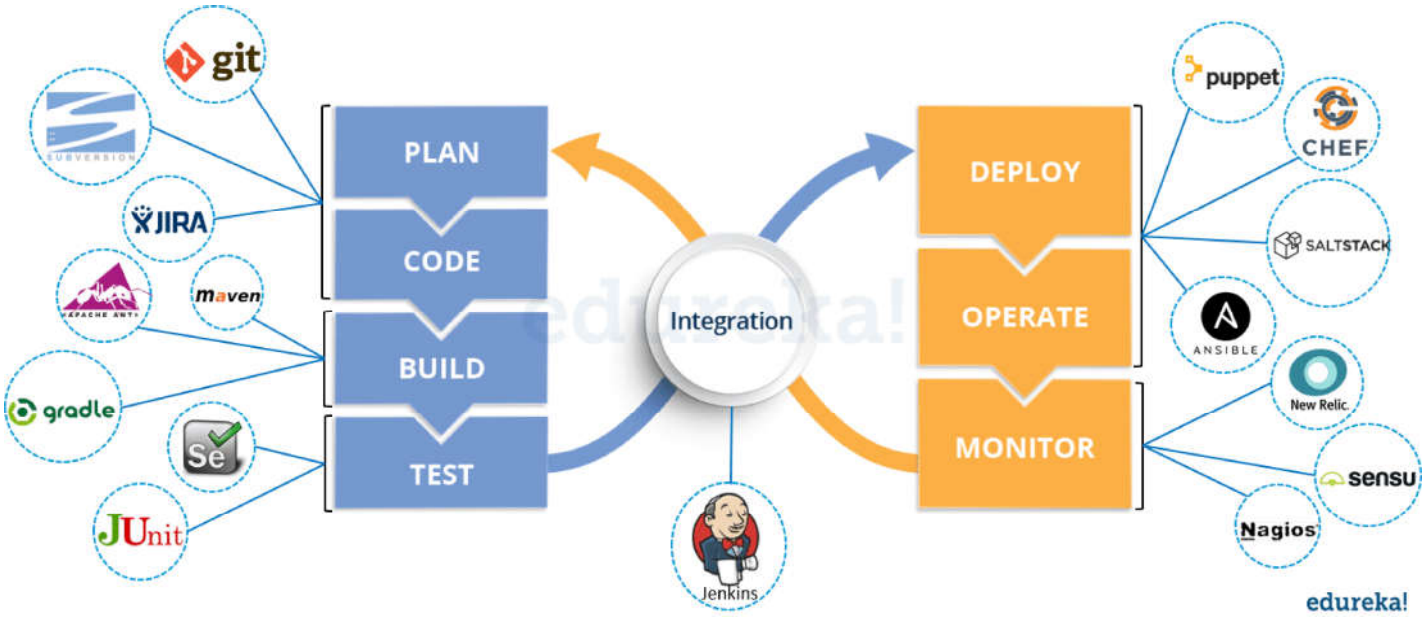
1. If you have experience with all the above tools then you can say that I have worked on all these tools for developing good quality software and deploying those softwares easily, frequently, and reliably.
2. If you have experience only with some of the above tools then mention those tools and say that I have specialization in these tools and have an overview about the rest of the tools.

Our DevOps certification course includes hands-on training on the most popular DevOps tools. Find out when the next batch starts.

Q5. How do all these tools work together?

Given below is a generic logical flow where everything gets automated for seamless delivery. However, this flow may vary from organization to organization as per the requirement.

1. Developers develop the code and this source code is managed by Version Control System tools like Git etc.
2. Developers send this code to the Git repository and any changes made in the code is committed to this Repository.
3. Jenkins pulls this code from the repository using the Git plugin and build it using tools like Ant or Maven.
4. Configuration management tools like puppet deploys & provisions testing environment and then Jenkins releases this code on the test environment on which testing is done using tools like selenium.
5. Once the code is tested, Jenkins send it for deployment on the production server (even production server is provisioned & maintained by tools like puppet).
6. After deployment It is continuously monitored by tools like Nagios.
7. Docker containers provides testing environment to test the build features.



Q6. What are the advantages of DevOps?

For this answer, you can use your past experience and explain how DevOps helped you in your previous job. If you don't have any such experience, then you can mention the below advantages.

Technical benefits:

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

Business benefits:

- Faster delivery of features
- More stable operating environments
- More time available to add value (rather than fix/maintain)

Q7. What is the most important thing DevOps helps us achieve?

According to me, the most important thing that DevOps helps us achieve is to get the changes into production as quickly as possible while minimizing risks in software quality assurance and compliance. This is the primary objective of DevOps. Learn more in this DevOps tutorial blog.

However, you can add many other positive effects of DevOps. For example, clearer communication and better working relationships between teams i.e. both the Ops team and Dev team collaborate together to deliver good quality software which in turn leads to higher customer satisfaction.

Q8. Explain with a use case where DevOps can be used in industry/ real-life.

There are many industries that are using DevOps so you can mention any of those use cases, you can also refer the below example:
Etsy is a peer-to-peer e-commerce website focused on handmade or vintage items and supplies, as well as unique factory-manufactured items. Etsy struggled with slow, painful site updates that frequently caused the site to go down. It affected sales for millions of Etsy's users who sold goods through online market place and risked driving them to the competitor.
With the help of a new technical management team, Etsy transitioned from its waterfall model, which produced four-hour full-site deployments twice weekly, to a more agile approach. Today, it has a fully automated deployment pipeline, and its continuous delivery practices have reportedly resulted in more than 50 deployments a day with fewer disruptions.

Q9. Explain your understanding and expertise on both the software development side and the technical operations side of an organization you have worked with in the past.

For this answer, share your past experience and try to explain how flexible you were in your previous job. You can refer the below example: DevOps engineers almost always work in a 24/7 business-critical online environment. I was adaptable to on-call duties and was available to take up real-time, live-system responsibility. I successfully automated processes to support continuous software deployments. I have experience with public/private clouds, tools like Chef or Puppet, scripting and automation with tools like Python and PHP, and a background in Agile.

Q10. What are the anti-patterns of DevOps?

A pattern is common usage usually followed. If a pattern commonly adopted by others does not work for your organization and you continue to blindly follow it, you are essentially adopting an anti-pattern. There are myths about DevOps. Some of them include:

- DevOps is a process
- Agile equals DevOps?
- We need a separate DevOps group
- Devops will solve all our problems
- DevOps means Developers Managing Production
- DevOps is Development-driven release management
    1. DevOps is not development driven.
    2. DevOps is not IT Operations driven.
- We can't do DevOps – We're Unique
- We can't do DevOps – We've got the wrong people

Version Control System (VCS) Interview Questions

Now let's look at some interview questions on VCS. If you would like to get hands-on training on a VCS like Git, it's included in our DevOps Certification course.

Q1. What is Version control?

This is probably the easiest question you will face in the interview. My suggestion is to first give a definition of Version control. It is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control systems consist of a central shared repository where teammates can commit changes to a file or set of file. Then you can mention the uses of version control.

Version control allows you to:

- Revert files back to a previous state.
- Revert the entire project back to a previous state.
- Compare changes over time.
- See who last modified something that might be causing a problem.
- Who introduced an issue and when.

Q2. What are the benefits of using version control?

I will suggest you to include the following advantages of version control:

1. With Version Control System (VCS), all the team members are allowed to work freely on any file at any time. VCS will later allow you to merge all the changes into a common version.
2. All the past versions and variants are neatly packed up inside the VCS. When you need it, you can request any version at any time and you'll have a snapshot of the complete project right at hand.
3. Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Additionally, you can see what exactly was changed in the file's content. This allows you to know who has made what change in the project.
4. A distributed VCS like Git allows all the team members to have complete history of the project so if there is a breakdown in the central server you can use any of your teammate's local Git repository.

Q3. Describe branching strategies you have used.

This question is asked to test your branching experience so tell them about how you have used branching in your previous job and what purpose does it serves, you can refer the below points:

- Feature branching
  A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.
- Task branching
  In this model each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.
- Release branching
  Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into develop branch, which may have progressed since the release was initiated.

In the end tell them that branching strategies varies from one organization to another, so I know basic branching operations like delete, merge, checking out a branch etc.

Q4. Which VCS tool you are comfortable with?

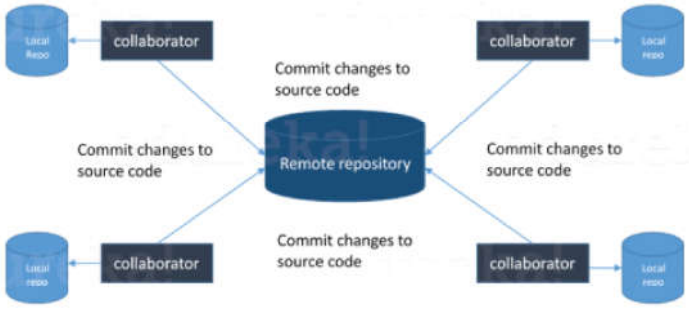
You can just mention the VCS tool that you have worked on like this: "I have worked on Git and one major advantage it has over other VCS tools like SVN is that it is a distributed version control system."

Distributed VCS tools do not necessarily rely on a central server to store all the versions of a project's files. Instead, every developer "clones" a copy of a repository and has the full history of the project on their own hard drive.

Q5. What is Git?

I will suggest that you attempt this question by first explaining about the architecture of git as shown in the below diagram. You can refer to the explanation given below:

- Git is a Distributed Version Control system (DVCS). It can track changes to a file and allows you to revert back to any particular change.
- Its distributed architecture provides many advantages over other Version Control Systems (VCS) like SVN one major advantage is that it does not rely on a central server to store all the versions of a project's files. Instead, every developer "clones" a copy of a repository I have shown in the diagram below with "Local repository" and has the full history of the project on his hard drive so that when there is a server outage, all you need for recovery is one of your teammate's local Git repository.
- There is a central cloud repository as well where developers can commit changes and share it with other teammates as you can see in the diagram where all collaborators are commiting changes "Remote repository".



Q6. Explain some basic Git commands?

Below are some basic Git commands:

Command	Function
git config --global user.name "name" git config --global user.email "E-mail"	Configure the author name and email address to be used with your commits
Git init	Create a new local local respository
Git clone /path/to/repository	Create a working copy of a local repository
git clone username@host:/path/to/repository	For a remote server, use
git add <Filename.> git add *	Add one or more file to staging
git commit -m "Commit message"	Commit changes to head
git commit -a	Commit any files you've added with git add, and also commit any files you've changed since then
git push origin master	Send changes to the master branch of your remote repository
git status	List the files you've changed and those you still need to add or commit
git remote add origin <server>	If you haven't connected your local repository to a remote server, add the server to be able to push to it

Q7. In Git how do you revert a commit that has already been pushed and made public?

There can be two answers to this question so make sure that you include both because any of the below options can be used depending on the situation:

- Remove or fix the bad file in a new commit and push it to the remote repository. This is the most natural way to fix an error. Once you have made necessary changes to the file, commit it to the remote repository for that I will use
  git commit -m "commit message"
- Create a new commit that undoes all changes that were made in the bad commit.to do this I will use a command
  git revert <name of bad commit>

Q8. How do you squash last N commits into a single commit?

There are two options to squash last N commits into a single commit. Include both of the below mentioned options in your answer:

- If you want to write the new commit message from scratch use the following command
  git reset –soft HEAD~N &&
  git commit
- If you want to start editing the new commit message with a concatenation of the existing commit messages then you need to extract those messages and pass them to Git commit for that I will use
  git reset –soft HEAD~N &&
  git commit –edit -m"$(git log –format=%B –reverse .HEAD@{N})"

Q9. What is Git bisect? How can you use it to determine the source of a (regression) bug?

I will suggest you to first give a small definition of Git bisect, Git bisect is used to find the commit that introduced a bug by using binary search. Command for Git bisect is

git bisect <subcommand> <options>

Now since you have mentioned the command above, explain what this command will do, This command uses a binary search algorithm to find which commit in your project's history introduced a bug. You use it by first telling it a "bad" commit that is known to contain the bug, and a "good" commit that is known to be before the bug was introduced. Then Git bisect picks a commit between those two endpoints and asks you whether the selected commit is "good" or "bad". It continues narrowing down the range until it finds the exact commit that introduced the change.

Q10. What is Git rebase and how can it be used to resolve conflicts in a feature branch before merge?

According to me, you should start by saying git rebase is a command which will merge another branch into the branch where you are currently working, and move all of the local commits that are ahead of the rebased branch to the top of the history on that branch.

Now once you have defined Git rebase time for an example to show how it can be used to resolve conflicts in a feature branch before merge, if a feature branch was created from master, and since then the master branch has received new commits, Git rebase can be used to move the feature branch to the tip of master.

The command effectively will replay the changes made in the feature branch at the tip of master, allowing conflicts to be resolved in the process. When done with care, this will allow the feature branch to be merged into master with relative ease and sometimes as a simple fast-forward operation.

Q11. How do you configure a Git repository to run code sanity checking tools right before making commits, and preventing them if the test fails?

I will suggest you to first give a small introduction to sanity checking, A sanity or smoke test determines whether it is possible and reasonable to continue testing.

Now explain how to achieve this, this can be done with a simple script related to the pre-commit hook of the repository. The pre-commit hook is triggered right before a commit is made, even before you are required to enter a commit message. In this script one can run other tools, such as linters and perform sanity checks on the changes being committed into the repository.

Finally give an example, you can refer the below script:

```
#!/bin/sh
files=$(git diff –cached –name-only –diff-filter=ACM | grep '.go$')
if [ -z files ]; then
exit 0
fi
unfmtd=$(gofmt -l $files)
if [ -z unfmtd ]; then
exit 0
fi
echo "Some .go files are not fmt'd"
exit 1
```

This script checks to see if any .go file that is about to be committed needs to be passed through the standard Go source code formatting tool gofmt. By exiting with a non-zero status, the script effectively prevents the commit from being applied to the repository.

Q12. How do you find a list of files that has changed in a particular commit?

For this answer instead of just telling the command, explain what exactly this command will do so you can say that, To get a list files that has changed in a particular commit use command

git diff-tree -r {hash}

Given the commit hash, this will list all the files that were changed or added in that commit. The -r flag makes the command list individual files, rather than collapsing them into root directory names only.

You can also include the below mention point although it is totally optional but will help in impressing the interviewer.

The output will also include some extra information, which can be easily suppressed by including two flags:

git diff-tree –no-commit-id –name-only -r {hash}

Here –no-commit-id will suppress the commit hashes from appearing in the output, and –name-only will only print the file names, instead of their paths.

Q13. How do you setup a script to run every time a repository receives new commits through push?

There are three ways to configure a script to run every time a repository receives new commits through push, one needs to define either a pre-receive, update, or a post-receive hook depending on when exactly the script needs to be triggered.

- Pre-receive hook in the destination repository is invoked when commits are pushed to it. Any script bound to this hook will be executed before any references are updated. This is a useful hook to run scripts that help enforce development policies.
- Update hook works in a similar manner to pre-receive hook, and is also triggered before any updates are actually made. However, the update hook is called once for every commit that has been pushed to the destination repository.
- Finally, post-receive hook in the repository is invoked after the updates have been accepted into the destination repository. This is an ideal place to configure simple deployment scripts, invoke some continuous integration systems, dispatch notification emails to repository maintainers, etc.

Hooks are local to every Git repository and are not versioned. Scripts can either be created within the hooks directory inside the ".git" directory, or they can be created elsewhere and links to those scripts can be placed within the directory.

Q14. How will you know in Git if a branch has already been merged into master?

I will suggest you to include both the below mentioned commands:

git branch –merged lists the branches that have been merged into the current branch.

git branch –no-merged lists the branches that have not been merged.

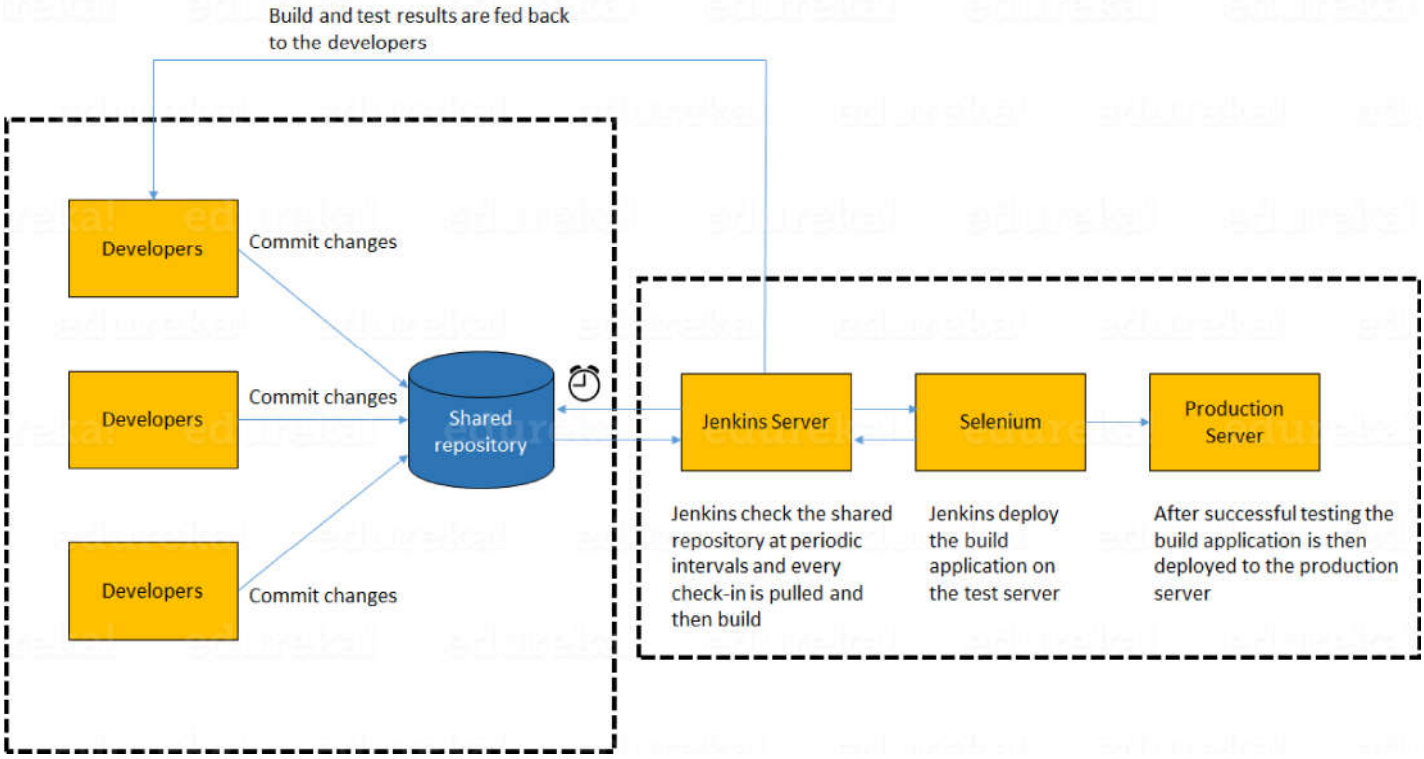
Continuous Integration questions

Now, let's look at Continuous Integration interview questions:

Q1. What is meant by Continuous Integration?

I will advise you to begin this answer by giving a small definition of Continuous Integration (CI). It is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

I suggest that you explain how you have implemented it in your previous job. You can refer the below given example:



In the diagram shown above:

1. Developers check out code into their private workspaces.
2. When they are done with it they commit the changes to the shared repository (Version Control Repository).
3. The CI server monitors the repository and checks out changes when they occur.
4. The CI server then pulls these changes and builds the system and also runs unit and integration tests.
5. The CI server will now inform the team of the successful build.
6. If the build or tests fails, the CI server will alert the team.
7. The team will try to fix the issue at the earliest opportunity.
8. This process keeps on repeating.

Q2. Why do you need a Continuous Integration of Dev & Testing?

For this answer, you should focus on the need of Continuous Integration. My suggestion would be to mention the below explanation in your answer:

Continuous Integration of Dev and Testing improves the quality of software, and reduces the time taken to deliver it, by replacing the traditional practice of testing after completing all development. It allows Dev team to easily detect and locate problems early because developers need to integrate code into a shared repository several times a day (more frequently). Each check-in is then automatically tested.

Q3. What are the success factors for Continuous Integration?

Here you have to mention the requirements for Continuous Integration. You could include the following points in your answer:

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Q4. Explain how you can move or copy Jenkins from one server to another?

I will approach this task by copying the jobs directory from the old server to the new one. There are multiple ways to do that; I have mentioned them below:

You can:

- Move a job from one installation of Jenkins to another by simply copying the corresponding job directory.
- Make a copy of an existing job by making a clone of a job directory by a different name.
- Rename an existing job by renaming a directory. Note that if you change a job name you will need to change any other job that tries to call the renamed job.

Q5. Explain how can create a backup and copy files in Jenkins?

Answer to this question is really direct. To create a backup, all you need to do is to periodically back up your JENKINS_HOME directory. This contains all of your build jobs configurations, your slave node configurations, and your build history. To create a back-up of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory.

Q6. Explain how you can setup Jenkins job?

My approach to this answer will be to first mention how to create Jenkins job. Go to Jenkins top page, select “New Job”, then choose “Build a free-style software project”.

Then you can tell the elements of this freestyle job:

- Optional SCM, such as CVS or Subversion where your source code resides.
- Optional triggers to control when Jenkins will perform builds.
- Some sort of build script that performs the build (ant, maven, shell script, batch file, etc.) where the real work happens.
- Optional steps to collect information out of the build, such as archiving the artifacts and/or recording javadoc and test results.
- Optional steps to notify other people/systems with the build result, such as sending e-mails, IMs, updating issue tracker, etc..

Q7. Mention some of the useful plugins in Jenkins.

Below, I have mentioned some important Plugins:

- Maven 2 project
- Amazon EC2
- HTML publisher
- Copy artifact
- Join
- Green Balls

These Plugins, I feel are the most useful plugins. If you want to include any other Plugin that is not mentioned above, you can add them as well. But, make sure you first mention the above stated plugins and then add your own.

Q8. How will you secure Jenkins?

The way I secure Jenkins is mentioned below. If you have any other way of doing it, please mention it in the comments section below:

- Ensure global security is on.
- Ensure that Jenkins is integrated with my company’s user directory with appropriate plugin.
- Ensure that matrix/Project matrix is enabled to fine tune access.
- Automate the process of setting rights/privileges in Jenkins with custom version controlled script.
- Limit physical access to Jenkins data/folders.
- Periodically run security audits on same.

Jenkins is one of the many popular tools that are used extensively in DevOps. Edureka's DevOps Certification course will provide you hands-on training with Jenkins and high quality guidance from industry experts. Give it a look:

Here's what Shyam Verma, one of our DevOps learners had to say about our DevOps Training course:

Continuous Testing Interview Questions:

Now let's move on to the Continuous Testing questions.

Q1. What is Continuous Testing?

I will advise you to follow the below mentioned explanation:
Continuous Testing is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with in the latest build. In this way, each build is tested continuously, allowing Development teams to get fast feedback so that they can prevent those problems from progressing to the next stage of Software delivery life-cycle. This dramatically speeds up a developer's workflow as there's no need to manually rebuild the project and re-run all tests after making changes.

Q2. What is Automation Testing?

Automation testing or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves use of separate testing tools which lets you create test scripts which can be executed repeatedly and doesn't require any manual intervention.

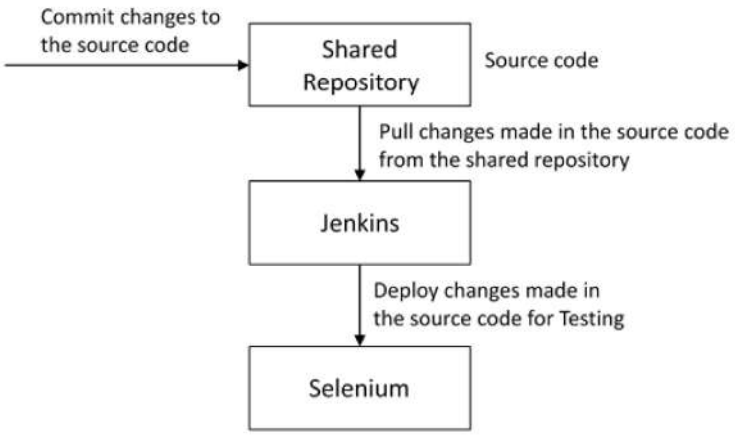
Q3. What are the benefits of Automation Testing?

I have listed down some advantages of automation testing. Include these in your answer and you can add your own experience of how Continuous Testing helped your previous company:

- Supports execution of repeated test cases
- Aids in testing a large test matrix
- Enables parallel execution
- Encourages unattended execution
- Improves accuracy thereby reducing human generated errors
- Saves time and money

Q4. How to automate Testing in DevOps lifecycle?

I have mentioned a generic flow below which you can refer to:
In DevOps, developers are required to commit all the changes made in the source code to a shared repository. Continuous Integration tools like Jenkins will pull the code from this shared repository every time a change is made in the code and deploy it for Continuous Testing that is done by tools like Selenium as shown in the below diagram.
In this way, any change in the code is continuously tested unlike the traditional approach.



Q5. Why is Continuous Testing important for DevOps?

You can answer this question by saying, "Continuous Testing allows any change made in the code to be tested immediately. This avoids the problems created by having "big-bang" testing left to the end of the cycle such as release delays and quality issues. In this way, Continuous Testing facilitates more frequent and good quality releases."

Want all your DevOps questions answered while you learn? Want 24×7 lifetime support and permanent access to all course material? Make sure you check out our DevOps Certification program.

Q6. What are the key elements of Continuous Testing tools?

Key elements of Continuous Testing are:

- **Risk Assessment:** It Covers risk mitigation tasks, technical debt, quality assessment and test coverage optimization to ensure the build is ready to progress toward next stage.
- **Policy Analysis:** It ensures all processes align with the organization's evolving business and compliance demands are met.
- **Requirements Traceability:** It ensures true requirements are met and rework is not required. An object assessment is used to identify which requirements are at risk, working as expected or require further validation.
- **Advanced Analysis:** It uses automation in areas such as static code analysis, change impact analysis and scope assessment/prioritization to prevent defects in the first place and accomplishing more within each iteration.
- **Test Optimization:** It ensures tests yield accurate outcomes and provide actionable findings. Aspects include Test Data Management, Test Optimization Management and Test Maintenance
- **Service Virtualization:** It ensures access to real-world testing environments. Service visualization enables access to the virtual form of the required testing stages, cutting the waste time to test environment setup and availability.

Q7. Which Testing tool are you comfortable with and what are the benefits of that tool?

Here mention the testing tool that you have worked with and accordingly frame your answer. I have mentioned an example below:

I have worked on Selenium to ensure high quality and more frequent releases.

Some advantages of Selenium are:

- It is free and open source
- It has a large user base and helping communities
- It has cross Browser compatibility (Firefox, chrome, Internet Explorer, Safari etc.)
- It has great platform compatibility (Windows, Mac OS, Linux etc.)
- It supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
- It has fresh and regular repository developments
- It supports distributed testing

Q8. What are the Testing types supported by Selenium?

Selenium supports two types of testing:

Regression Testing: It is the act of retesting a product around an area where a bug was fixed.

Functional Testing: It refers to the testing of software features (functional points) individually.

Q9. What is Selenium IDE?

My suggestion is to start this answer by defining Selenium IDE. It is an integrated development environment for Selenium scripts. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests. Selenium IDE includes the entire Selenium Core, allowing you to easily and quickly record and play back tests in the actual environment that they will run in.

Now include some advantages in your answer. With autocomplete support and the ability to move commands around quickly, Selenium IDE is the ideal environment for creating Selenium tests no matter what style of tests you prefer.

Q10. What is the difference between Assert and Verify commands in Selenium?

I have mentioned differences between Assert and Verify commands below:

- Assert command checks whether the given condition is true or false. Let's say we assert whether the given element is present on the web page or not. If the condition is true, then the program control will execute the next test step. But, if the condition is false, the execution would stop and no further test would be executed.
- Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn't halts i.e. any failure during verification would not stop the execution and all the test steps would be executed.

Q11. How to launch Browser using WebDriver?

The following syntax can be used to launch Browser:

WebDriver driver = new FirefoxDriver();

WebDriver driver = new ChromeDriver();

WebDriver driver = new InternetExplorerDriver();

Q12. When should I use Selenium Grid?

For this answer, my suggestion would be to give a small definition of Selenium Grid. It can be used to execute same or different test scripts on multiple platforms and browsers concurrently to achieve distributed test execution. This allows testing under different environments and saving execution time remarkably.

Learn Automation testing and other DevOps concepts in live instructor-led online classes in our DevOps Certification course.

Configuration Management Interview Questions

Now let's check how much you know about Configuration Management.

Q1. What are the goals of Configuration management processes?

The purpose of Configuration Management (CM) is to ensure the integrity of a product or system throughout its life-cycle by making the development or deployment process controllable and repeatable, therefore creating a higher quality product or system. The CM process allows orderly management of system information and system changes for purposes such as to:

- Revise capability,
- Improve performance,
- Reliability or maintainability,
- Extend life,
- Reduce cost,
- Reduce risk and
- Liability, or correct defects.

Q2. What is the difference between Asset management and Configuration Management?

Given below are few differences between Asset Management and Configuration Management:

Asset Management	Configuration Management
Concerned with finances	Concerned with operations
Scope is everything you own	Scope is everything you deploy
Interfaces to purchasing and leasing	Interfaces to ITIL processes
Maintains data for taxes	Maintains data for troubleshooting
Lifecycle from Purchase to disposal	Lifecycle from deploy to retirement
Only incidental relationships	All operational relationships

Q3. What is the difference between an Asset and a Configuration Item?

According to me, you should first explain Asset. It has a financial value along with a depreciation rate attached to it. IT assets are just a sub-set of it. Anything and everything that has a cost and the organization uses it for its asset value calculation and related benefits in tax calculation falls under Asset Management, and such item is called an asset.

Configuration Item on the other hand may or may not have financial values assigned to it. It will not have any depreciation linked to it. Thus, its life would not be dependent on its financial value but will depend on the time till that item becomes obsolete for the organization.

Now you can give an example that can showcase the similarity and differences between both:

1) Similarity:

Server – It is both an asset as well as a CI.

2) Difference:

Building – It is an asset but not a CI.

Document – It is a CI but not an asset

Q4. What do you understand by "Infrastructure as code"? How does it fit into the DevOps methodology? What purpose does it achieve?

Infrastructure as Code (IAC) is a type of IT infrastructure that operations teams can use to automatically manage and provision through code, rather than using a manual process.

Companies for faster deployments treat infrastructure like software: as code that can be managed with the DevOps tools and processes. These tools let you make infrastructure changes more easily, rapidly, safely and reliably.

Q5. Which among Puppet, Chef, SaltStack and Ansible is the best Configuration Management (CM) tool? Why?

This depends on the organization's need so mention few points on all those tools:

Puppet is the oldest and most mature CM tool. Puppet is a Ruby-based Configuration Management tool, but while it has some free features, much of what makes Puppet great is only available in the paid version. Organizations that don't need a lot of extras will find Puppet useful, but those needing more customization will probably need to upgrade to the paid version.

Chef is written in Ruby, so it can be customized by those who know the language. It also includes free features, plus it can be upgraded from open source to enterprise-level if necessary. On top of that, it's a very flexible product.

Ansible is a very secure option since it uses Secure Shell. It's a simple tool to use, but it does offer a number of other services in addition to configuration management. It's very easy to learn, so it's perfect for those who don't have a dedicated IT staff but still need a configuration management tool.

SaltStack is python based open source CM tool made for larger businesses, but its learning curve is fairly low.

Q6. What is Puppet?

I will advise you to first give a small definition of Puppet. It is a Configuration Management tool which is used to automate administration tasks.

Now you should describe its architecture and how Puppet manages its Agents. Puppet has a Master-Slave architecture in which the Slave has to first send a Certificate signing request to Master and Master has to sign that Certificate in order to establish a secure connection between Puppet Master and Puppet Slave as shown on the diagram below. Puppet Slave sends request to Puppet Master and Puppet Master then pushes configuration on Slave.

Refer the diagram below that explains the above description.

Q7. Before a client can authenticate with the Puppet Master, its certs need to be signed and accepted. How will you automate this task?

The easiest way is to enable auto-signing in puppet.conf.

Do mention that this is a security risk. If you still want to do this:

- Firewall your puppet master – restrict port tcp/8140 to only networks that you trust.
- Create puppet masters for each 'trust zone', and only include the trusted nodes in that Puppet masters manifest.
- Never use a full wildcard such as *.

Q8. Describe the most significant gain you made from automating a process through Puppet.

For this answer, I will suggest you to explain you past experience with Puppet. you can refer the below example:
I automated the configuration and deployment of Linux and Windows machines using Puppet. In addition to shortening the processing time from one week to 10 minutes, I used the roles and profiles pattern and documented the purpose of each module in README to ensure that others could update the module using Git. The modules I wrote are still being used, but they've been improved by my teammates and members of the community

Q9. Which open source or community tools do you use to make Puppet more powerful?

Over here, you need to mention the tools and how you have used those tools to make Puppet more powerful. Below is one example for your reference:
Changes and requests are ticketed through Jira and we manage requests through an internal process. Then, we use Git and Puppet's Code Manager app to manage Puppet code in accordance with best practices. Additionally, we run all of our Puppet changes through our continuous integration pipeline in Jenkins using the beaker testing framework.

Q10. What are Puppet Manifests?

It is a very important question so make sure you go in a correct flow. According to me, you should first define Manifests. Every node (or Puppet Agent) has got its configuration details in Puppet Master, written in the native Puppet language. These details are written in the language which Puppet can understand and are termed as Manifests. They are composed of Puppet code and their filenames use the .pp extension.
Now give an exampl. You can write a manifest in Puppet Master that creates a file and installs apache on all Puppet Agents (Slaves) connected to the Puppet Master.

Q11. What is Puppet Module and How it is different from Puppet Manifest?

For this answer, you can go with the below mentioned explanation:
A Puppet Module is a collection of Manifests and data (such as facts, files, and templates), and they have a specific directory structure. Modules are useful for organizing your Puppet code, because they allow you to split your code into multiple Manifests. It is considered best practice to use Modules to organize almost all of your Puppet Manifests.
Puppet programs are called Manifests which are composed of Puppet code and their file names use the .pp extension.

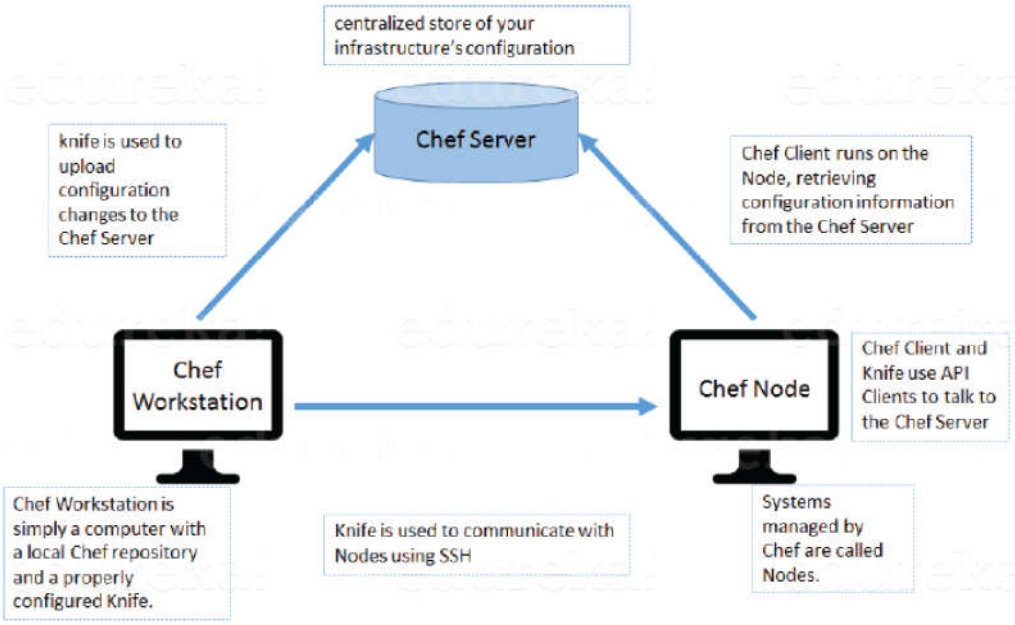
Q12. What is Facter in Puppet?

You are expected to answer what exactly Facter does in Puppet so according to me, you should say, "Facter gathers basic information (facts) about Puppet Agent such as hardware details, network settings, OS type and version, IP addresses, MAC addresses, SSH keys, and more. These facts are then made available in Puppet Master's Manifests as variables."

Q13. What is Chef?

Begin this answer by defining Chef. It is a powerful automation platform that transforms infrastructure into code. Chef is a tool for which you write scripts that are used to automate processes. What processes? Pretty much anything related to IT.
Now you can explain the architecture of Chef, it consists of:

- **Chef Server:** The Chef Server is the central store of your infrastructure's configuration data. The Chef Server stores the data necessary to configure your nodes and provides search, a powerful tool that allows you to dynamically drive node configuration based on data.
- **Chef Node:** A Node is any host that is configured using Chef-client. Chef-client runs on your nodes, contacting the Chef Server for the information necessary to configure the node. Since a Node is a machine that runs the Chef-client software, nodes are sometimes referred to as "clients".
- **Chef Workstation:** A Chef Workstation is the host you use to modify your cookbooks and other configuration data.



Q14. What is a resource in Chef?

My suggestion is to first define Resource. A Resource represents a piece of infrastructure and its desired state, such as a package that should be installed, a service that should be running, or a file that should be generated.

You should explain about the functions of Resource for that include the following points:

- Describes the desired state for a configuration item.
- Declares the steps needed to bring that item to the desired state.
- Specifies a resource type such as package, template, or service.
- Lists additional details (also known as resource properties), as necessary.
- Are grouped into recipes, which describe working configurations.

Q15. What do you mean by recipe in Chef?

For this answer, I will suggest you to use the above mentioned flow: first define Recipe. A Recipe is a collection of Resources that describes a particular configuration or policy. A Recipe describes everything that is required to configure part of a system.

After the definition, explain the functions of Recipes by including the following points:

- Install and configure software components.
- Manage files.
- Deploy applications.
- Execute other recipes.

Q16. How does a Cookbook differ from a Recipe in Chef?

The answer to this is pretty direct. You can simply say, "a Recipe is a collection of Resources, and primarily configures a software package or some piece of infrastructure. A Cookbook groups together Recipes and other information in a way that is more manageable than having just Recipes alone."

Q17. What happens when you don't specify a Resource's action in Chef?

My suggestion is to first give a direct answer: when you don't specify a resource's action, Chef applies the default action.

Now explain this with an example, the below resource:

file 'C:UsersAdministratorchef-reposettings.ini' do
content 'greeting=hello world'
end

is same as the below resource:

file 'C:UsersAdministratorchef-reposettings.ini' do
action :create
content 'greeting=hello world'
end

because: create is the file Resource's default action.

Q18. What is Ansible module?

Modules are considered to be the units of work in Ansible. Each module is mostly standalone and can be written in a standard scripting language such as Python, Perl, Ruby, bash, etc.. One of the guiding properties of modules is idempotency, which means that even if an operation is repeated multiple times e.g. upon recovery from an outage, it will always place the system into the same state.

Q19. What are playbooks in Ansible?

Playbooks are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. Playbooks are designed to be human-readable and are developed in a basic text language.

At a basic level, playbooks can be used to manage configurations of and deployments to remote machines.

Q20. How do I see a list of all of the ansible_ variables?

Ansible by default gathers "facts" about the machines under management, and these facts can be accessed in Playbooks and in templates. To see a list of all of the facts that are available about a machine, you can run the "setup" module as an ad-hoc action:

Ansible -m setup hostname

This will print out a dictionary of all of the facts that are available for that particular host.

Q21. How can I set deployment order for applications?

WebLogic Server 8.1 allows you to select the load order for applications. See the Application MBean Load Order attribute in Application. WebLogic Server deploys server-level resources (first JDBC and then JMS) before deploying applications. Applications are deployed in this order: connectors, then EJBs, then Web Applications. If the application is an EAR, the individual components are loaded in the order in which they are declared in the application.xml deployment descriptor.

Q22. Can I refresh static components of a deployed application without having to redeploy the entire application?

Yes, you can use weblogic.Deployer to specify a component and target a server, using the following syntax:
java weblogic.Deployer -adminurl http://admin:7001 -name appname -targets server1,server2 -deploy jsps/*.jsp

Q23. How do I turn the auto-deployment feature off?

The auto-deployment feature checks the applications folder every three seconds to determine whether there are any new applications or any changes to existing applications and then dynamically deploys these changes.

The auto-deployment feature is enabled for servers that run in development mode. To disable auto-deployment feature, use one of the following methods to place servers in production mode:

- In the Administration Console, click the name of the domain in the left pane, then select the Production Mode checkbox in the right pane.
- At the command line, include the following argument when starting the domain's Administration Server:
  -Dweblogic.ProductionModeEnabled=true
- Production mode is set for all WebLogic Server instances in a given domain.

Q24. When should I use the external_stage option?

Set -external_stage using weblogic.Deployer if you want to stage the application yourself, and prefer to copy it to its target by your own means.

Ansible and Puppet are two of the most popular configuration management tools among DevOps engineers. Learn them and more in our DevOps Masters Program designed by industry experts to certify you as a DevOps Engineer!

Continuous Monitoring Interview Questions

Let's test your knowledge on Continuous Monitoring.

Q1. Why is Continuous monitoring necessary?

I will suggest you to go with the below mentioned flow:

Continuous Monitoring allows timely identification of problems or weaknesses and quick corrective action that helps reduce expenses of an organization. Continuous monitoring provides solution that addresses three operational disciplines known as:

- continuous audit
- continuous controls monitoring
- continuous transaction inspection

Q2. What is Nagios?

You can answer this question by first mentioning that Nagios is one of the monitoring tools. It is used for Continuous monitoring of systems, applications, services, and business processes etc in a DevOps culture. In the event of a failure, Nagios can alert technical staff of the problem, allowing them to begin remediation processes before outages affect business processes, end-users, or customers. With Nagios, you don't have to explain why an unseen infrastructure outage affect your organization's bottom line.
Now once you have defined what is Nagios, you can mention the various things that you can achieve using Nagios.
By using Nagios you can:

- Plan for infrastructure upgrades before outdated systems cause failures.
- Respond to issues at the first sign of a problem.
- Automatically fix problems when they are detected.
- Coordinate technical team responses.
- Ensure your organization's SLAs are being met.
- Ensure IT infrastructure outages have a minimal effect on your organization's bottom line.
- Monitor your entire infrastructure and business processes.

This completes the answer to this question. Further details like advantages etc. can be added as per the direction where the discussion is headed.

Q3. How does Nagios works?

I will advise you to follow the below explanation for this answer:
Nagios runs on a server, usually as a daemon or service. Nagios periodically runs plugins residing on the same server, they contact hosts or servers on your network or on the internet. One can view the status information using the web interface. You can also receive email or SMS notifications if something happens.
The Nagios daemon behaves like a scheduler that runs certain scripts at certain moments. It stores the results of those scripts and will run other scripts if these results change.

Now expect a few questions on Nagios components like Plugins, NRPE etc..

Q4. What are Plugins in Nagios?

Begin this answer by defining Plugins. They are scripts (Perl scripts, Shell scripts, etc.) that can run from a command line to check the status of a host or service. Nagios uses the results from Plugins to determine the current status of hosts and services on your network.
Once you have defined Plugins, explain why we need Plugins. Nagios will execute a Plugin whenever there is a need to check the status of a host or service. Plugin will perform the check and then simply returns the result to Nagios. Nagios will process the results that it receives from the Plugin and take the necessary actions.

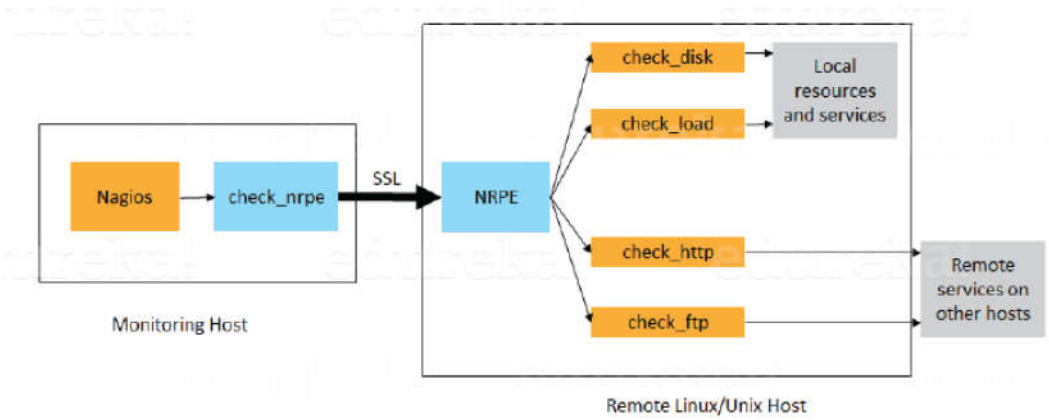
Q5. What is NRPE (Nagios Remote Plugin Executor) in Nagios?

For this answer, give a brief definition of Plugins. The NRPE addon is designed to allow you to execute Nagios plugins on remote Linux/Unix machines. The main reason for doing this is to allow Nagios to monitor "local" resources (like CPU load, memory usage, etc.) on remote machines. Since these public resources are not usually exposed to external machines, an agent like NRPE must be installed on the remote Linux/Unix machines.

I will advise you to explain the NRPE architecture on the basis of diagram shown below. The NRPE addon consists of two pieces:

- The check_nrpe plugin, which resides on the local monitoring machine.
- The NRPE daemon, which runs on the remote Linux/Unix machine.

There is a SSL (Secure Socket Layer) connection between monitoring host and remote host as shown in the diagram below.



Q6. What do you mean by passive check in Nagios?

According to me, the answer should start by explaining Passive checks. They are initiated and performed by external applications/processes and the Passive check results are submitted to Nagios for processing.
Then explain the need for passive checks. They are useful for monitoring services that are Asynchronous in nature and cannot be monitored effectively by polling their status on a regularly scheduled basis. They can also be used for monitoring services that are Located behind a firewall and cannot be checked actively from the monitoring host.

Q7. When Does Nagios Check for external commands?

Make sure that you stick to the question during your explanation so I will advise you to follow the below mentioned flow. Nagios check for external commands under the following conditions:

- At regular intervals specified by the command_check_interval option in the main configuration file or,
- Immediately after event handlers are executed. This is in addition to the regular cycle of external command checks and is done to provide immediate action if an event handler submits commands to Nagios.

Q8. What is the difference between Active and Passive check in Nagios?

For this answer, first point out the basic difference Active and Passive checks. The major difference between Active and Passive checks is that Active checks are initiated and performed by Nagios, while passive checks are performed by external applications.
If your interviewer is looking unconvinced with the above explanation then you can also mention some key features of both Active and Passive checks:
Passive checks are useful for monitoring services that are:

- Asynchronous in nature and cannot be monitored effectively by polling their status on a regularly scheduled basis.
- Located behind a firewall and cannot be checked actively from the monitoring host.

The main features of Actives checks are as follows:

- Active checks are initiated by the Nagios process.
- Active checks are run on a regularly scheduled basis.

Q9. How does Nagios help with Distributed Monitoring?

The interviewer will be expecting an answer related to the distributed architecture of Nagios. So, I suggest that you answer it in the below mentioned format:
With Nagios you can monitor your whole enterprise by using a distributed monitoring scheme in which local slave instances of Nagios perform monitoring tasks and report the results back to a single master. You manage all configuration, notification, and reporting from the master, while the slaves do all the work. This design takes advantage of Nagios's ability to utilize passive checks i.e. external applications or processes that send results back to Nagios. In a distributed configuration, these external applications are other instances of Nagios.

Q10. Explain Main Configuration file of Nagios and its location?

First mention what this main configuration file contains and its function. The main configuration file contains a number of directives that affect how the Nagios daemon operates. This config file is read by both the Nagios daemon and the CGIs (It specifies the location of your main configuration file).
Now you can tell where it is present and how it is created. A sample main configuration file is created in the base directory of the Nagios distribution when you run the configure script. The default name of the main configuration file is nagios.cfg. It is usually placed in the etc/ subdirectory of you Nagios installation (i.e. /usr/local/nagios/etc/).

Q11. Explain how Flap Detection works in Nagios?

I will advise you to first explain Flapping first. Flapping occurs when a service or host changes state too frequently, this causes lot of problem and recovery notifications.
Once you have defined Flapping, explain how Nagios detects Flapping. Whenever Nagios checks the status of a host or service, it will check to see if it has started or stopped flapping. Nagios follows the below given procedure to do that:

- Storing the results of the last 21 checks of the host or service analyzing the historical check results and determine where state changes/transitions occur
- Using the state transitions to determine a percent state change value (a measure of change) for the host or service
- Comparing the percent state change value against low and high flapping thresholds

A host or service is determined to have started flapping when its percent state change first exceeds a high flapping threshold. A host or service is determined to have stopped flapping when its percent state goes below a low flapping threshold.

Q12. What are the three main variables that affect recursion and inheritance in Nagios?

According to me the proper format for this answer should be:
First name the variables and then a small explanation of each of these variables: