

Git and git hub

1. What is Git?

Git is a version control system for tracking changes in computer files and is used to help coordinate work among several people on a project while tracking progress over time. In other words, it's a tool that facilitates source code management in software development.

Git favors both programmers and non-technical users by keeping track of their project files. It enables multiple users to work together and handles large projects efficiently.

2. What do you understand by the term 'Version Control System'?

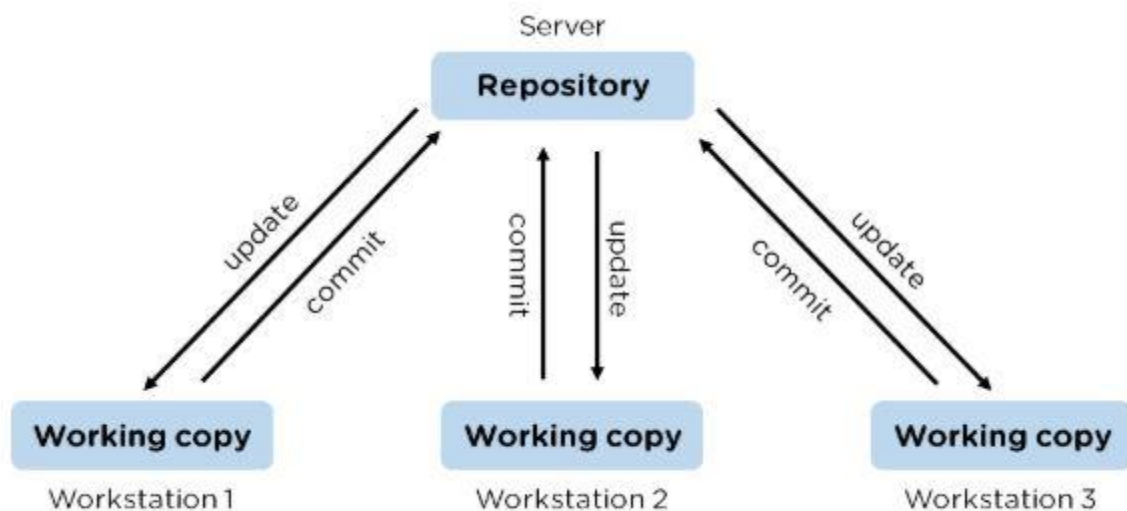
A version control system (VCS) records all the changes made to a file or set of data, so a specific version may be called later if needed.

This helps ensure that all team members are working on the latest version of the file

2. What do you understand by the term 'Version Control System'?

A version control system (VCS) records all the changes made to a file or set of data, so a specific version may be called later if needed.

This helps ensure that all team members are working on the latest version of the file



3. What is GitHub?

To provide Internet hosting for version control and software development, GitHub makes use of Git.

4. Mention some popular Git hosting services.

1.GitHub 2.Source Forge 3.GitLab 4.Bitbucket

5. Different types of version control systems

- Local version control systems have a database that stores all file changes under revision control on disc in a special format.
- Centralized version control systems have a single repository, from which each user receives their working copy.
- Distributed version control systems contain multiple repositories, and different users can access each one with their working copy.

6. What benefits come with using GIT?

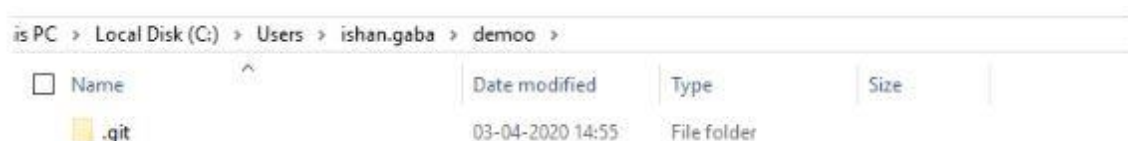
- Data replication and redundancy are both possible.
- It is a service with high availability.
- There can only be one Git directory per repository.
- Excellent network and disc performance are achieved.
- On any project, collaboration is very simple.

7. What's the difference between [Git and GitHub](#)?

Git	GitHub
Git is a software	GitHub is a service
Git can be installed locally on the system	GitHub is hosted on the web
Provides a desktop interface called git GUI	Provides a desktop interface called GitHub Desktop.
It does not support user management features	Provides built-in user management

8. What is a Git repository?

Git repository refers to a place where all the Git files are stored. These files can either be stored on the local repository or on the remote repository.



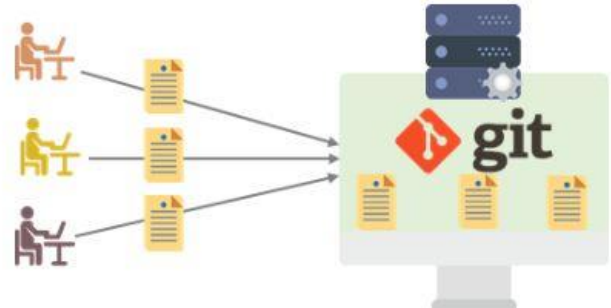
9. How can you initialize a repository in Git?

If you want to initialize an empty repository to a directory in Git, you need to enter the git init command. After this command, a hidden .git folder will appear.

```
SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo
$ pwd
/c/Users/Taha/git_demo/FirstRepo

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo
$ git init
Initialized empty Git repository in c:/Users/Taha/git_demo/FirstRepo/.git/

SL-LP-DNS-0223+Taha@SL-LP-DNS-0223 MINGW64 ~/git_demo/FirstRepo (master)
$
```



10. How is Git different from Subversion (SVN)?

GIT	SVN
Git is a distributed decentralized version control system	SVN is a centralized version control system.
Git stores content in the form of metadata.	SVN stored data in the form of files.
The master contains the latest stable release.	In SVN, the trunk directory has the latest stable release
The contents of Git are hashed using the SHA-1 hash algorithm.	SVN doesn't support hashed contents.

11. Name a few Git commands with their function.

- Git config - Configure the username and email address
- Git add - Add one or more files to the staging area
- Git diff - View the changes made to the file
- Git init - Initialize an empty Git repository
- Git commit - Commit changes to head but not to the remote repository

12. What are the advantages of using Git?

- 1.Faster release cycles
- 2.Easy team collaboration
- 3.Widespread acceptance
- 4.Maintains the integrity of source code
- 5.[Pull requests](#)

13. What language is used in Git?

Git is a fast and reliable version control system, and the language that makes this possible is 'C.'

Using [C language](#) reduces the overhead of run times, which are common in high-level languages.

14. What is the correct syntax to add a message to a commit?

```
git commit -m "x files created"
```

15. Which command is used to create an empty Git repository?

git init - This [command](#) helps to create an empty repository while working on a project.

16. What does git pull origin master do?

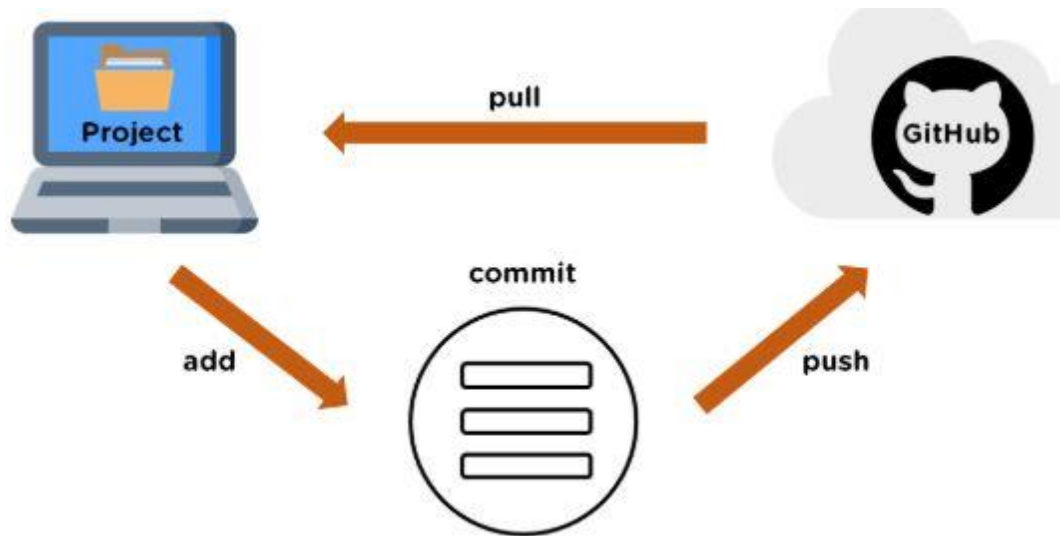
The git pull origin master fetches all the changes from the master branch onto the origin and integrates them into the local branch.

```
git pull = git fetch + git merge origin/ master
```

Intermediate Git Interview Questions

17. What does the git push command do?

The [Git push command](#) is used to push the content in a local repository to a remote repository. After a local repository has been modified, a push is executed to share the modifications with remote team members.



18. Difference between git fetch and git pull.?

Git Fetch	Git Pull
The Git fetch command only downloads new data from a remote repository.	Git pull updates the current HEAD branch with the latest changes from the remote server.
It does not integrate any of these new data into your working files.	Downloads new data and integrate it with the current working files.
Command - git fetch origin git fetch --all	Tries to merge remote changes with your local ones. Command - git pull origin master

19. GitHub, GitLab and Bitbucket are examples of git repository _____ function?

hosting. All the three are services for hosting Git repositories

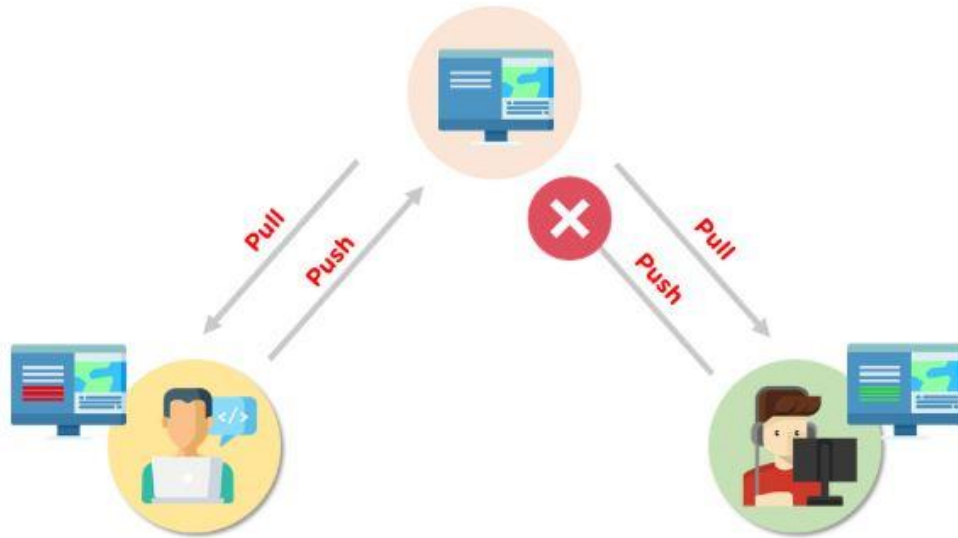
20. How do you resolve conflicts in Git?

Here are the steps that will help you resolve conflicts in Git:

- Identify the files responsible for the conflicts.
- Implement the desired changes to the files
- Add the files using the git add command.
- The last step is to commit the changes in the file with the help of the git commit command.

21. What do you understand about the Git merge conflict?

A [Git merge conflict](#) is an event that occurs when Git is unable to resolve the differences in code between the two commits automatically. Git is capable of automatically merging the changes only if the commits are on different lines or branches.



22. What is the functionality of git ls-tree?

The git ls-tree command is used to list the contents of a tree object.

23. What is the process to revert a commit that has already been pushed and made public?

There are two processes through which you can revert a commit:

1. Remove or fix the bad file in a new commit and push it to the remote repository. Then commit it to the remote repository using:

```
git commit -m "commit message"
```

2. Create a new commit to undo all the changes that were made in the bad commit. Use the following command:

```
git revert <commit id>
```

24. How is a bare repository different from the standard way of initializing a Git repository?

Standard way	Bare way
You create a working directory with the git init command.	Does not contain any working or checked out copy of source files.
A .git subfolder is created with all the git-related change history.	Bare repositories store git revision history in the root folder of your repository instead of the .git subfolder.

25. What does git clone do?

Git clone allows you to create a local copy of the remote GitHub repository. Once you clone a repo, you can make edits locally in your system rather than directly in the source files of the remote repo

26. What is Git stash?

Let's say you're a developer and you want to switch branches to work on something else. The issue is you don't want to make commits in uncompleted work, so you just want to get back to this point later. The solution here is the Git stash.

Git stash takes your modified tracked files and saves it on a stack of unfinished changes that you can reapply at any time. To go back to the work you can use the stash pop.

27. What does the git reset --mixed and git merge --abort commands do?

git reset --mixed is used to undo changes made in the working directory and staging area.

git merge --abort helps stop the merge process and return back to the state before the merging began.

28. What do you understand about the Staging area in Git?

The Staging Area in Git is when it starts to track and save the changes that occur in files. These saved changes reflect in the .git directory. Staging is an intermediate area that helps to format and review commits before their completion.

29. What is Git Bisect and how do you use it?

The Git Bisect command performs a binary search to detect the commit which introduced a bug or regression in the project's history.

Syntax: git bisect <subcommand> <options>

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

The command to get a list of files that has been changed in a particular commit is: git diff-tree -r {commit hash}

- -r flag allows the command to list individual files
- commit hash lists all the files that were changed or added in the commit.

31. What is the use of the git config command?

The git config command is used to set git configuration values on a global or local level. It alters the configuration options in your git installation. It is generally used to set your Git email, editor, and any aliases you want to use with the git command.

32. What is the functionality of git clean command?

The git clean command removes the untracked files from the working directory.

33. What is SubGit and why is it used?

SubGit is a tool that is used to migrate SVN to Git. It transforms the SVN repositories to Git and allows you to work on both systems concurrently. It auto-syncs the SVN with Git.

34. If you recover a deleted branch, what work is restored?

The files that were stashed and saved in the stashed index can be recovered. The files that were untracked will be lost. Hence, it's always a good idea to stage and commit your work or stash them.

35. Explain these commands one by one– git status, git log, git diff, git revert <commit>, git reset <file>.

- Git status - It shows the current status of the working directory and the staging area.
- Git revert<commit> - It is used for undoing changes to a repository's commit history.
- Git log- It is a key tool for reviewing and reading the history of everything that happens to a repository.
- Git diff- It is a multi-purpose Git command that performs a diff function on Git data sources when executed.
- Git reset<file>- it is used to unstage a file.

36. What exactly is tagging in Git?

Tagging enables developers to mark all significant checkpoints as their projects progress.

37. What exactly is forking in Git?

It is a repository duplicate and forking allows one to experiment with changes without being concerned about the original project.

38. How to change any older commit messages?

You can change the most recent commit message with the `git commit —amend` command.

39. How to handle huge binary files in Git?

Git LFS is a Git extension for dealing with large and binary files in a separate Git repository.

40. Name a few GIT tools.

Git comes with a few built-in tools like Git Bash and Git GUI.

41. Will you make a new commit or amend an existing one?

The `git commit —amend` command allows you to easily modify the most recent commit.

42. What do you mean by branching strategy?

It is employed by a software development team while writing and managing code with a version control system.

43. Difference between head, working tree, and index.

They are all names for various branches. Even Though a single git repository can track an arbitrary number of branches, the working tree is only associated with one of them, and HEAD points to that branch.

44. Is there a git GUI client available for Linux?

Git includes built-in GUI tools for committing (`git-gui`) and browsing (`gitk`), but there are a number of third-party tools available for users seeking platform-specific experience.

45. What is the benefit of a version control system?

Version control enables software teams to maintain efficiency and agility while the team grows by adding more developers

46. What do you mean by git instaweb?

It is a script used to set up a temporary instance of Gitweb.

47. What exactly is the forking workflow?

Forking is a git clone operation that is performed on a server copy of a project's repository.

48. Mention benefits of forking workflow.

Contributions can be integrated without everyone trying to push to a single central repository.

49. What is the Gitflow workflow?

The Gitflow Workflow specifies a strict branching model centered on the project release.

50. What does the commit object contain?

The commit object contains a tree of blob objects and other tree objects that represent the project revision's directory structure.

51. Write the syntax of rebasing in git.

Syntax is as follows: `$git rebase <branch name>`

52. What are Git Hooks?

They are scripts that are executed automatically whenever a specific event occurs in a Git repository.

53. What is Git stash vs Git stash pop?

Git stash pop removes the (topmost, by default) stash when applied, whereas git stash apply keeps it in the stash list for future use.

54. Explain git reflog This command is used by Git to record changes made to the branches' tips.

ans:

55. Role of the git annotate command.

In git, it is used to track each line of the file based on the commit information.

56. What is a git Directory?

It is the storage place of the metadata and object database of the project.

57. How can a conflict be settled in Git?

Edit the files to resolve any incompatible changes first, then use "git add" to add the corrected files and "git commit" to save the repaired merge.

58. What is the standard method for branching in GIT?

In GIT, the best way to create a branch is to have one 'main' branch and then another branch for implementing the changes that we want to make.

59. How do you set up a Git repository?

If you want to add an empty repository to a directory in Git, use the git init command.

60. What is the proper syntax for appending a message to a commit?

Git commit -m "x files created" is the syntax.

61. Use of git instaweb.

It is used to launch a web browser and a webserver with an interface into a local repository automatically.

62. Describe git ls-tree.

It represents a tree object with each item's mode and name included.

63. What exactly is git cherry-pick?

A command typically used to move specific commits from one branch of a repository to another.

64. State the difference between “git remote” and “git clone”?

“Git remote” allows you to create an entry in the git configuration which specifies a URL.

“Git clone” lets you create a new git repository by letting you copy it from the current URL.

65. Difference between “pull request” and “branch”?

“Pull request” is done when you feel like changing the developer's change to another person's code branch. And “Branch” is just a separate version of code.

66. How might you recover a branch that has previously pushed changes in the main repository yet has been coincidentally erased from each team member's local machines?

We can easily recover this by seeing the latest commit of the branch in the reflog and then going through the new branch.

67. What is a detached head?

Detach head refers to that the currently checked repository is not in the local branch.

68. What command helps us to know the branches merged into master and which are not?

`git branch - -merged` lets us get the list of the branches which are currently merged into the current branch

`git branch - -no-merged` shows the branches which are not merged

69. Is LDAP Authentication Supported?

GitLab API only supports LDAP authentication since version 6.0 and higher.

Popular Git Interview Questions

70. A simple definition of Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

71. Is the C++ language used in Git?

Yes, but C programming language is a widely used language in Git.

72. How does Git work?

Git works by tracking changes to files in a project and allowing developers to easily revert to previous versions if necessary. Git also makes it easy to collaborate on projects, as it allows multiple developers to work on the same codebase simultaneously.

73. What are some of the most popular Git commands?

Some of the most popular Git commands are,

- `"git init"` (which initializes a Git repository)
- `"git add"` (which adds files to a Git repository)
- And `"git commit"` (which saves changes to a Git repository).

74. List out the functions provided by Git repository hosting service

There are many functions that a Git repository hosting service can provide. Some of the most common and useful functions include:

- Providing a web interface for users to interact with the repositories
- Allowing users to clone repositories
- Allowing users to view and download statistics about the repositories
- Providing a way for users to push changes to the repositories
- Keeping track of changes made to the repositories

75. What is the difference between Git and other revision control systems?

- Git is a distributed revision control system, which means that it can be used without a central server. This allows for a great deal of flexibility in how projects are managed.
- On the other side, revision control systems are often centralized, which can limit the flexibility of how projects are managed.

76. How does Git work?

Git works by tracking changes to files in a repository. When a file is changed, Git calculates a unique identifier for the change, called a "commit hash". The commit hash allows Git to identify the change and track it over time.

77. How do I install Git?

Installing Git is simple. Just download the latest version from the Git website (<https://git-scm.com/>).

78. How do I use Git?

To use Git, a developer first creates a local repository on their computer. This repository contains all the files for a project and the history of all the changes made to those files. Or just follow the instructions in the Git documentation (<https://git-scm.com/doc>).

79. What are some of the drawbacks of Git?

- One of the main drawbacks is that it can be difficult to learn and use, especially for those who are not familiar with version control systems.
- And Git is not always reliable and can sometimes be slow.

80. What are some of the most important commands in Git?

Some of the most important Git commands are "commit", "push", and "pull".

- The "commit" command is used to save changes to the local repository.
- And the "push" command is used to send changes to the remote repository.
- Then the "pull" command is used to retrieve changes from the remote repository.

81. What are some of the most important features of Git?

Some of the most important features of Git are its distributed nature, its ability to track changes, and its support for branches.

- The distributed nature of Git allows developers to work independently and offline.
- The ability to track changes helps developers to keep track of their work and revert to previous versions if necessary.
- The support for branches allows developers to experiment with new features without affecting the main codebase.

82. What is a branch in Git?

A branch is a way to isolate development work on a particular aspect of a project. When a branch is created, it diverges from the primary branch. It allows developers to work on a new feature or bug fix without affecting the main codebase.

83. What is a commit in Git?

A commit is a way to save changes to a branch. When a commit is made, a snapshot of the current state of the branch is created. This snapshot can be used to revert the branch to that state if necessary.

84. What is conflict in Git?

Conflict in Git occurs when two or more developers have made changes to the same part of a file, and those changes can't be automatically merged. When this happens, Git will mark the file as conflicted and leave it up to the developers to resolve the conflict.

Resolving a conflict can be done by manually editing the file to choose which changes should be kept, or by using a tool like Git's merge command to automatically merge the changes.

85. What does the git status command do?

The git status command is used to obtain the current state of a Git repository. This command can be used to determine whether the repository is clean or dirty, and to see which files have been modified. The git status command will also show which branch is currently checked out and whether there are any uncommitted changes.

86. Why is it considered to be easy to work on Git?

There are many reasons that Git is considered an easy tool to work with.

- It has a straightforward learning curve. Even those new to programming can easily learn how to use Git with just a few hours of practice.
- Git is highly flexible and can be easily customized to fit the needs of any project.
- And, Git is very stable and reliable, so users can trust that their work will be safe and sound.

87. What do you know about Git Stash?

Git stash is a powerful tool that allows you to save your changes and revert your working directory to a previous state. This is especially useful when switching branches or reverting to a previous commit. And Git Stash takes a snapshot of your changes and stores them away for later use.

88. What differentiates between the commands git remote and git clone?

The main difference between the git remote and git clone commands is that the git remote adds a remote repository as a shortcut to your current repository, while the git clone creates an entirely new copy of a remote repository.

89. Tell me the difference between git pull and git fetch?

Both of these commands will fetch any new commits from the remote repository, but they differ in how they handle these commits.

Git pull will merge the remote commits into the current branch, while git fetch will simply retrieve the commits and store them in the local repository. This means that if you have any uncommitted changes, git pull may result in merge conflicts, while git fetch will not.

90. Is Git and GitHub the same thing?

No, Git and GitHub are two different things.

- Git is a version control system that lets you track changes to your code.
- GitHub is a hosting service for Git repositories. You can use GitHub to store your code remotely, or you can use it to collaborate with other developers on a project.

96. Explain the different points when a merge can enter a conflicted stage.

There are two stages when a merge can enter a conflicted stage.

1. **Starting the merge process:** If there are changes in the working directory of the stage area in the current project, the merge will fail to start. In this case, conflicts happen due to pending changes that need to be stabilized using different Git commands.
2. **During the merge process:** The failure during the merge process indicates that there's a conflict between the local branch and the branch being merged. In this case, Git resolves as much as possible, but some things have to be fixed manually in the conflicted files.

91. What about Git reflog?

Git reflog is a history of all the changes made to a git repository. It is a valuable tool for debugging and troubleshooting purposes.

And Git reflog can be used to view the history of a repository, see who made what changes, and when those changes were made.

92. What is a detached head?

A detached HEAD is a state where the HEAD pointer is not pointing to the current commit. This can happen if you check out a commit that is not the most recent, or if you reset your head to a previous commit.

93. How to avoid a detached head?

There are a few different ways to avoid a detached HEAD.

- The first is to simply commit your changes before switching branches. This will ensure that your changes are saved to a specific branch, and you won't have to worry about them being lost when you switch branches.

- Another way to avoid detached HEAD is to use the "git checkout" command with the "-b" option.

94. How will you resolve conflict in Git?

To resolve a conflict in Git, you will need to first identify the source of the conflict. Once you have identified the source of the conflict, you can use the "git pull" command. This will pull the latest changes from the remote repository and merge them with your local copy.

If the "git pull" command doesn't resolve the conflict, you can try the "git merge" command. This will merge the two versions of the code manually. You will need to resolve the conflicts manually and then commit the merged code.

95. What is Subgit and where do you use Subgit?

Subgit is a tool for managing Git repositories with Subversion history. It allows you to keep your existing subversion history while moving to Git, and it also provides a way to keep your Git history synchronized with subversion.

There are many reasons that you might want to use Subgit,

- For example, if you have a large subversion repository with a lot of history, moving to Git can be a huge undertaking; that's where Subgit can help transition smoother.
- And if you work in an environment where subversion is the primary version control system, using Subgit can help you keep your Git history in sync with the rest of the team.

97. What has to be run to squash the last N commits into a single commit?

In Git, squashing commits means combining two or more commits into one.

Use the below command to write a new commit message from the beginning.

```
git reset -soft HEAD~N &&git commit
```

But, if you want to edit a new commit message and add the existing commit messages, then you must extract the messages and pass them to Git commit.

The below command will help you achieve this:

```
git reset -soft HEAD~N &&git commit -edit -m“$(git log -format=%B -reverse  
.HEAD@{N})”
```

98. What is the difference between fork, branch, and clone?

Fork	Branch	Clone
The fork is the process when a copy of the repository is made. It's usually experimentation in the project without affecting the original project. They're used to advise changes or take inspiration from someone else's project.	Git branches refer to individual projects within a git repository. If there are several branches in a repository, then each branch can have entirely different files and folders.	Git clone refers to creating a clone or a copy of an existing git repository in a new directory. Cloning automatically creates a connection that points back to the original repository, which makes it very easy to interact with the central repository.

99. What is the command used to fix a broken commit?

To fix a broken commit in Git, you may use the “git commit --amend” command, which helps you combine the staged changes with the previous commits instead of creating an entirely new commit.

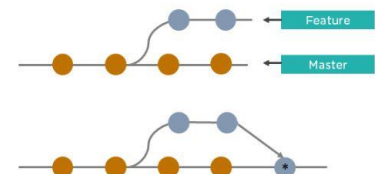
100. How do you recover a deleted branch that was not merged?

To recover a deleted branch, first, you can use the git reflog command. It will list the local recorded logs for all the references. Then, you can identify the history stamp and recover it using the git checkout command.

101. How is Git merge different from Git rebase?

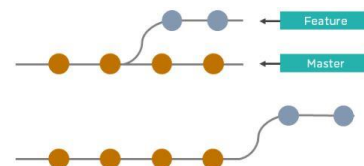
Git merge is used to incorporate new commits into your feature branch.

- Git merge creates an extra merge commit every time you need to incorporate changes.
- It pollutes your feature branch history.



As an alternative to merging, you can rebase the feature branch into master.

- Git rebase Incorporates all the new commits in the master branch.
- It rewrites the project history by creating brand new commits for each commit in the original branch



102. What is git stash drop?

The Git stash drop command is used to remove a particular stash. If there's a stash you're no longer using or you want to remove a specific item of stash from the list, you can use the stash commands.

Let's say you want to delete an item named `stash@{abc}`; you can use the command:

```
git stash drop stash@{abc}.
```

103. What's the difference between reverting and resetting?

Reverting	Resetting
The revert command in Git is used to create a new commit that undoes the changes made in the previous commit. When you use this command, a new history is added to the project; the existing history is not modified	Git reset is a command that is used to undo the local changes that have been made to a Git repository. Git reset operates on the following: commit history, the staging index, and the working directory.

105. What is “git cherry-pick”?

The command `git cherry-pick` enables you to pick up commits from a branch within a repository and apply it to another branch. This command is useful to undo changes when any commit is accidentally made to the wrong branch. Then, you can switch to the correct branch and use this command to cherry-pick the commit.

104. How can you discover if a branch has already been merged or not?

There are two commands to determine these two different things.

`git branch --merged` - Returns the list of branches that have been merged into the current branch.

`git branch --no-merged` - Returns the list of branches that have not been merged.

--MAVEN--

1. What is Maven?

[Maven is a popular open-source](#) build tool developed by the Apache Group to build, publish, and deploy several projects. It is [written in Java](#) and is used to build projects written in C#, Scala, Ruby, etc.

This tool is used to develop and manage any Java-based project. It simplifies the day-to-day work of Java developers and aids them in their projects.

2. What does Maven help with?

- [Apache Maven](#) helps manage all the processes, such as building, documentation, releasing, and distribution in project management.
- The tool simplifies the process of project building. It increases the performance of the project and the building process.
- The task of downloading JAR files and other dependencies is done automatically.
- Maven provides easy access to all the required information.
- Maven makes it easy for the developer to build a project in different environments, without worrying about the dependencies, processes, etc.
- In Maven, it is simple to add new dependencies, all you need to do is to write the dependency code in the pom file.

3. What are the different elements that Maven takes care of?

In the process, it takes care of the following:

- Builds
- Dependencies
- Reports
- Distribution
- Releases
- Mailing list



4. What is the difference between ANT and Maven?

ANT	Maven
ANT has no formal conventions, so information is to be provided in the build.xml file.	Maven has conventions, so information is not to be provided in the pom.xml file.
ANT is procedural.	Maven is declarative.
ANT has no life cycle.	Maven has a life cycle.
ANT scripts are not reusable.	Maven is mainly used as a project management tool.
ANT is specifically a build tool.	Maven plugins are reusable.

5. What is POM?

Project Object Model (POM) refers to the XML files with all the information regarding project and configuration details.

- It has the description of the project, information regarding the versioning and configuration management of the project.
- The XML file is in the project home directory. When we tend to execute a task, Maven searches for the POM in the current directory.

6. What all is included in the POM?

- Dependencies
- Developers and contributors
- Plugins
- Plugin configuration
- Resources

7. What are the minimum required elements for POM?

The minimum required elements for POM are:

- project root
- model Version – should be 4.0.0
- groupId – project's group id
- artifactId – artifact (project) id
- version – version of the artifact

8. What is meant by the term 'Build Tool'?

A build tool is essential for the process of building. It is needed for the following procedures:

- Generating source code.
- Generating documentation from the source code.
- Compiling of source code.
- Packaging of the compiled codes into JAR files.
- Installing the packaged code in a local repository, server, or central repository.

9. What are the steps to install Maven on Windows?

To install Maven on Windows, observe the following steps:

- Download Maven first, and then extract it.
- In the environment variable, add JAVA_HOME, and MAVEN_HOME.
- Then, add the environment path in the Maven variable.
- Lastly, verify [the Maven installation](#) by checking its version. The command mvn -version will display the version installed in the system.

10. What are the steps to install Maven on Ubuntu?

To [install Maven on to Ubuntu](#), observe the following steps:


- The primary step is to install Java.
- Then, download Maven.
- Configure environment variables JAVA_HOME, M3_HOME, MAVEN_HOME, and PATH.
- Lastly, verify the Maven installation by checking its version.

11. What is the command to install JAR files in the Local Repository?

- mvn install is used to install JAR files in the local repository.
- To install the JAR manually into the local Maven repository, the following plugin is used: mvn install:install-file-Dfile=<path to file>.

12. How do you know the version of Maven being used?

- mvn -version is used to check the version of Maven present in the system.
- The command is typed in the command prompt, and the version of Maven in the system will eventually appear on the screen.



```
C:\Users\Lenovo>mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\Maven\bin\..
Java version: 1.8.0_251, vendor: Oracle Corporation, runtime: C
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family:
C:\Users\Lenovo>
```

13. What is Clean, Default, and Site in Maven?

There are three built-in build life cycles:

- Clean: The clean lifecycle looks after project cleaning.
- Default: The default lifecycle handles the project deployment.
- Site: The site lifecycle refers to the creation of the project's site documentation.

14. What is a Maven Repository?

Maven repositories refer to the directories of packaged JAR files that contain metadata. The metadata refers to the POM files relevant to each project. This metadata is what allows Maven to download dependencies.

There are three types of repositories:

1. Local Repository 2. Remote Repository 3. Central Repository

15. What are the different types of Maven Repositories?

There are three types of Maven repositories:

1. Local Repository:

- Local repository refers to the machine of the developer where all the project material is saved.
- The local repository contains all the dependency jars.

2. Remote Repository:

- The remote repository refers to the repository present on a server that is used when Maven needs to download dependencies.
- Whenever anything is required from the remote repository, it is first downloaded to the local repository, and then it is used.

3. Central Repository:

- Central repository refers to the Maven community that comes into action when there is a need for dependencies, and those dependencies cannot be found in the local repository.
- Maven downloads the dependencies from here in the local repository whenever needed.

16. What is Maven Build Lifecycle?

Maven lifecycle is a collection of steps that are to be followed, to build a project. There are three built-in build lifecycles:

- Default: Handles project deployment.
- Clean: Handles project cleaning.
- Site: Handles the creation of the project site's documentation.

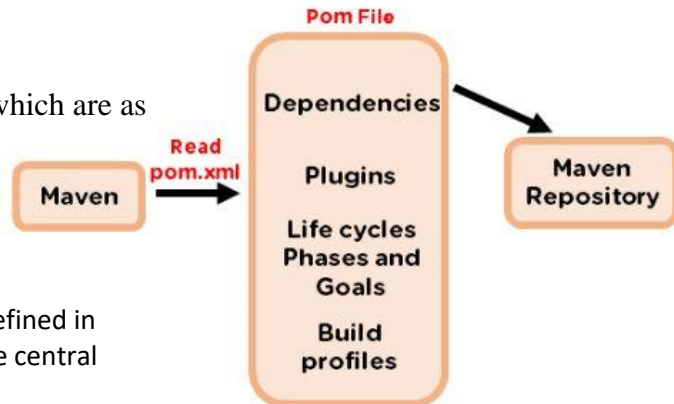
17. How does Maven Architecture work?

Maven architecture works in three steps, which are as follows:

The first step is to read the pom.xml file.

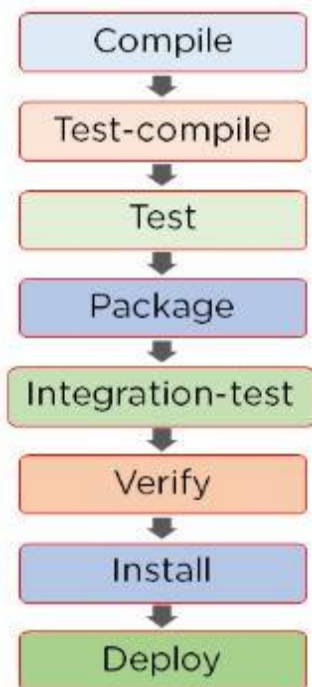
1. Then, it downloads the dependencies defined in pom.xml into the local repository from the central repository.

2. Lastly, it creates and generates a report according to the requirements, and executes life-cycles, phases, goals, plugins, etc.



18. What are the different phases in the Maven Build Lifecycle?

Build Lifecycle has different build phases or stages, which are below:



19. Which command is used to build a Maven site?

- - mvn site command is used to build a Maven site
- The resulting site, by default, is target/site/...

20. What are the different conventions used while naming a project in Maven?

The full name of a project in Maven includes:

<GroupId>: <artifactId>: <version>

For example: org.apache.maven: maven: 2.0.1

21. What is a Maven Artifact?

Maven Artifact refers to a file, usually a JAR that gets deployed to a Maven repository. The tool creates one or more artifacts, such as a compiled JAR and a source JAR.

Every artifact has its groupId, an artifact ID, and a version string. These three together identify the artifact. For example:

com.your.package, any name, and a version string (to uniquely identify).

22. What are the phases of a Clean Life Cycle?

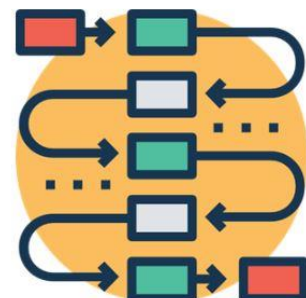
The phases of a clean life cycle include:

- Pre-clean
- Clean
- Post-clean

23. What are the phases of a Site Life Cycle?

The phases of a site life cycle include:

- Pre-site
- Site
- Post-site
- Site-deploy



24. What is meant by Maven Plugins?

- Maven plugins are essential features of Maven that are used to reuse the common build logic across several projects.
- The plugins are needed to carry out tasks like compiling code, testing them, creating JAR files, and documenting the projects.

25. Why are Maven Plugins used?

Maven Plugins are used to:

- Create a JAR file.
- Create a WAR file.
- Compile code files.
- Unit testing of code.
- Create project documentation.
- Create project reports.

26. What are the types of Maven Plugins?

There are two types of Maven Plugins:

- Build plugins – These plugins are executed during the build and are configured in the <build/> element of pom.xml
- Reporting plugins – These plugins are executed during the stage generation and are configured in the <reporting/> element of the pom.xml.

27. Why is it said that “Maven uses convention over configuration”?

- Maven uses convention instead of configuration, due to the fact that the developers only need to create a Maven project. The rest of the structure is automatically created.
- There are many conventions present in Maven for setting up a project, building the artifacts, releasing the code, and running unit tests.

28. What is the difference between Convention and Configuration in Maven?

Convention	Configuration
The convention is when the developers are not required to create the build processes.	The configuration is when developers are supposed to create the build processes manually.
The users do not have to specify the configuration in detail, and once the project is created it will automatically create a structure.	They must specify every configuration in detail.

29. What is Maven's order of inheritance?

The order of inheritance in Maven is:

- Settings
- CLI parameters
- Parent POM
- Project POM

30. What do Build Life Cycles and phases imply in the basic concepts of Maven?

- Build Life Cycles consist of a sequence of build phases, and each build phase consists of a series of goals.
- When a stage is run, all the purposes related to that phase, and its plugins are also compiled.

31. What is the 'Goal' in the Maven terminology?

- The term goal refers to a specific task that makes it possible for the project to be built and organized.
- Phases that are a stage in the life cycle define the sequence in which the desired goals are to be accomplished.

32. What is meant by the term 'Dependencies and Repositories' in Maven?

- Dependencies refer to the Java libraries that are needed for the project. Repositories refer to the directories of packaged JAR files.
- If the dependencies are not present in your local repository; then Maven downloads them from a central repository and stores them in the local repository.

33. What is a 'Snapshot' in Maven?

Snapshot refers to the version available in the Maven remote repository. It signals the latest development copy. Maven inspects for a new version of Snapshot in the remote repository, for every new build. The snapshot is updated by the data service team with an updated source code every time to the repository for each Maven build.

34. What types of projects are available in Maven?

- There are thousands of Java projects provided as templates by Maven.
- This helps the user, as they no longer have to remember every configuration to set up a particular project. For example Spring Project, Spring MVC, Spring Boot, etc.

35. What is a Maven Archetype?

- Maven Archetype refers to a Maven plugin entitled to create a project structure as per its template.
- These archetypes are just project templates that are generated by Maven when any new project is created.

36. What is the command to create a new project based on an archetype?

- The command – mvn archetype: generate, is used to create a new project based on an archetype.
- The command is typed in the command prompt, after navigating to the directory in which the project is residing.

37. What does 'Maven Clean' imply?

- Maven clean is a plugin as the name suggests, that approaches to clean the files and directories generated by Maven at the time of its build.
- The plugin removes the target folder that contains all the class files, docs, JAR files.

38. What is a Build Profile?

- Build profiles refer to the set of configuration values required to build a project using different configurations.
- Different build profiles are added to the POM files while enabling different builds.
- A build profile helps in customizing build for different environments.

39. What are the different types of Build Profiles?

There are three different types of build profiles:

- Per Project - defined in pom.xml
- Per - User - defined in Maven settings.xml
- Global - defined in Maven global settings.xml

40. What is meant by the term ‘System Dependency’?

System dependency refers to the dependency that is present with the scope system. These dependencies are commonly used to help Maven know the dependencies that are provided by the JDK or VM.

System dependencies are mostly used to resolve dependencies on artifacts that are provided by the JDK.

41. What is the reason for using an Optional Dependency?

- Optional dependencies are used to decrease the transitive burden of some libraries.
- These dependencies are used when it is not feasible to divide a project into sub-modules.
- Some dependencies are only used for a specific feature in the project, and if that feature is not there, then that dependency will not be used.

42. What is the ‘Dependency Scope’, and how many types of Dependency Scopes are there?

Dependency scope refers to all the dependencies about the current stage of the build.

The different types of dependency scopes are:

- Compile
- Provided
- Runtime
- Test
- System
- Import

43. What is meant by ‘Transitive Dependency’ in Maven?

- Maven avoids the need to find out and specify libraries that the dependencies require, by including transitive dependencies automatically.
- Transitive dependency says that if X depends on Y and Y depends on Z, then X depends on both Y and Z.

44. How can a Maven Build Profile be activated?

A Maven build profile may be activated in the following ways:

- Explicitly using command console input.
- Through Maven settings.
- Based on environment variables.
- OS Settings.
- Present/missing files.

45. What is meant by ‘Dependency Exclusion’?

The exclusion element is used to exclude any transitive dependency. The concept says if X depends on Y and Y depends on Z, then X can be marked Z as excluded.

46. What is MOJO?

MOJO can be defined as a Maven plain Old Java Object. Every MOJO is an executable goal in Maven, and a plugin refers to the distribution of such MOJOs. MOJO enables Maven to extend the functionality that is already not found in it.

47. What is the command to produce a new project based on a hard drive?

- -mvn archetype: create is used to create a new project.
- The archetype is created after reading the source and resource file, the values of its parameters, and other properties.

48. What is meant by the term ‘Super POM’?

Super POM refers to the default POM of Maven. The POMs of Maven can be derived from a parent or by default. To execute any particular objectives, effective POM is used.

Super POM supports the developers to configure the pom.xml file with the least configurations.

49. What is the Maven settings.xml file?

The Maven settings.xml file contains elements used to define the values needed to configure Maven execution differently.

It contains the following configurations:

- Proxy configuration
- Local repository configuration
- Remote repository configuration
- Central repository configuration

50. Where are Maven dependencies stored?

- All the JARS, dependency files, etc. that are downloaded by Maven are saved in the Maven local repository.
 - The Maven local repository is a folder location on the local system where all the artifacts are locally stored.
-

--Ansible--

What is Ansible?

Ansible is an open-source automation tool, or platform, used for IT tasks such as [configuration management](#), application deployment, intraservice orchestration, and provisioning. Automation is crucial these days, with IT environments that are too complex and often need to scale too quickly for system administrators and developers to keep up if they had to do everything manually. Automation simplifies complex tasks, not just making developers' jobs more manageable but allowing them to focus attention on other tasks that add value to an organization. In other words, it frees up time and increases efficiency. And Ansible, as noted above, is rapidly rising to the top in the world of automation tools. Let's look at some of the reasons for Ansible's popularity.

[Ansible](#) is a configuration and orchestration management tool where applications are deployed automatically in a variety of environments.

Benefits of Ansible

- **Free:** Ansible is an open-source tool.
- **Very simple to set up and use:** No special [coding](#) skills are necessary to use Ansible's playbooks (more on playbooks later).
- **Powerful:** Ansible lets you model even highly complex IT workflows.
- **Flexible:** You can orchestrate the entire application environment no matter where it's deployed. You can also customize it based on your needs.
- **Agentless:** You don't need to install any other software or firewall ports on the client systems you want to automate. You also don't have to set up a separate management structure.
- **Efficient:** Because you don't need to install any extra software, there's more room for application resources on your server.
- Next, in our path to understanding what ansible is, let us find out the features and capabilities of Ansible.

Ansible Features

1. Configuration Management

Ansible is designed to be very simple, reliable, and consistent for configuration management. If you're already in IT, you can get up and running with it very quickly. Ansible configurations are simple data descriptions of infrastructure and are both readable by humans and parsable by machines. All you need to start managing systems is a password or an SSH (Secure Socket Shell, a network protocol) key. An example of how easy Ansible makes configuration management: If you want to install an updated version of a specific type of software on all the machines in your enterprise, all you have to do is write out all the IP addresses of the nodes (also called remote hosts) and write an Ansible playbook to install it on all the nodes, then run the playbook from your control machine.

2. Application Deployment

Ansible lets you quickly and easily deploy multitier apps. You won't need to write custom code to automate your systems; you list the tasks required to be done by writing a playbook, and Ansible will figure out how to get your systems to the state you want them to be in. In other words, you won't have to configure the applications on every machine manually. When you run a playbook from your control machine, Ansible uses SSH to communicate with the remote hosts and run all the commands (tasks).

3. Orchestration

As the name suggests, orchestration involves bringing different elements into a beautifully run whole operation—similar to the way a musical conductor brings the notes produced by all the different instruments into a cohesive artistic work. For example, with application deployment, you need to manage not just the front-end and backend services but the databases, networks, storage, and so on. You also need to make sure that all the tasks are handled in the proper order. Ansible uses automated workflows, provisioning, and more to make orchestrating tasks easy. And once you've defined your infrastructure using the [Ansible playbooks](#), you can use that same orchestration wherever you need to, thanks to the portability of Ansible playbooks.

4. Security and Compliance

As with application deployment, sitewide security policies (such as firewall rules or locking down users) can be implemented along with other automated processes. If you configure the security details on the control machine and run the associated playbook, all the remote hosts will automatically be updated with those details. That means you won't need to monitor each machine for security compliance continually manually. And for extra security, an admin's user ID and password aren't retrievable in plain text on Ansible.

5. Cloud Provisioning

The first step in automating your applications' life cycle is automating the provisioning of your infrastructure. With Ansible, you can provision cloud platforms, virtualized hosts, network devices, and bare-metal servers.

Ansible Architecture

1. Modules

Modules are like small programs that Ansible pushes out from a control machine to all the nodes or remote hosts. The modules are executed using playbooks (see below), and they control things such as services, packages, and files. Ansible executes all the modules for installing updates or whatever the required task is, and then removes them when finished. Ansible provides more than 450 modules for everyday tasks.

2. Plugins

As you probably already know from many other tools and platforms, plugins are extra pieces of code that augment functionality. Ansible comes with a number of its plugins, but you can write your own as well. Action, cache, and callback plugins are three examples.

3. Inventories

All the machines you're using with Ansible (the control machine plus nodes) are listed in a single simple file, along with their IP addresses, databases, servers, and so on. Once you register the inventory, you can assign variables to any of the hosts using a simple text file. You can also pull inventory from sources like [EC2 \(Amazon Elastic Compute Cloud\)](#).

4. Playbooks

Ansible playbooks are like instruction manuals for tasks. They are simple files written in YAML, which stands for YAML Ain't Markup Language, a human-readable data serialization language. Playbooks are really at the heart of what makes Ansible so popular is because they describe the tasks to be done quickly and without the need for the user to know or remember any particular syntax. Not only can they declare configurations, but they can orchestrate the steps of any manually ordered task, and can execute tasks at the same time or at different times.

Each playbook is composed of one or multiple plays, and the goal of a play is to map a group of hosts to well-defined roles, represented by tasks.

5. APIs

Various APIs (application programming interfaces) are available so you can extend Ansible's connection types (meaning more than just SSH for transport), call-back, and more.

Now that we've come this far to understand what Ansible is, let us next look into the Ansible tower.

What is Ansible Tower?

[Ansible Tower](#) is Red Hat's commercial web-based solution for managing Ansible. Its best-known feature is an easy-to-use UI (user interface) for managing configurations and deployments, which is a significant improvement over the original UI. Ansible Tower contains the essential features of Ansible, especially those that are easier to see in a graphical format rather than a text-based format. It is free for up to 10 nodes.

Advantages of Using Ansible With Docker

Ansible does a great job of automating Docker and operationalizing the process of building and deploying containers. If you're managing a traditional IT system, for example, it can be hard to add container-tooling functionality. But Ansible removes the need to do processes manually. There are four main advantages of using Ansible with Docker:

1. Portability/Flexibility

The fact that Ansible playbooks are portable, meaning they can be used anywhere, as well as repeatable, can save you a lot of time and effort. For example, if you use a pure [Dockerfile](#) to build a container, then you can reproduce the application only in a [Docker container](#). If you use an Ansible playbook to create a container, on the other hand, then you can reproduce the application in Docker, on the cloud, and so on.

2. Auditability

Even if you create containers, you'll still need to monitor code and track vulnerabilities. Using Ansible with Docker, you can easily track who has deployed which containers as well as what's in all of the containers, and know that you can rebuild any containers as necessary.

3. Management of Entire Environments

With Ansible, you already know you can manage your Docker containers. But you can also maintain the environment that all the containers are in, even in highly complex environments. Ansible can monitor containers and non-container at the same time, which is essential because containerized applications often need to "talk" with noncontainerized applications.

4. Similar Syntax

As mentioned, Ansible used YAML files for its playbooks. Docker uses its non-YAML scripts, but they are very similar and can do almost the same things.

In order to get a complete understanding of what Ansible is, we will learn how Ansible can be used with Docker.

What's the Solution?

With Ansible, a system administrator can write simple codes that are deployed to all systems, and they can be configured to the correct states.

1. Let's begin with the basics. What is Ansible?

Ansible is an open-source platform that facilitates configuration management, task automation, or application deployment. It is a valuable DevOps tool. It was written in [Python](#) and powered by Red Hat. It uses SSH to deploy SSH without incurring any downtime.

2. List Ansible's advantages

Ansible has many strengths, including:

- ✚ It's agentless and only requires SSH service running on the target machines
- ✚ Python is the only required dependency and, fortunately, most systems come with the language pre-installed
- ✚ It requires minimal resources, so there's low overhead
- ✚ It's easy to learn and understand since Ansible tasks are written in YAML.
- ✚ Unlike other tools, most of which are Procedural, ansible is declarative; define the desired state, and Ansible fulfills the requirements needed to achieve it

3. What are CD and CI, and what is Ansible's relationship with them?

CD stands for continuous delivery, and CI stands for [continuous integration](#); both are software development practices.

In CD, developers build software that can be released into production at any given time. CI, on the other hand, consists of each developer uploading regularly scheduled integrations (usually daily), resulting in multiple integrations every day. Ansible is an ideal tool for [CI/CD processes](#), providing a stable infrastructure for provisioning the target environment and then deploying the application to it.

4. Describe how Ansible works.

This is one of the most frequently asked ansible interview questions where the interviewer wants to know whether you actually know the tool in and out or not. You can start this way - ansible is broken down into two types of servers: controlling machines and nodes. [Ansible is installed](#) on the controlling computer, and the controlling machines manage the nodes via SSH.

The controlling machine contains an inventory file that holds the node system's location. Ansible runs the playbook on the controlling machine to deploy the modules on the node systems. Since Ansible is agentless, there's no need for a third-party tool to connect the nodes.

5. State the requirements for the Ansible server.

You need a virtual machine with Linux installed on it, running with Python version 2.6 or higher.

6. Explain what a “playbook” is.?

A playbook has a series of YAML-based files that send commands to remote computers via scripts. Developers can configure entire complex environments by passing a script to the required systems rather than using individual commands to configure computers from the command line remotely. Playbooks are one of Ansible’s strongest selling points and often referred to as the tool’s building blocks.

7. How do you set up Ansible?

You can use either the Python installer or a Linux-based installation process, such as apt or yum.

8. What is Ansible Tower?

It’s an enterprise-level web-based solution that increases Ansible’s accessibility to other [IT teams](#) by including an easy-to-use UI (user interface). Tower’s primary function is to serve as the hub for all of an organization’s automation tasks, allowing users to monitor configurations and conduct rapid deployments.

9. What is “idempotency”?

Idempotency is an important Ansible feature. It prevents unnecessary changes in the managed hosts. With idempotency, you can execute one or more tasks on a server as many times as you need to, but it won’t change anything that’s already been modified and is working correctly. To put it in basic terms, the only changes added are the ones needed and not already in place.

10. What is Ansible Galaxy?

This is a tool bundled with Ansible to create a base directory structure. Galaxy is a website that lets users find and share Ansible content. You can use this command to download roles from the website:

```
$ ansible-galaxy install username.role_name
```

11. How do you use Ansible to create encrypted files?

To create an encrypted file, use the ‘ansible-vault create’ command.

```
$ ansible-vault create filename.yaml
```

You will get a prompt to create a password, and then to type it again for confirmation. You will now have access to a new file, where you can add and edit data.

12. What are “facts” in the context of Ansible?

Facts are newly discovered and known system variables, found in the playbooks, used mostly for implementing conditionals executions. Additionally, they gather ad-hoc system information.

You can get all the facts by using this command:

```
$ ansible all- m setup
```

13. Explain what an ask_pass module is.

It's a playbook control module used to control a password prompt. It's set to True by default.

14. What's an ad hoc command?

Users initiate ad hoc commands to initiate actions on a host without using a playbook. Consider it a one-shot command.

15. Explain the difference between a playbook and a play.

A play is a set of tasks that run on one or more managed hosts. Plays consist of one or more tasks. A playbook consists of one or more plays.

16. What exactly is a configuration management tool?

[Configuration management tools](#) help keep a system running within the desired parameters. They help reduce deployment time and substantially reduce the effort required to perform repetitive tasks. Popular configuration management tools on the market today include [Chef](#), [Puppet](#), Salt, and of course, Ansible.

Finally, let us go through the Ansible interview questions at an advanced level.

17. What are tags?

When there's an extensive playbook involved, sometimes it's more expedient to run just a part of it as opposed to the entire thing. That's what tags are for.

18. Speaking of tags, how do you filter out tasks?

You can filter out tasks in one of two ways:

Use `--tags` or `--skip-tags` options on the command line

If you're in Ansible configuration settings, use the `TAGS_RUN` and `TAGS_SKIP` options.

19. What's a handler?

In Ansible, a handler is similar to a regular task in a playbook, but it will only run if a task alerts the handler. Handlers are automatically loaded by `roles/<role_name>/handlers/main.yaml`. Handlers will run once, after all of the tasks are completed in a particular play.

20. How do you upgrade Ansible?

Upgrading Ansible is easy. Just use this command: `sudo pip install ansible==<version-number>`

21. When do you use `{{ }}`?

One of Ansible's most basic rules is: "Always use `{{ }}` except when:"

22. Explain how to access shell environment variables.

You can access the controlling machine's existing variables by using the "env" lookup plugin. For instance, to access the value of the management machine's home environment variable, you'd enter:

```
local_home:"{{lookup('env','HOME')}}"
```

23. How do you test Ansible projects?

This is another frequently asked ansible interview question. Try elaborating the answer to this question rather than just answering the testing methods in one word. There are three testing methods available:

Asserts

Asserts duplicates how the test runs in other languages like Python. It verifies that your system has reached the actual intended state, not just as a simulation that you'd find in check mode. Asserts shows that the task did the job it was supposed to do and changed the appropriate resources.

Check Mode

Check mode shows you how everything would run if no simulation was done. Therefore, you can easily see if the project behaves the way you want it to. On the downside, check mode doesn't run scripts and commands used in roles and playbooks. To get around this, you have to disable check mode for specific tasks by running "check_mode: no."

Manual Run

Just run the play and verify that the system is in its desired state. This testing choice is the easiest method, but it carries an increased risk because the results in a test environment may not be the same in a production environment.

24. How do you keep data secret in a playbook?

If you want to keep secret data but still be able to share it publicly, then use Vault in playbooks. But if you're using -v (verbose) mode and don't want anyone to see the results, then use:

```
name: secret task
```

```
shell: /usr/bin/do_something --value={{ secret_value }}
```

```
no_log: True
```

--Jenkins--

Q1. What is the difference between Jenkins and Bamboo?

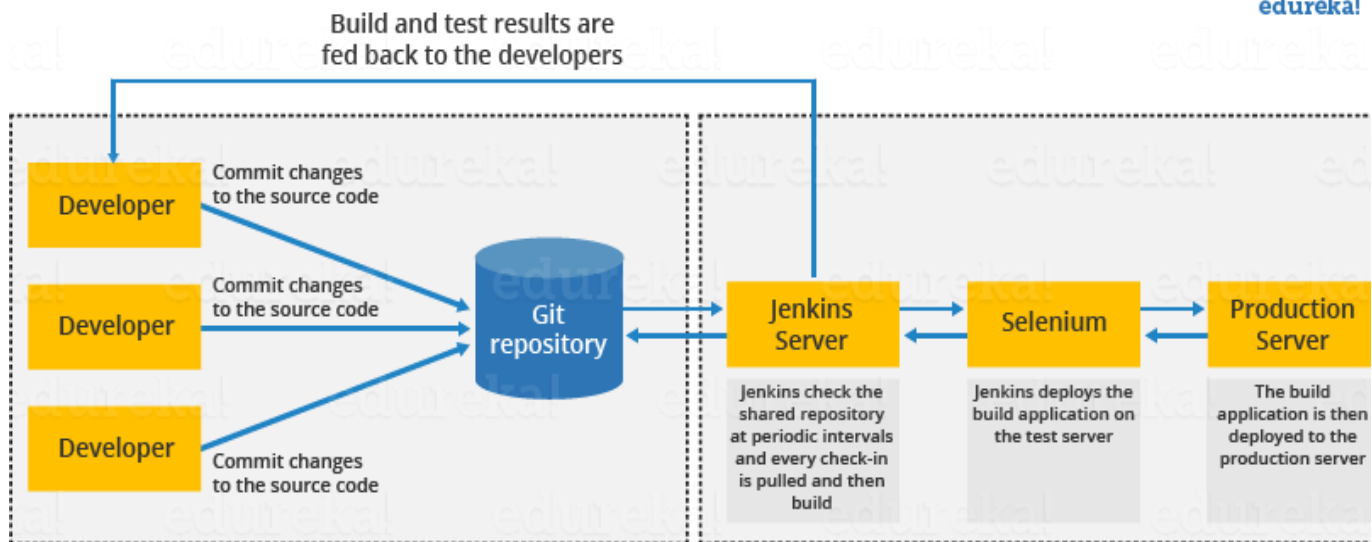
Parameters	Jenkins	Bamboo
Open Source	Jenkin is open-source	Bamboo is not open source
Pricing Logic	Jenkin is completely free	It charges for the number of build agents required
Operating System	Windows, Ubuntu, Red Hat, Mac OS	Windows, Linux, Solaris
Browsers	Chrome, Firefox, Internet Explorer	Firefox, Chrome, Safari, Edge
Plugin Support	Yes, It supports a lot of plugins	It does not support many plugins as compared to Jenkins
Support	Being open-source, it has a lot of support from communities	It has less support as compared to Jenkins

Q2. What is Jenkins?

Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

Q3. Define the process of Jenkins.

- First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If the build is successful, then Jenkins deploys the build in the test server.
- After testing, Jenkins generates feedback and then notifies the developers about the build and test results.
- It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.



Q4. What are the benefits of using Jenkins?

I will suggest you include the following benefits of Jenkins if you can recall any other benefit apart from the below-mentioned points you can include that as well.

- At the integration stage, you can cache build failures.
- For each change in the source code, you generate an automatic build report notification.
- To notify developers about build report success or failure, Jenkins integrates with the LDAP mail server.
- Achieves continuous integration agile development and test driven development.
- With simple steps, you can automate the maven release project.
- Easy tracking of bugs at an early stage in a development environment than production.

Q5. What are the pre-requisites for using Jenkins?

The answer to this is pretty straightforward. To use Jenkins you require:

- A source code repository which is accessible, for instance, a Git repository.
- A working build script, e.g., a Maven script, checked into the repository.

Q6. What is the relation between Hudson and Jenkins?

You can just say Hudson was the earlier name and version of current Jenkins. After some issues, they renamed the project from Hudson to Jenkins.

Q7. How do you install Jenkins?

To install Jenkins, you just need to follow these five steps:

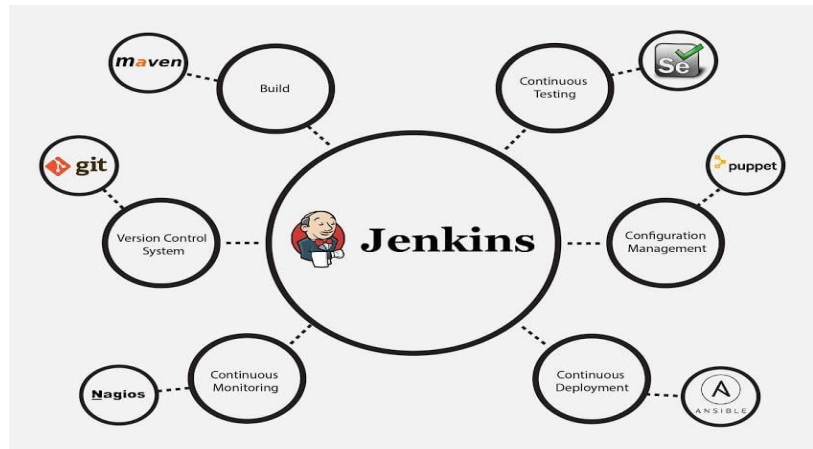
1. **Install Java Version 8** – Jenkins is a Java based application, hence Java is a must.
2. **Install Apache Tomcat Version 9** – Tomcat is essential to deploy Jenkins war file.
3. **Download Jenkins war File** – This war is must to install Jenkins.
4. **Deploy Jenkins war File** – You deploy Jenkins war file using Tomcat to run Jenkins.
5. **Install Suggested Plugins** – Install a list of plugins suggested by Jenkins.

Once the installation is complete, you will be able to see the Jenkins dashboard.

Q8. Mention some of the useful plugins in Jenkins

Below I have mentioned some important Plugins:

- Maven 2 project
- Git
- 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 that as well, but make sure you first mention the above-stated plugins and then add your own.

Q9. What are the two components that you can integrate Jenkins with?

According to me, the integration of Jenkins is possible with the following:

- Version Control system like GIT, SVN.
- Build tools like Apache Maven.

If you have anything else in your mind then mention that as well but make sure you include the above two components in your answer.

Q10. What is Maven? What is the benefit of integrating Maven with Jenkins?

[Maven](#) is a build management tool. It uses a simple pom.xml to configure all the dependencies needed to build, test and run the code. Maven manages the full lifecycle of a test project. Once integrated with Jenkins, the maven Webdriver will build the project and execute all tests efficiently.

Q11. Mention what are the commands you can use to start Jenkins manually.

For this answer I will suggest you go with the below-mentioned flow:

To start Jenkins manually open Console/Command line, then go to your Jenkins installation directory. Over there you can use the below commands:

Start Jenkins: **jenkins.exe start**

Stop Jenkins: **jenkins.exe stop**

Restart Jenkins: **jenkins.exe restart**

Q12. Which SCM tools Jenkins supports?

Here are some of the Source Code Management tools supported by Jenkins:

- AccuRev
- CVS
- Subversion
- Git
- Mercurial
- Perforce
- Clearcase
- RTC

Q13. How will you define Post in Jenkins?

Post is a section that contains several additional steps that might execute after the completion of the pipeline. The execution of all the steps within the condition block depends upon the completion status of the pipeline. The condition block includes the following conditions – **changed success, always, failure, unstable and aborted**.

Q14. What are Parameters in Jenkins?

Parameters are supported by Agent section and they are used to support various use-cases pipelines. Parameters are defined at the top-level of the pipeline or inside an individual stage directive.

Q15. What is Groovy?

Groovy from Apache is a language designed for the Java platform. It is the native scripting language for Jenkins. Groovy-based plugins enhance Jenkins with great interfaces and build reports that are of dynamic and consistent nature.

Q16. How Can You Clone A Git Repository Via Jenkins?

If you want to clone a Git repository via Jenkins, you have to enter the e-mail and user name for your Jenkins system. Switch into your job directory and execute the “git config” command for that.

Q17. How to create a backup and copy files in Jenkins?

The 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.

Q18. Explain how you can set up 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”.

Now 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..

Q19. How will you secure Jenkins?

The way I secure Jenkins is mentioned below if you have any other way to do it than mention that:

- Make sure that the global security is on.
- Check if Jenkins is integrated with my company’s user directory with an appropriate plugin.
- Ensure that the 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 the same.

Q20. Explain how you can deploy a custom build of a core plugin?

Below are the steps to deploy a custom build of a core plugin:

- Stop Jenkins.
- Copy the custom HPI to **\$Jenkins_Home/plugins**.
- Delete the previously expanded plugin directory.
- Make an empty file called **<plugin>.hpi.pinned**.
- Start Jenkins.

Q21. What you do when you see a broken build for your project in Jenkins?

There can be multiple answers to this question I will approach this task in the following way:

I will open the console output for the broken build and try to see if any file changes were missed. If I am unable to find the issue that way, then I will clean and update my local workspace to replicate the problem on my local and try to solve it.

If you do it in a different way then just mention that in your answer.

Q22. What are the various ways in which build can be scheduled in Jenkins?

You can schedule a build in Jenkins in the following ways:

- By source code management commits
- After completion of other builds
- Can be scheduled to run at a specified time (crons)
- Manual Build Requests

Q23. What is the use of Pipelines in Jenkins?

Pipeline plugin is used in Jenkins for making the Jenkins Pipeline, which gives us the view of stages or tasks to perform one after the other in the pipeline form. It models a series of related tasks. Pipelines help the teams to review, edit and iterate upon the tasks. Pipelines are durable and it can optionally stop and wait for human approval as well to start the next task. A pipeline is extensible and can perform work in parallel. It supports complex CD requirements.

Q24. Explain the terms Agent, post-section, Jenkinsfile

Agent: It is directive to tell Jenkins to execute the pipeline in a particular manner and order.

Post-section: If we have to add some notification and to perform other tasks at the end of a pipeline, post-section will definitely run at the end of every pipeline's execution.

Jenkinsfile: The text file where all the definitions of pipelines are defined is called Jenkinsfile. It is being checked in the source control repository.

Q25. Do you know about cloud computing? How can Jenkins fit into a cloud computing environment? Explain with an example.

Let us take the example of AWS cloud service. Cloud computing services use the CI/CD model so that they can push their work to the customers and constantly receive feedback. Jenkins is used to automating the CI/CD pipelines. For example, a lot of Jenkins plugins are available for many of the AWS services like Amazon EC2 and ECS.

Q26. What is Kubernetes? How can you integrate Jenkins with Kubernetes?

[Kubernetes](#) is a container orchestration tool. With Kubernetes, one can create multiple container instances to achieve more fault tolerance. You can use the Kubernetes deploy plugin to use it with Jenkins for continuous deploy.

Q27. Have you run automated tests on Jenkins? How is it done?

Yes, this can be done easily. Automated tests can be run through tools like Selenium or maven. Developers can schedule the test runs. Jenkins displays the test results and sends a report to the developers.

Q28. Let us say, you have a pipeline. The first job was successful, but the second failed. What should you do next?

You just need to restart the pipeline from the point where it failed by doing 'restart from stage'.

Q29. What is the use of JENKINS_HOME directory?

All the settings, logs and configurations are stored in the JENKINS_HOME directory.

Q30. What is a backup plugin? Why is it used?

This is a helpful plugin that backs up all the critical settings and configurations to be used in the future. This is useful in cases when there is a failure so that we don't lose the settings.

Q31. What is a trigger? Give an example of how the repository is polled when a new commit is detected.

Triggers are used to define when and how pipelines should be executed.

When Jenkins is integrated with an SCM tool, for example, Git, the repository can be polled every time there is a commit.

- The Git plugin should be first installed and set up.
- After this, you can build a trigger that specifies when a new build should be started. For example, you can create a job that polls the repository and triggers a build when a change is committed.

Q32. How do you define parameters for a build in Jenkins?

A build can take several input parameters to execute. For example, if you have multiple test suites, but you want to run only one. You can set a parameter so that you are able to decide which one should be run. To have parameters in a job, you need to specify the same while defining the parameter. The parameter can be anything like a string, a file or a custom.

Q33. How does Jenkins authenticate users?

There are 3 ways –

- The default way is to store user data and credentials in an internal database.
- Configure Jenkins to use the authentication mechanism defined by the application server on which it is deployed.
- Configure Jenkins to authenticate against LDAP server.

Q34. What are the ways to configure Jenkins node agent to communicate with Jenkins master?

There are 2 ways to start the node agent –

- **Browser** – if Jenkins node agent is launched from a browser, a JNLP (Java Web Start) file is downloaded. This file launches a new process on the client machine to run jobs.
- **Command-line** – to start the node agent using the command line, the client needs the executable agent.jar file. When this file is run, it simply launches a process on the client to communicate with the Jenkins master to run build jobs.

Q35. How can you use a third-party tool in Jenkins?

Below are the steps used for working with a third-party tool in Jenkins.

- First install the third-party software
- Download the plug-in that supports the third-party tool.
- Configure the third-party tool in the admin console.
- Then use the required plug-in from the Jenkins build job.

For different third-party tools, the procedure may vary slightly, because of the difference in configuration settings.

Q36. What are the types of pipelines in Jenkins?

There are 3 types –

1. CI CD pipeline (Continuous Integration Continuous Delivery)
2. Scripted pipeline
3. Declarative pipeline

Q37. What syntax does Jenkins use to schedule build job or SVN polling?

The cron syntax.

Cron syntax is represented using five asterisks each separated by a space. The syntax is as follows – [minutes] [hours] [day of the month] [month] [day of the week]. Example, if you want to set up a cron for every Monday at 11.59 pm, it would be 59 11 * * 1

Q38. What is DevOps and in which stage does Jenkins fit in?

DevOps is a software development practice that blends software development (Dev) with the IT operations (Ops) making the whole development lifecycle simpler and shorter by constantly delivering builds, fixes, updates, and features. Jenkins plays a crucial role because it helps in this integration by automating the build, test and deployment process.

Q39. Do you know any other Continuous Integration tools? How is Jenkins better than any of those?

There are many other CI tools, and the most prominent ones are –

- TeamCity
- Bamboo
- Perforce
- Circle CI
- Go
- ThoughtWorks
- Integrity
- Travis CI

There are many more. We cannot say if Jenkins is better than each because each has its own unique features. For example, TeamCity offers great .NET support but is complex and costly, Travis CI is free just like Jenkins and has good documentation too. Bamboo too offers efficient and faster builds but is not completely free and so on.

Q40. Name a Jenkins environment variable you have used in a shell script or batch file.

There are numerous environment variables that are available by default in any Jenkins build job. A few commonly used ones include:

- \$JOB_NAME
- \$NODE_NAME
- \$WORKSPACE
- \$BUILD_URL
- \$JOB_URL

Note that, as new Jenkins plug-ins are configured, more environment variables become available. For example, when the Jenkins Git plug-in is configured, new Jenkins Git environment variables, such as \$GIT_COMMIT and \$GIT_URL, become available to be used in scripts.

Q41. What is Continuous Integration In Jenkins?

In software development, multiple developers or teams work on different segments of the same web application. So in this case, you have to perform integration testing by integrating all modules. In order to do that an automated process for each piece of code is performed on a daily bases so that all your codes get tested. This process is known as continuous integration.

Q42. How do you achieve continuous integration using Jenkins?

Here are the steps –

- All the developers commit their source code changes to the shared Git repository.
- Jenkins server checks the shared Git repository at specified intervals and detected changes are then taken into the build.
- The build results and test results are shared to the respective developers
- The built application is displayed on a test server like Selenium and automated tests are run.
- The clean and tested build is deployed to the production server.

Q43. What is a DSL Jenkins?

The Jenkins “Job DSL / Plugin” is made up of two parts – first, The Domain Specific Language (DSL) itself that allows users to describe jobs using a Groovy-based language, and second, a Jenkins plugin which manages the scripts and the updating of the Jenkins jobs which are created and maintained as a result.

Q44. How do you create Multibranch Pipeline in Jenkins?

The Multibranch Pipeline project type enables you to implement different Jenkins files for different branches of the same project. In a Multibranch Pipeline project, Jenkins automatically discovers, manages and executes Pipelines for branches that contain a Jenkins file in source control.

Q.45 What are the types of jobs or projects in Jenkins?

These are the types of jobs/projects in Jenkins –

- Freestyle project
- Maven project
- Pipeline
- Multibranch pipeline
- External Job
- Multi-configuration project
- GitHub organization

Q46. What is blue ocean in Jenkins?

It is a project that was started with the purpose to rethink the user experience of Jenkins, modeling and presenting the process of software delivery by surfacing information that's

important to development teams. This is done with as few clicks as possible, while still staying true to the extensibility that is core to Jenkins. While this project is in the alpha stage of development, the intent is that Jenkins users can install Blue Ocean side-by-side with the Jenkins Classic UI via a plugin

Q47. What is Continuous Testing?

Continuous Testing is the process where you execute automated tests as part of the software delivery pipeline. This is done so that you get the feedback on the business risks associated with software as early as possible. It consists of evolving and extending test automation to address the increased complexity and pace of modern application development and delivery.

Continuous Testing means that testing takes place on a continuous basis without any disruption of any kind. In a Continuous DevOps process, a software change is continuously moving from Development to Testing to Deployment. The code undergoes continuous development, delivery, testing and deployment.

Q48. 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 it below:

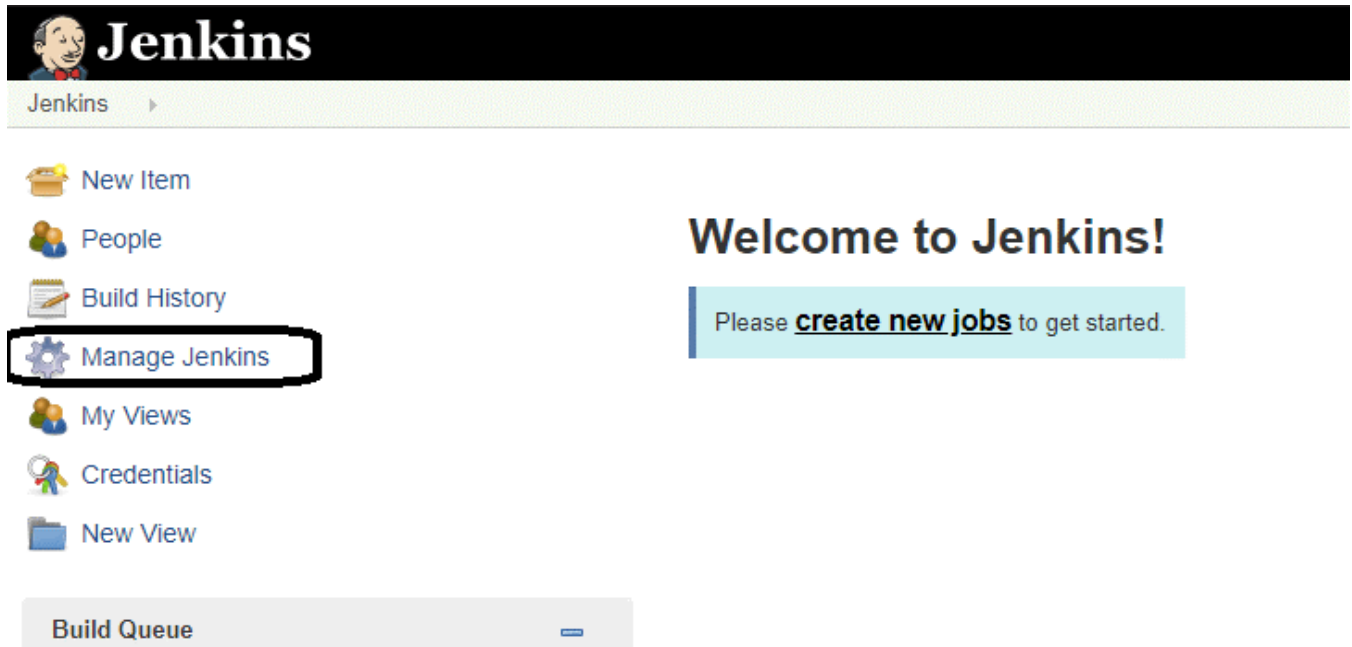
You can:

- 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.

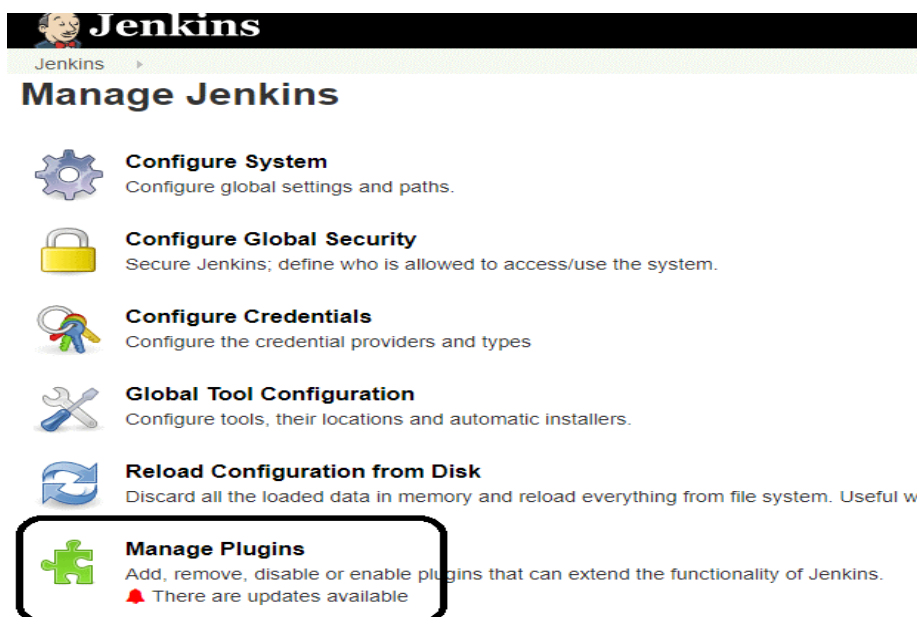
Q49. How do you integrate Git with Jenkins?

These are the steps to integrate [Git](#) with Jenkins –

1. Click on the **Manage Jenkins** button on your Jenkins dashboard:



2. Click on **Manage Plugins**.



3. In the Plugins Page

1. Select the GIT Plugin
2. Click on **Install without restart**. The plugin will take a few moments to finish downloading depending on your internet connection, and will be installed automatically.
3. You can also select the option **Download now and Install after restart**.

<input checked="" type="checkbox"/>	GitHub plugin This plugin integrates GitHub to Jenkins.	1.29.1	Uninstall
<input checked="" type="checkbox"/>	GitHub Branch Source Plugin Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.	2.3.6	Uninstall
<input checked="" type="checkbox"/>	GitHub API Plugin This plugin provides GitHub API for other plugins.	1.92	Uninstall
<input checked="" type="checkbox"/>	GIT server Plugin Allows Jenkins to act as a Git server.	1.7	Uninstall
<input checked="" type="checkbox"/>	Git plugin This plugin integrates Git with Jenkins.	3.9.1	Uninstall
<input checked="" type="checkbox"/>	Git client plugin Utility plugin for Git support in Jenkins	2.7.2	Uninstall

4. You will now see a “No updates available” message if you already have the Git plugin installed.

4. Once you install the plugins , go to **Manage Jenkins** on your Jenkins dashboard. You will see your plugins listed among the rest.

Q50. How can you temporarily turn off Jenkins security if the administrative users have locked themselves out of the admin console?

The JENKINS_HOME folder contains a file named config.xml. When you enable the security, this file contains an XML element named useSecurity that changes to true. If you change this setting to false, security will be disabled the next time Jenkins is restarted.

```
<useSecurity>false</useSecurity>
```

However, we must understand that disabling security should always be both a last resort and a temporary measure. Once you resolve the authentication issues, make sure that you re-enable Jenkins security and reboot the CI server.

Q51. Can you define a Continuous Delivery Workflow?

The flowchart below shows the Continuous Delivery Workflow. Hope it will be much easier to understand with visuals.

Q52. What do you mean by Pipeline as a Code?

Pipeline as Code describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository. These features allow Jenkins to discover, manage, and run jobs for multiple source repositories and branches — eliminating the need for manual job creation and management.

To use Pipeline as Code, projects must contain a file named **Jenkinsfile** in the repository root, which contains a “Pipeline script.”

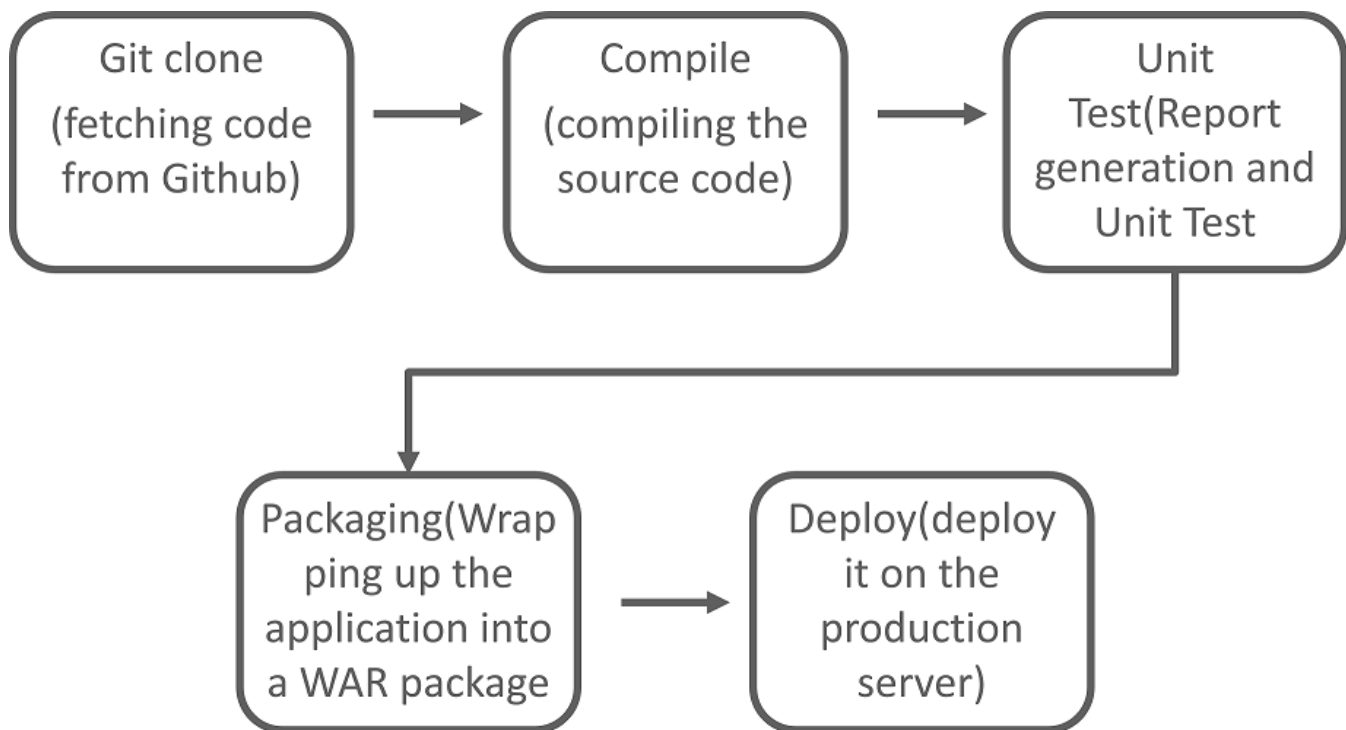
Additionally, one of the enabling jobs needs to be configured in Jenkins:

- **Multibranch Pipeline:** build multiple branches of a *single* repository automatically

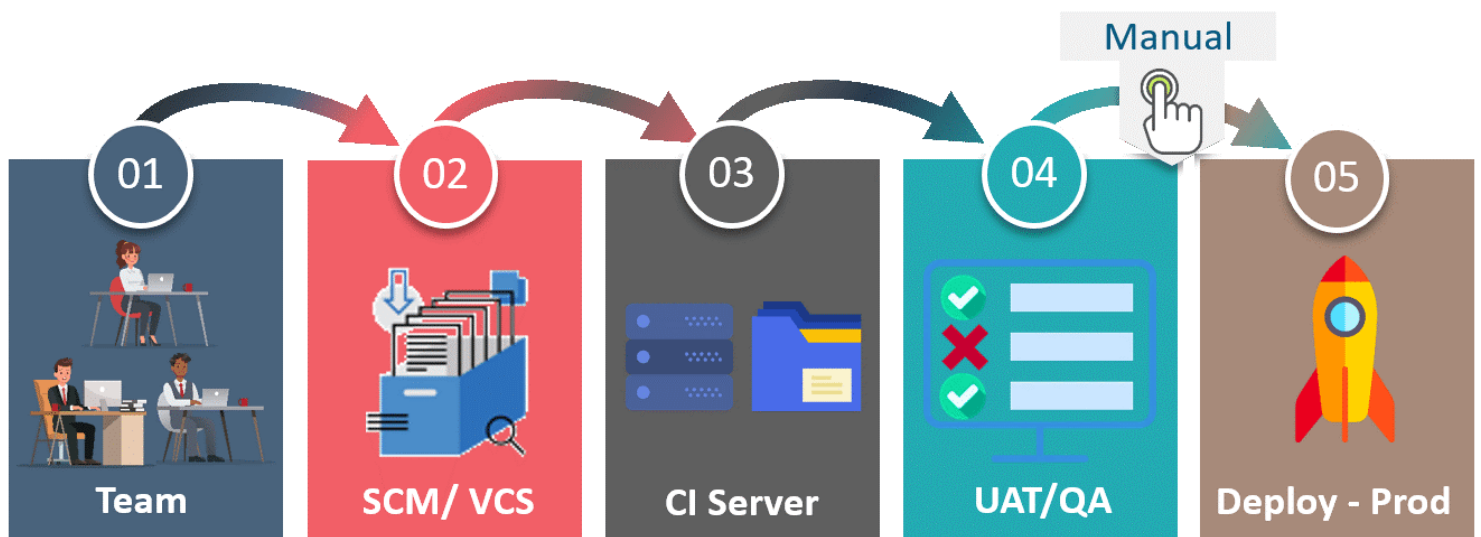
- **Organization Folders:** scan a **GitHub Organization** or **Bitbucket Team** to discover an organization's repositories, automatically creating managed Multibranch Pipeline jobs for them

Q52. What is the difference between Continuous Delivery and Continuous Deployment?

Continuous Delivery: (Manual Deployment to Production. Does not involve every change to be deployed.)



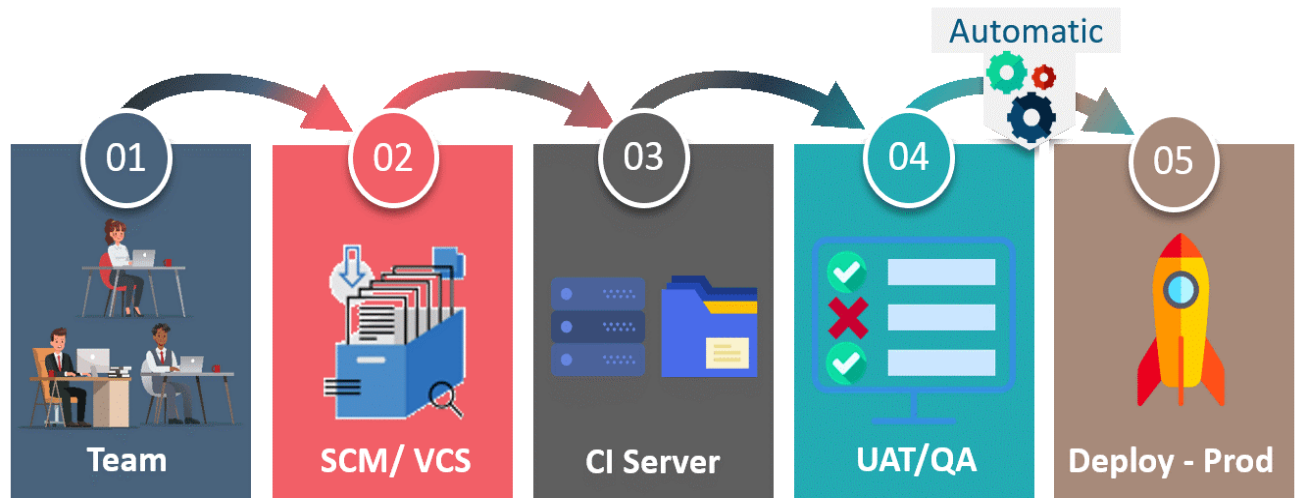
Continuous Delivery is a software development practice where you build software in such a way that the software can be released to the production at any time. You achieve Continuous Delivery by continuously integrating the products built by the development team, running automated tests on those built products to detect problems and then push those files into production-like



environments to ensure that the software works in production.

Continuous Deployment: (Automated Deployment to Production. Involves deploying every change automatically)

Continuous deployment means that every change that you make, goes through the pipeline, and if it passes all the tests, it automatically gets deployed into production. So, with this approach, the quality of the software release completely depends on the quality of the test suite as you have



automated everything.

--Docker--

1. What is Hypervisor?

A hypervisor is a software that makes virtualization possible. It is also called Virtual Machine Monitor. It divides the host system and allocates the resources to each divided virtual environment. You can basically have multiple OS on a single host system. There are two types of Hypervisors:

- Type 1: It's also called Native Hypervisor or Bare metal Hypervisor. It runs directly on the underlying host system. It has direct access to your host's system hardware and hence does not require a base server operating system.
- Type 2: This kind of hypervisor makes use of the underlying host operating system. It's also called Hosted Hypervisor.

2. What is virtualization?

Virtualization is the process of creating a software-based, virtual version of something (compute storage, servers, application, etc.). These virtual versions or environments are created from a single physical hardware system. Virtualization lets you split one system into many different sections which act like separate, distinct individual systems. A software called Hypervisor makes this kind of splitting possible. The virtual environment created by the hypervisor is called Virtual Machine.

3. What is containerization?

Let me explain this with an example. Usually, in the software development process, code developed on one machine might not work perfectly fine on any other machine because of the dependencies. This problem was solved by the containerization concept. So basically, an application that is being developed and deployed is bundled and wrapped together with all its configuration files and dependencies. This bundle is called a container. Now when you wish to run the application on another system, the container is deployed which will give a bug-free environment as all the dependencies and libraries are wrapped together. Most famous containerization environments are Docker and Kubernetes.

4. Difference between virtualization and containerization

Once you've explained containerization and virtualization, the next expected question would be differences. The question could either be differences between virtualization and containerization or differences between virtual machines and containers. Either way, this is how you respond.

Containers provide an isolated environment for running the application. The entire user space is explicitly dedicated to the application. Any changes made inside the container is never reflected on the host or even other containers running on the same host. Containers are an abstraction of the application layer. Each container is a different application.

Whereas in Virtualization, hypervisors provide an entire virtual machine to the guest (including Kernel). Virtual machines are an abstraction of the hardware layer. Each VM is a physical machine.

5. What is Docker?

Since its a Docker interview, there will be an obvious question about what is Docker. Start with a small definition.

Docker is a containerization platform which packages your application and all its dependencies together in the form of containers so as to ensure that your application works seamlessly in any environment, be it development, test or production. Docker containers, wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries, etc. It wraps basically anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

6. What is a Docker Container?

Docker containers include the application and all of its dependencies. It shares the kernel with other containers, running as isolated processes in user space on the host operating system. Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud. Docker containers are basically runtime instances of Docker images.

7. What are Docker Images?

When you mention Docker images, your very next question will be “what are Docker images”.

Docker image is the source of Docker container. In other words, Docker images are used to create containers. When a user runs a Docker image, an instance of a container is created. These docker images can be deployed to any Docker environment.

8. What is Docker Hub?

Docker images create docker containers. There has to be a registry where these docker images live. This registry is Docker Hub. Users can pick up images from Docker Hub and use them to create customized images and containers. Currently, the [Docker Hub](https://hub.docker.com/) is the world's largest public repository of image containers.

9. Explain Docker Architecture?

Docker Architecture consists of a Docker Engine which is a client-server application with three major components:

1. A server which is a type of long-running program called a daemon process (the docker command).
2. A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
3. A command line interface (CLI) client (the docker command).
4. The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

10. What is a Dockerfile?

Let's start by giving a small explanation of Dockerfile and proceed by giving examples and commands to support your arguments.

Docker can build images automatically by reading the instructions from a file called Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build, users can create an automated build that executes several command-line instructions in succession.

The interviewer does not just expect definitions, hence explain how to use a Dockerfile which comes with experience. Have a look at [this](#) tutorial to understand how Dockerfile works.

11. Tell us something about Docker Compose.

Docker Compose is a YAML file which contains details about the services, networks, and volumes for setting up the Docker application. So, you can use Docker Compose to create separate containers, host them and get them to communicate with each other. Each container will expose a port for communicating with other containers.

12. What is Docker Swarm?

You are expected to have worked with Docker Swarm as it's an important concept of Docker.

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

13. What is a Docker Namespace?

A namespace is one of the Linux features and an important concept of containers. Namespace adds a layer of isolation in containers. Docker provides various namespaces in order to stay portable and not affect the underlying host system. Few namespace types supported by Docker – PID, Mount, IPC, User, Network

14. What is Docker Machine?

Docker machine is a tool that lets you install Docker Engine on virtual hosts. These hosts can now be managed using the docker-machine commands. Docker machine also lets you provision Docker Swarm Clusters.

Docker Basic Commands

Once you've aced the basic conceptual questions, the interviewer will increase the difficulty level. So let's move on to the next section of this Docker Interview Questions article. This section talks about the commands that are very common amongst docker users.

15. What is the lifecycle of a Docker Container?

Docker containers have the following lifecycle:

- Create a container
- Run the container
- Pause the container(optional)
- Un-pause the container(optional)
- Start the container
- Stop the container
- Restart the container
- Kill the container
- Destroy the container

16. How to check for Docker Client and Docker Server version?

The following command gives you information about Docker Client and Server versions:

```
$ docker version
```

17. How do you get the number of containers running, paused and stopped?

You can use the following command to get detailed information about the docker installed on your system.

```
$ docker info
```

You can get the number of containers running, paused, stopped, the number of images and a lot more.

18. If you vaguely remember the command and you'd like to confirm it, how will you get help on that particular command?

The following command is very useful as it gives you help on how to use a command, the syntax, etc.

```
$ docker --help
```

The above command lists all Docker commands. If you need help with one specific command, you can use the following syntax:

```
$ docker <command> --help
```

19. How to login into docker repository?

You can use the following command to login into hub.docker.com:

```
$ docker login
```

You'll be prompted for your username and password, insert those and congratulations, you're logged in.

20. If you wish to use a base image and make modifications or personalize it, how do you do that?

You pull an image from docker hub onto your local system

It's one simple command to pull an image from docker hub:

```
$ docker pull <image_name>
```

21. How do you create a docker container from an image?

Pull an image from docker repository with the above command and run it to create a container. Use the following command:

```
$ docker run -it -d <image_name>
```

Most probably the next question would be, what does the '-d' flag mean in the command?

-d means the container needs to start in the detached mode. Explain a little about the detach mode. Have a look at [this](#) blog to get a better understanding of different docker commands.

22. How do you list all the running containers?

The following command lists down all the running containers:

```
$ docker ps
```

23. Suppose you have 3 containers running and out of these, you wish to access one of them. How do you access a running container?

The following command lets us access a running container:

```
$ docker exec -it <container_id> bash
```

The exec command lets you get inside a container and work with it.

24. How to start, stop and kill a container?

The following command is used to start a docker container:

```
$ docker start <container_id>
```

and the following for stopping a running container:

```
$ docker stop <container_id>
```

kill a container with the following command:

```
$ docker kill <container_id>
```

25. Can you use a container, edit it, and update it? Also, how do you make it a new and store it on the local system?

Of course, you can use a container, edit it and update it. This sounds complicated but it's actually just one command.

```
$ docker commit <container_id> <username/imagename>
```

26. Once you've worked with an image, how do you push it to docker hub?

```
$ docker push <username/image name>
```

27. How to delete a stopped container?

Use the following command to delete a stopped container:

```
$ docker rm <container id>
```

28. How to delete an image from the local storage system?

The following command lets you delete an image from the local system:

```
$ docker rmi <image-id>
```

29. How to build a Dockerfile?

Once you've written a Dockerfile, you need to build it to create an image with those specifications. Use the following command to build a Dockerfile:

```
$ docker build <path to docker file>
```

The next question would be when do you use “.dockerfile_name” and when to use the entire path?

Use “.dockerfile_name” when the dockerfile exists in the same file directory and you use the entire path if it lives somewhere else.

30. Do you know why *docker system prune* is used? What does it do?

```
$ docker system prune
```

The above command is used to remove all the stopped containers, all the networks that are not used, all dangling images and all build caches. It's one of the most useful docker commands.

31. Will you lose your data, when a docker container exists?

No, you won't lose any data when Docker container exists. Any data that your application writes to the container gets preserved on the disk until you explicitly delete the container. The file system for the container persists even after the container halts.

32. How is Docker different from other containerization methods?

Docker containers are very easy to deploy in any cloud platform. It can get more applications running on the same hardware when compared to other technologies, it makes it easy for developers to quickly create, ready-to-run containerized applications and it makes managing and deploying applications much easier. You can even share containers with your applications.

If you have some more points to add you can do that but make sure the above explanation is there in your answer.

33. Can I use JSON instead of YAML for my compose file in Docker?

You can use JSON instead of YAML for your compose file, to use JSON file with compose, specify the JSON filename to use, for eg:

```
$ docker-compose -f docker-compose.json up
```

34. Where all do you think Docker is being used?

When asked such a question, respond by talking about applications of Docker. Docker is being used in the following areas:

- Simplifying configuration: Docker lets you put your environment and configuration into code and deploy it.
- Code Pipeline Management: There are different systems used for development and production. As the code travels from development to testing to production, it goes through a difference in the environment. Docker helps in maintaining the code pipeline consistency.
- Developer Productivity: Using Docker for development gives us two things – We're closer to production and development environment is built faster.
- Application Isolation: As containers are applications wrapped together with all dependencies, your apps are isolated. They can work by themselves on any hardware that supports Docker.
- Debugging Capabilities: Docker supports various debugging tools that are not specific to containers but work well with containers.
- Multi-tenancy: Docker lets you have multi-tenant applications avoiding redundancy in your codes and deployments.
- Rapid Deployment: Docker eliminates the need to boost an entire OS from scratch, reducing the deployment time.

35. How have you used Docker in your previous position?

Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used it with other tools like Puppet, Chef or Jenkins. If you have no past practical experience in Docker and instead have experience with other tools in a similar space, be honest and explain the same. In this case, it makes sense if you can compare other tools to Docker in terms of functionality.

36. How far do Docker containers scale? Are there any requirements for the same?

Large web deployments like Google and Twitter and platform providers such as Heroku and dot Cloud, all run on container technology. Containers can be scaled to hundreds of thousands or even millions of them running in parallel. Talking about requirements, containers require the memory and the OS at all the times and a way to use this memory efficiently when scaled.

37. Can you remove a paused container from Docker?

The answer is no. You cannot remove a paused container. The container has to be in the stopped state before it can be removed.

38. Can a container restart by itself?

No, it's not possible for a container to restart by itself. By default the flag `-restart` is set to false.

39. What platforms does docker run on?

This is a very straightforward question but can get tricky. Do some company research before going for the interview and find out how the company is using Docker. Make sure you mention the platform company is using in this answer.

Docker runs on various Linux administration:

- Ubuntu 12.04, 13.04 et al
- Fedora 19/20+
- RHEL 6.5+
- CentOS 6+
- Gentoo
- ArchLinux
- openSUSE 12.3+
- CRUX 3.0+

It can also be used in production with Cloud platforms with the following services:

- Amazon EC2
- Amazon ECS
- Google Compute Engine
- Microsoft Azure
- Rackspace

40. Is there a way to identify the status of a Docker container?

There are six possible states a container can be at any given point – Created, Running, Paused, Restarting, Exited, Dead.

Use the following command to check for docker state at any given point:

```
$ docker ps
```

The above command lists down only running containers by default. To look for all containers, use the following command:

```
$ docker ps -a
```

41. Is it better to directly remove the container using the rm command or stop the container followed by remove container?

Its always better to stop the container and then remove it using the remove command.

```
$ docker stop <container_id>  
$ docker rm -f <container_id>
```

Stopping the container and then removing it will allow sending SIG_HUP signal to recipients. This will ensure that all the containers have enough time to clean up their tasks. This method is considered a good practice, avoiding unwanted errors.

42. Will cloud overtake the use of Containerization?

Docker containers are gaining popularity but at the same time, Cloud services are giving a good fight. In my personal opinion, Docker will never be replaced by Cloud. Using cloud services with containerization will definitely hype the game. Organizations need to take their requirements and dependencies into consideration into the picture and decide what's best for them. Most of the companies have integrated Docker with the cloud. This way they can make the best out of both the technologies.

43. How many containers can run per host?

There can be as many containers as you wish per host. Docker does not put any restrictions on it. But you need to consider every container needs storage space, CPU and memory which the hardware needs to support. You also need to consider the application size. Containers are considered to be lightweight but very dependant on the host OS.

44. Is it a good practice to run stateful applications on Docker?

The concept behind stateful applications is that they store their data onto the local file system. You need to decide to move the application to another machine, retrieving data becomes painful. I honestly would not prefer running stateful applications on Docker.

45. Suppose you have an application that has many dependant services. Will docker compose wait for the current container to be ready to move to the running of the next service?

The answer is yes. Docker compose always runs in the dependency order. These dependencies are specifications like depends_on, links, volumes_from, etc.

46. How will you monitor Docker in production?

Docker provides functionalities like docker stats and docker events to monitor docker in production. Docker stats provides CPU and memory usage of the container. Docker events provide information about the activities taking place in the docker daemon.

47. Is it a good practice to run Docker compose in production?

Yes, using docker compose in production is the best practical application of docker compose. When you define applications with compose, you can use this compose definition in various production stages like CI, staging, testing, etc.

48. What changes are expected in your docker compose file while moving it to production?

These are the following changes you need make to your compose file before migrating your application to the production environment:

- Remove volume bindings, so the code stays inside the container and cannot be changed from outside the container.
- Binding to different ports on the host.
- Specify a restart policy
- Add extra services like log aggregator

49. Have you used Kubernetes? If you have, which one would you prefer amongst Docker and Kubernetes?

Be very honest in such questions. If you have used Kubernetes, talk about your experience with Kubernetes and Docker Swarm. Point out the key areas where you thought docker swarm was more efficient and vice versa. Have a look at [this](#) blog for understanding differences between Docker and Kubernetes.

You Docker interview questions are not just limited to the workarounds of docker but also other similar tools. Hence be prepared with tools/technologies that give Docker competition. One such example is Kubernetes.

50. Are you aware of load balancing across containers and hosts? How does it work?

While using docker service with multiple containers across different hosts, you come across the need to load balance the incoming traffic. Load balancing and HAProxy is basically used to balance the incoming traffic across different available(healthy) containers. If one container crashes, another container should automatically start running and the traffic should be re-routed to this new running container. Load balancing and HAProxy works around this concept.

--Kubernetes--

Q1. How is Kubernetes different from Docker Swarm?

Features	Kubernetes	Docker Swarm
Installation & Cluster Config	Setup is very complicated, but once installed cluster is robust.	Installation is very simple, but the cluster is not robust.
GUI	GUI is the Kubernetes Dashboard .	There is no GUI.
Scalability	Highly scalable and scales fast.	Highly scalable and scales 5x faster than Kubernetes.
Auto-scaling	Kubernetes can do auto-scaling.	Docker swarm cannot do auto-scaling.
Load Balancing	Manual intervention needed for load balancing traffic between different containers and pods.	Docker swarm does auto load balancing of traffic between containers in the cluster.
Rolling Updates & Rollbacks	Can deploy rolling updates and does automatic rollbacks.	Can deploy rolling updates, but not automatic rollback.
DATA Volumes	Can share storage volumes only with the other containers in the same pod.	Can share storage volumes with any other container.
Logging & Monitoring	In-built tools for logging and monitoring.	3rd party tools like ELK stack should be used for logging and monitoring.

Q2. What is Kubernetes?

Fig 1: What is Kubernetes – Kubernetes Interview



Questions

Kubernetes is an open-source container management tool that holds the responsibilities of container deployment, scaling & descaling of containers & load balancing. Being Google's brainchild, it offers excellent community and works brilliantly with all the cloud providers. So, we can say that Kubernetes is not a *containerization platform*, but it is a *multi-container management solution*.

Q3. How is Kubernetes related to Docker?

It's a known fact that Docker provides the lifecycle management of containers and a Docker image builds the runtime containers. But, since these individual containers have to communicate, Kubernetes is used. So, Docker builds the containers and these containers communicate with each other via Kubernetes. So, containers running on multiple hosts can be manually linked and orchestrated using Kubernetes.

Q4. What is the difference between deploying applications on hosts and containers?

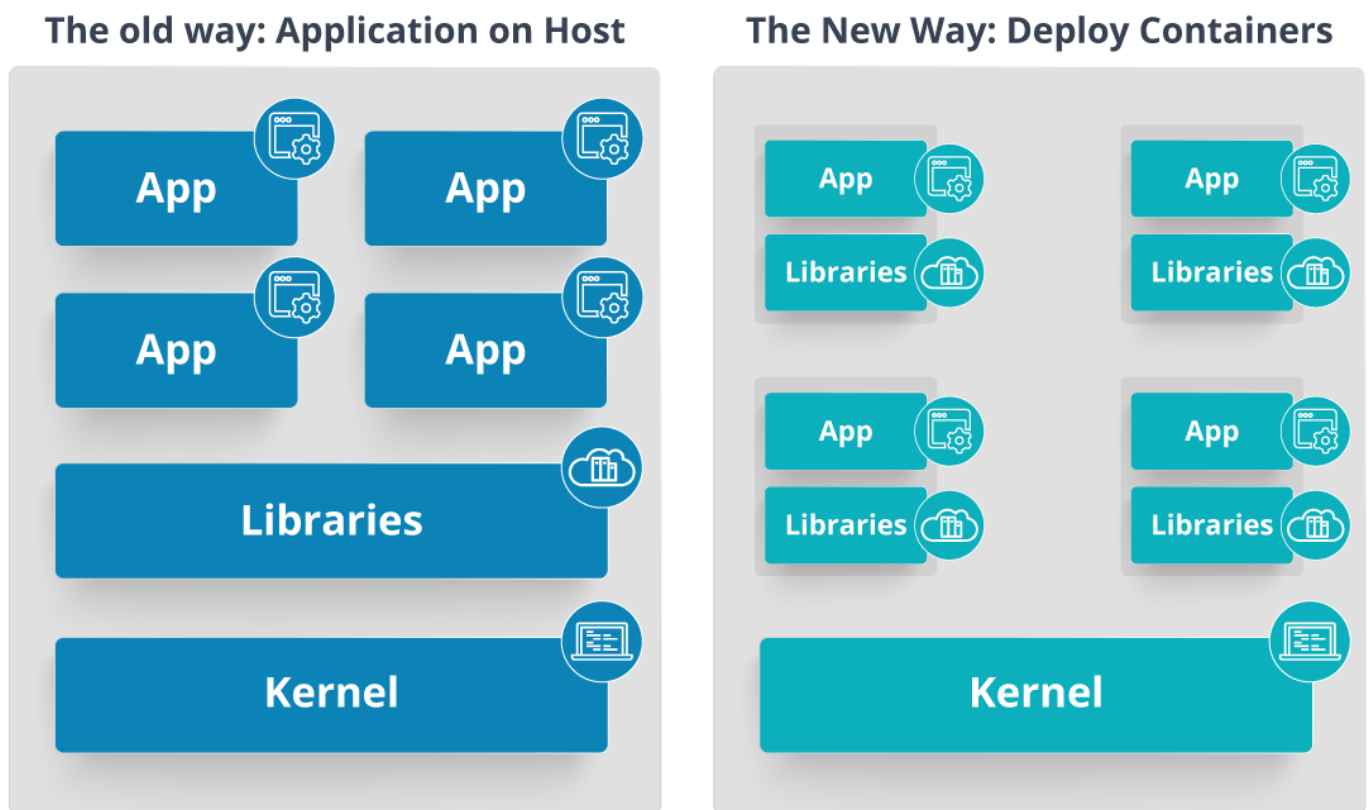


Fig 2: Deploying Applications On Host vs Containers – Kubernetes Interview Questions

Refer to the above diagram. The left side architecture represents deploying applications on hosts. So, this kind of architecture will have an operating system and then the operating system will have a kernel that will have various libraries installed on the operating system needed for the application. So, in this kind of framework you can have a number of applications and all the applications will share the libraries present in that

operating system whereas while deploying applications in containers the architecture is a little different.

This kind of architecture will have a kernel and that is the only thing that's going to be the only thing common between all the applications. So, if there's a particular application that needs Java then that particular application we'll get access to Java and if there's another application that needs Python then only that particular application will have access to Python.

The individual blocks that you can see on the right side of the diagram are basically containerized and these are isolated from other applications. So, the applications have the necessary libraries and binaries isolated from the rest of the system, and cannot be encroached by any other application.

Q5. What is Container Orchestration?

Consider a scenario where you have 5-6 microservices for an application. Now, these microservices are put in individual containers, but won't be able to communicate without container orchestration. So, as orchestration means the amalgamation of all instruments playing together in harmony in music, similarly container orchestration means all the services in individual containers working together to fulfill the needs of a single server.

Q6. What is the need for Container Orchestration?

Consider you have 5-6 microservices for a single application performing various tasks, and all these microservices are put inside containers. Now, to make sure that these containers communicate with each other we need container orchestration.

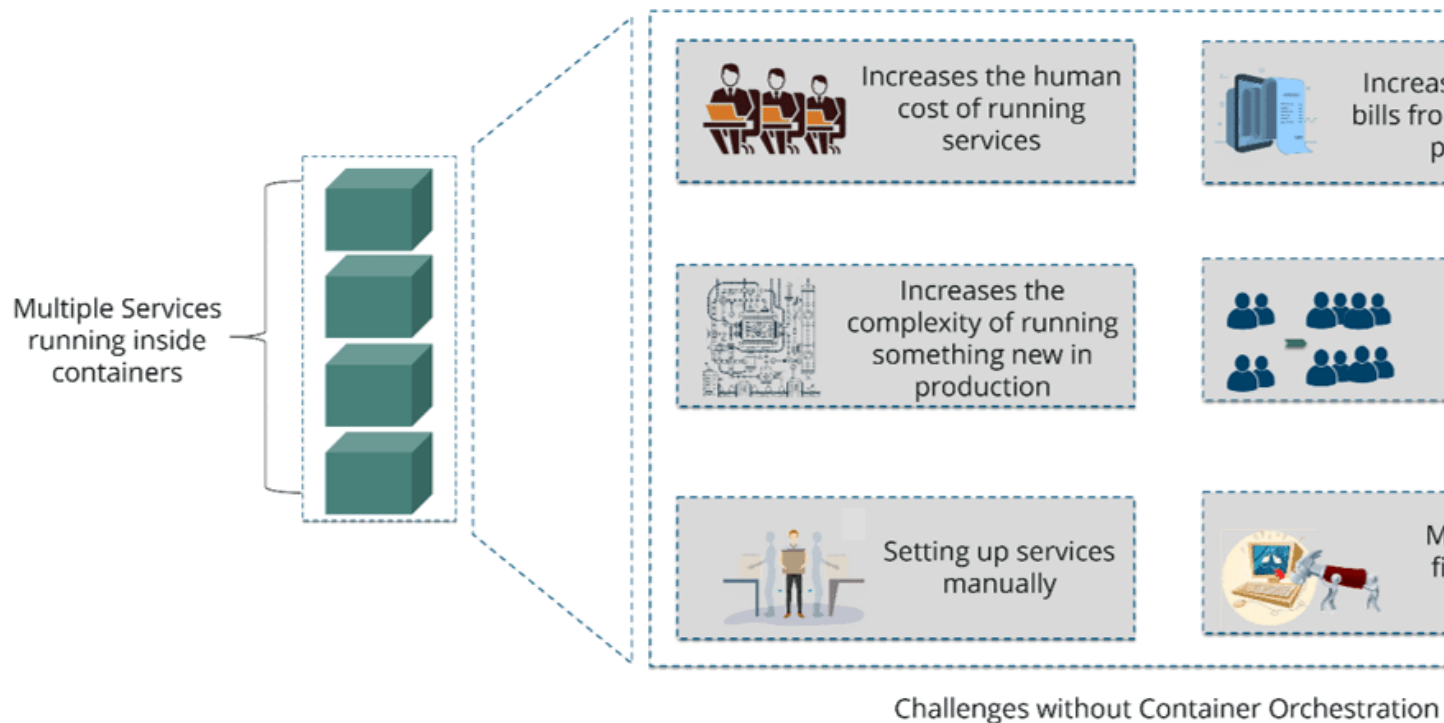


Fig 3: Challenges Without Container Orchestration – Kubernetes Interview Questions

As you can see in the above diagram, there were also many challenges that came into place without the use of container orchestration. So, to overcome these challenges the container orchestration came into place.

Q7. What are the features of Kubernetes?

The features of Kubernetes, are as follows:

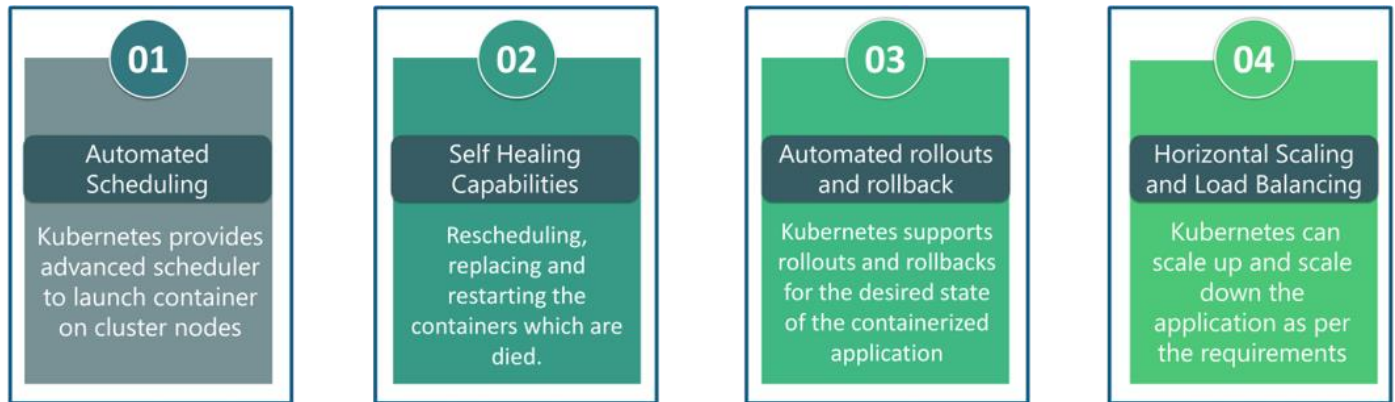


Fig 4: Features Of Kubernetes – Kubernetes Interview Questions

Q8. How does Kubernetes simplify containerized Deployment?

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

Q9. What do you know about clusters in Kubernetes?

The fundamental behind Kubernetes is that we can enforce the desired state management, by which I mean that we can feed the cluster services of a specific configuration, and it will be up to the cluster services to go out and run that configuration in the infrastructure.

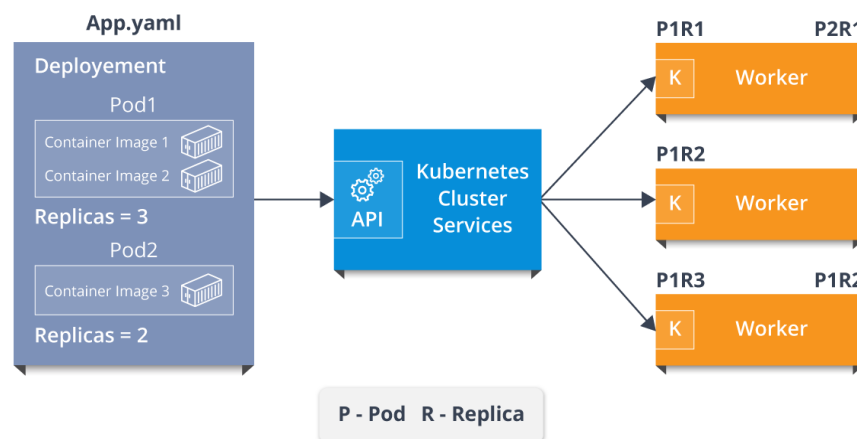


Fig 5: Representation Of Kubernetes Cluster – Kubernetes Interview Questions

So, as you can see in the above diagram, the deployment file will have all the configurations required to be fed into the cluster services. Now, the deployment file will be fed to the API and then it will be up to the cluster services to figure out how to schedule these pods in the environment and make sure that the right number of pods are running.

So, the API which sits in front of services, the worker nodes & the Kubelet process that the nodes run, all together make up the Kubernetes Cluster.

Q10. What is Google Container Engine?

Google Container Engine (GKE) is an open-source management platform for Docker containers and clusters. This Kubernetes based engine supports only those clusters which run within Google's public cloud services.

Q11. What is Heapster?

Heapster is a cluster-wide aggregator of data provided by Kubelet running on each node. This container management tool is supported natively on Kubernetes cluster and runs as a pod, just like any other pod in the cluster. So, it basically discovers all nodes in the cluster and queries usage information from the Kubernetes nodes in the cluster, via on-machine Kubernetes agent.

Q12. What is Minikube?

Minikube is a tool that makes it easy to run Kubernetes locally. This runs a single-node Kubernetes cluster inside a virtual machine.

Q13. What is Kubectl?

Kubectl is the platform using which you can pass commands to the cluster. So, it basically provides the CLI to run commands against the Kubernetes cluster with various ways to create and manage the Kubernetes component.

Q14. What is Kubelet?

This is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running.

Q15. What do you understand by a node in Kubernetes?

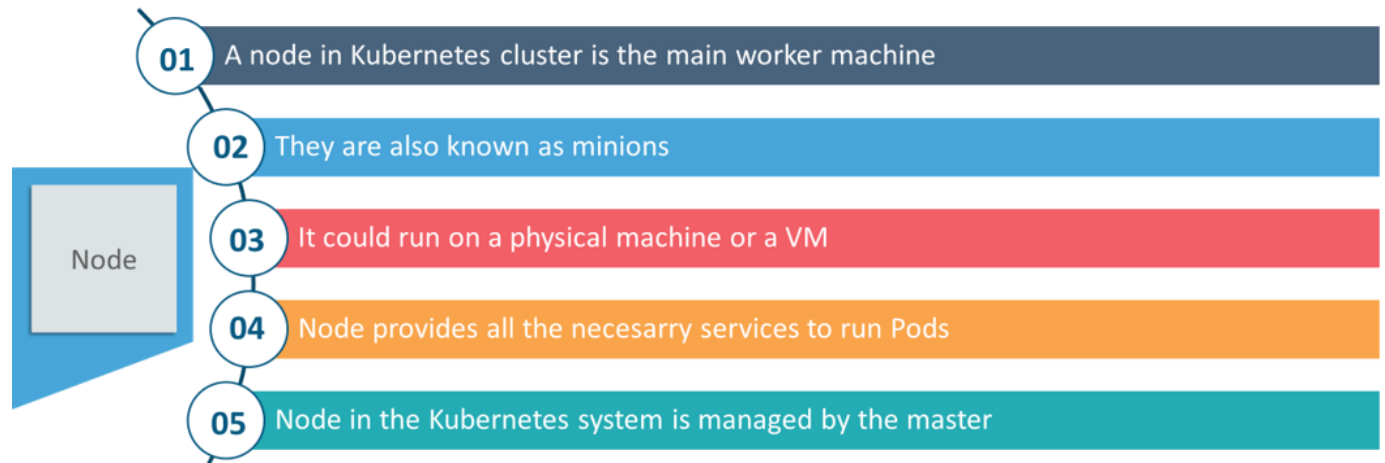


Fig 6: Node In Kubernetes – Kubernetes Interview Questions

Q1. What are the different components of Kubernetes Architecture?

The [Kubernetes Architecture](#) has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node.

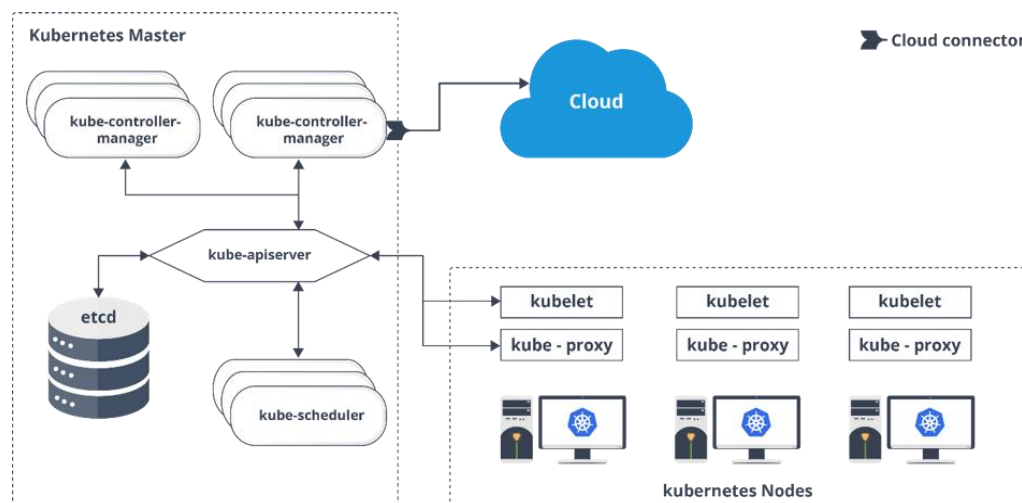


Fig 7: Architecture Of Kubernetes – Kubernetes Interview Questions

Q2. What do you understand by Kube-proxy?

Kube-proxy can run on each and every node and can do simple TCP/UDP packet forwarding across backend network service. So basically, it is a network proxy that reflects the services as configured in Kubernetes API on each node. So, the Docker-linkable compatible environment variables provide the cluster IPs and ports which are opened by proxy.

Q3. Can you brief on the working of the master node in Kubernetes?

Kubernetes master controls the nodes and inside the nodes the containers are present. Now, these individual containers are contained inside pods and inside each pod, you can have a various number of containers based upon the configuration and requirements. So, if the pods have to be deployed, then they can either be deployed using user interface or command-line interface. Then, these pods are scheduled on the nodes, and based on the resource requirements, the pods are allocated to these nodes. The kube-apiserver makes sure that there is communication established between the Kubernetes node and the master components.

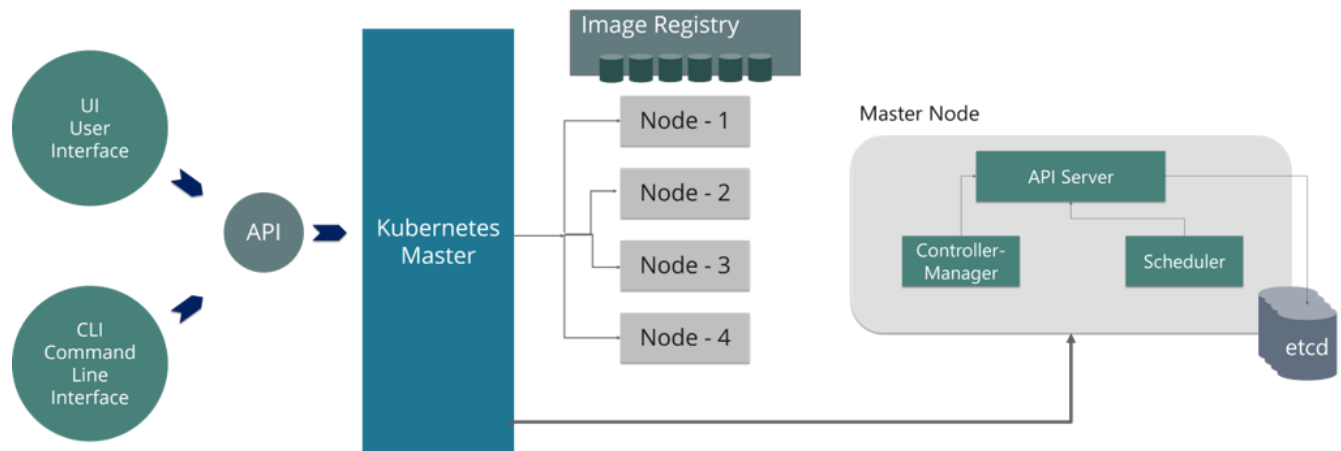


Fig 8: Representation Of Kubernetes Master Node – Kubernetes Interview Questions

Q4. What is the role of kube-apiserver and kube-scheduler?

The kube – apiserver follows the scale-out architecture and is the front end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.

The kube-scheduler is responsible for distributing and managing the workload on the worker nodes. So, it selects the most suitable node to run the unscheduled pod based on resource requirements and keeps track of resource utilization. It ensures that the workload is not scheduled on already full nodes.

Q5. Can you brief me about the Kubernetes controller manager?

Multiple controller processes run on the master node but are compiled together to run as a single process: the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.

So, the different types of controller manager running on the master node are :

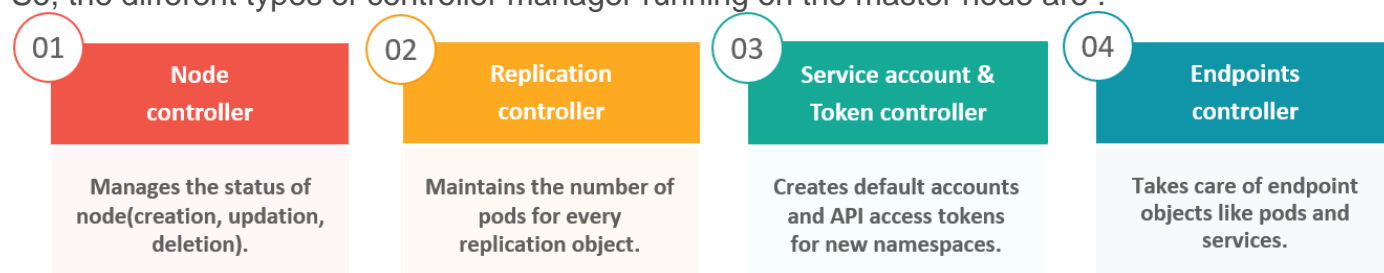


Fig 9: Types Of Controllers – Kubernetes Interview Questions

Q6. What is ETCD?

EtcD is written in [Go programming language](#) and is a distributed key-value store used for coordinating distributed work. So, EtcD stores the configuration data of the Kubernetes cluster, representing the state of the cluster at any given point in time.

Q7. What are the different types of services in Kubernetes?

The following are the different types of services used:

Cluster IP	Node Port	Load Balancer	External Name
<ul style="list-style-type: none">Exposes the service on a cluster-internal IP.Makes the service only reachable from within the cluster.This is the default Service Type.	<ul style="list-style-type: none">Exposes the service on each Node's IP at a static port.A Cluster IP service to which Node Port service will route, is automatically created.	<ul style="list-style-type: none">Exposes the service externally using a cloud provider's load balancer.Services, to which the external load balancer will route, are automatically created.	<ul style="list-style-type: none">Maps the service to the contents of the External Name field by returning a CNAME record with its value.No proxying of any kind is set up.

Fig 10: Types Of Services – Kubernetes Interview Questions

Q8. What do you understand by load balancer in Kubernetes?

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

Q9. What is Ingress network, and how does it work?

Ingress network is a collection of rules that acts as an entry point to the Kubernetes cluster. This allows inbound connections, which can be configured to give services externally through reachable URLs, load balance traffic, or by offering name-based virtual hosting. So, Ingress is an API object that manages external access to the services in a cluster, usually by HTTP and is the most powerful way of exposing service.

Now, let me explain to you the working of Ingress network with an example.

There are 2 nodes having the pod and root network namespaces with a Linux bridge. In addition to this, there is also a new virtual ethernet device called flannel0(network plugin) added to the root network.

Now, suppose we want the packet to flow from pod1 to pod 4. Refer to the below diagram.

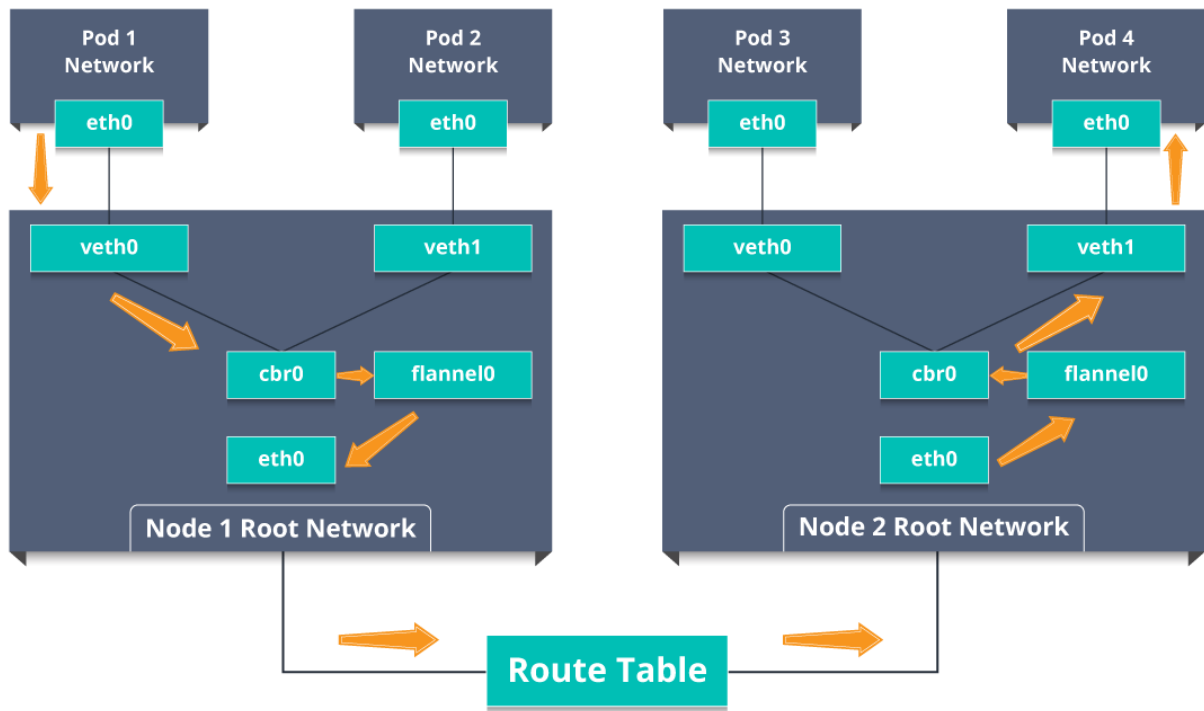


Fig 11: Working Of Ingress Network – Kubernetes Interview Questions

- So, the packet leaves pod1's network at eth0 and enters the root network at veth0.
- Then it is passed on to cbr0, which makes the ARP request to find the destination and it is found out that nobody on this node has the destination IP address.
- So, the bridge sends the packet to flannel0 as the node's route table is configured with flannel0.
- Now, the flannel daemon talks to the API server of Kubernetes to know all the pod IPs and their respective nodes to create mappings for pods IPs to node IPs.
- The network plugin wraps this packet in a UDP packet with extra headers changing the source and destination IP's to their respective nodes and sends this packet out via eth0.
- Now, since the route table already knows how to route traffic between nodes, it sends the packet to the destination node2.
- The packet arrives at eth0 of node2 and goes back to flannel0 to de-capsulate and emits it back in the root network namespace.

- Again, the packet is forwarded to the Linux bridge to make an ARP request to find out the IP that belongs to veth1.
- The packet finally crosses the root network and reaches the destination Pod4.

Q10. What do you understand by Cloud controller manager?

The Cloud Controller Manager is responsible for persistent storage, network routing, abstracting the cloud-specific code from the core Kubernetes specific code, and managing the communication with the underlying cloud services. It might be split out into several different containers depending on which cloud platform you are running on and then it enables the cloud vendors and Kubernetes code to be developed without any inter-dependency. So, the cloud vendor develops their code and connects with the Kubernetes cloud-controller-manager while running the Kubernetes.

The various types of cloud controller manager are as follows:

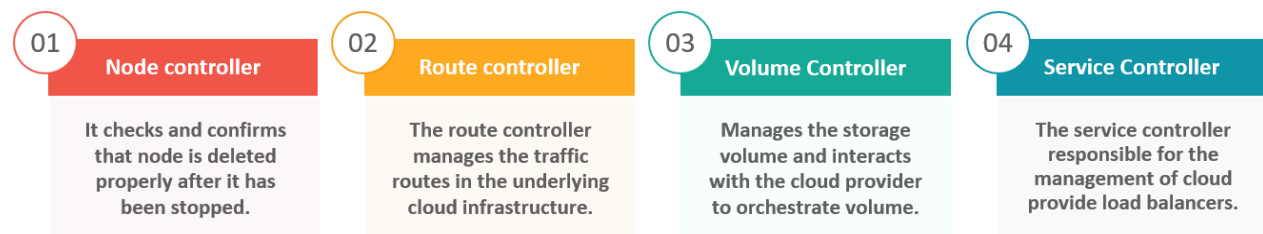


Fig 12: Types Of Cloud Controller Manager – Kubernetes Interview Questions

Q11. What is Container resource monitoring?

As for users, it is really important to understand the performance of the application and resource utilization at all the different abstraction layer, Kubernetes factored the management of the cluster by creating abstraction at different levels like container, pods, services and whole cluster. Now, each level can be monitored and this is nothing but Container resource monitoring.

The various container resource monitoring tools are as follows:

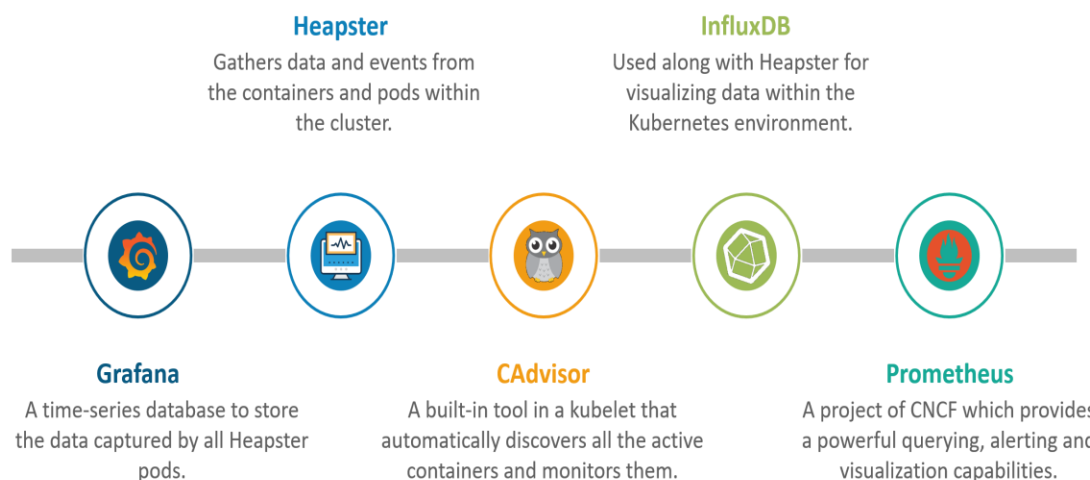


Fig 13: Container Resource Monitoring Tools – Kubernetes Interview Questions

Q12. What is the difference between a replica set and a replication controller?

Replica Set and Replication Controller do almost the same thing. Both ensure that a specified number of pod replicas are running at any given time. The difference comes with the usage of selectors to replicate pods. Replica Set uses Set-Based selectors while replication controllers use Equity-Based selectors.

- **Equity-Based Selectors:** This type of selector allows filtering by label key and values. So, in layman's terms, the equity-based selector will only look for the pods with the exact same phrase as the label.
Example: Suppose your label key says `app=nginx`; then, with this selector, you can only look for those pods with label `app` equal to `nginx`.
- **Selector-Based Selectors:** This type of selector allows filtering keys according to a set of values. So, in other words, the selector-based selector will look for pods whose label has been mentioned in the set.
Example: Say your label key says `app` in (`Nginx`, `NPS`, `Apache`). Then, with this selector, if your `app` is equal to any of `Nginx`, `NPS`, or `Apache`, the selector will take it as a true result.

Q13. What is a Headless Service?

Headless Service is similar to that of a 'Normal' service but does not have a Cluster IP. This service enables you to directly reach the pods without the need to access them through a proxy.

Q14. What are the best security measures that you can take while using Kubernetes?

The following are the best security measures that you can follow while using Kubernetes:

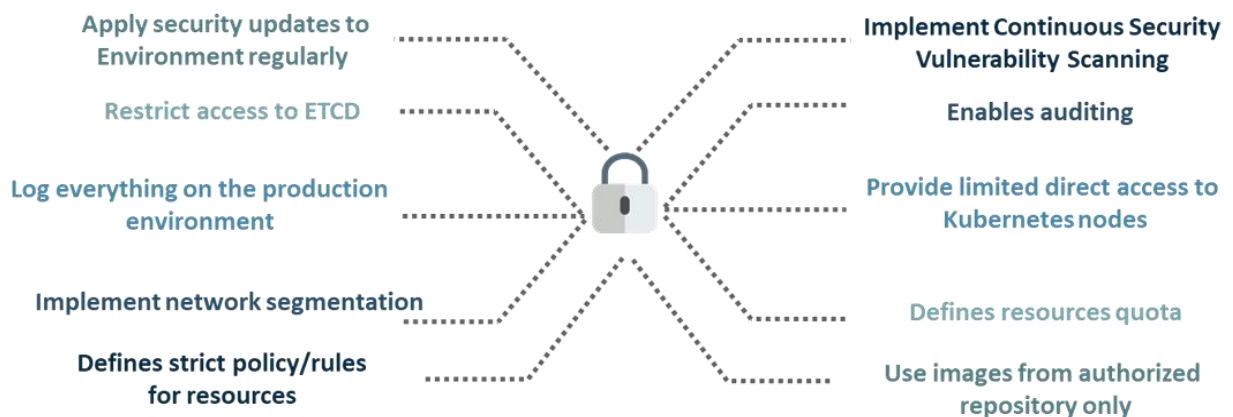


Fig 14: Best Security Measures – Kubernetes Interview Questions

Q15. What are federated clusters?

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data center/cloud and use federation to control/manage them all at one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.

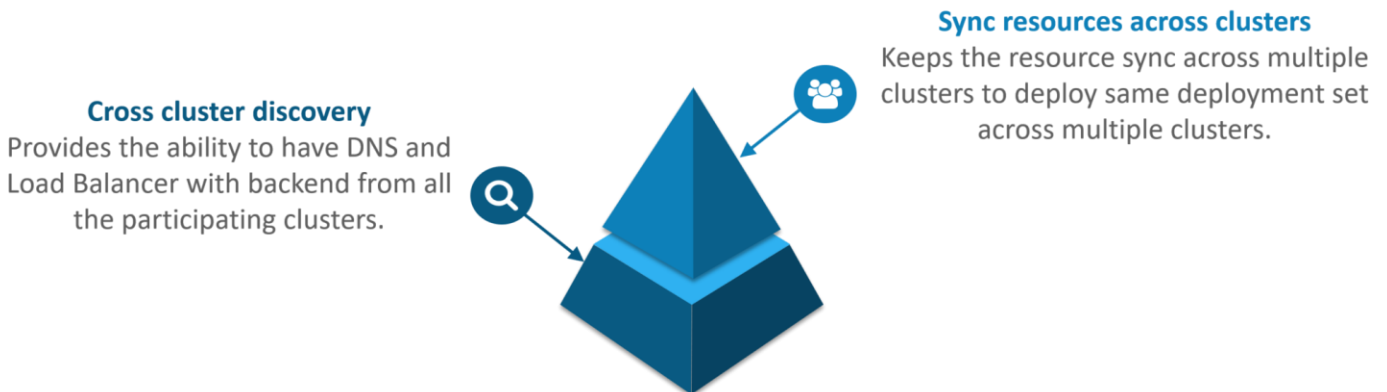


Fig 15: Federated Clusters – Kubernetes Interview Questions

Scenario-Based Interview Questions

This section of questions will consist of various scenario-based questions that you may face in your interviews.

Scenario 1: Suppose a company built on monolithic architecture handles numerous products. Now, as the company expands in today's scaling industry, their monolithic architecture started causing problems.

How do you think the company shifted from monolithic to microservices and deploy their services containers?

Solution:

As the company's goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes cluster.

Scenario 2: Consider a multinational company with a very much distributed system, with a large number of data centers, virtual machines, and many employees working on various tasks.

How do you think can such a company manage all the tasks in a consistent way with Kubernetes?

Solution:

As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system.

In such a situation the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications.

So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with an extensive support for various container networking solutions and container storage.

Scenario 3: Consider a situation, where a company wants to increase its efficiency and the speed of its technical operations by maintaining minimal costs.

How do you think the company will try to achieve this?

Solution:

The company can implement the DevOps methodology, by building a CI/CD pipeline, but one problem that may occur here is the configurations may take time to go up and running. So, after implementing the CI/CD pipeline the company's next step should be to work in the cloud environment. Once they start working on the cloud environment, they can schedule containers on a cluster and can orchestrate with the help of Kubernetes. This kind of approach will help the company reduce their deployment time, and also get faster across various environments.

Scenario 4: Suppose a company wants to revise its deployment methods and wants to build a platform which is much more scalable and responsive.

How do you think this company can achieve this to satisfy their customers?

Solution:

In order to give millions of clients the digital experience they would expect, the company needs a platform that is scalable, and responsive, so that they could quickly get data to the client website. Now, to do this the company should move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform available i.e. Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.

Scenario 5: Consider a multinational company with a very much distributed system, looking forward to solving the monolithic code base problem.

How do you think the company can solve their problem?

Solution

Well, to solve the problem, they can shift their monolithic code base to a microservice design and then each and every microservices can be considered as a container. So, all these containers can be deployed and orchestrated with the help of Kubernetes.

Kubernetes Interview Questions

Scenario 6: All of us know that the shift from monolithic to microservices solves the problem from the development side, but increases the problem at the deployment side.

How can the company solve the problem on the deployment side?

Solution

The team can experiment with container orchestration platforms, such as Kubernetes and run it in data centers. So, with this, the company can generate a templated application, deploy it within five minutes, and have actual instances containerized in the staging environment at that point. This kind of Kubernetes project will have dozens of microservices running in parallel to improve the production rate as even if a node goes down, then it can be rescheduled immediately without performance impact.

Scenario 7: Suppose a company wants to optimize the distribution of its workloads, by adopting new technologies.

How can the company achieve this distribution of resources efficiently?

Solution

The solution to this problem is none other than Kubernetes. Kubernetes makes sure that the resources are optimized efficiently, and only those resources are used which are needed by that particular application. So, with the usage of the best container orchestration tool, the company can achieve the distribution of resources efficiently.

Scenario 8: Consider a carpooling company wants to increase their number of servers by simultaneously scaling their platform.

How do you think will the company deal with the servers and their installation?

Solution

The company can adopt the concept of containerization. Once they deploy all their application into containers, they can use Kubernetes for orchestration and use container monitoring tools like Prometheus to monitor the actions in containers. So, with such usage of containers, giving them better capacity planning in the data center because they will now have fewer constraints due to this abstraction between the services and the hardware they run on.

Scenario 9: Consider a scenario where a company wants to provide all the required hand-outs to its customers having various environments.

How do you think they can achieve this critical target in a dynamic manner?

Solution

The company can use Docker environments, to put together a cross-sectional team to build a web application using Kubernetes. This kind of framework will help the company achieve the goal of getting the required things into production within the shortest time frame. So, with such a machine running, the company can give the hands-outs to all the customers having various environments.

Scenario 10: Suppose a company wants to run various workloads on different cloud infrastructure from bare metal to a public cloud.

How will the company achieve this in the presence of different interfaces?

Solution

The company can decompose its infrastructure into microservices and then adopt Kubernetes. This will let the company run various workloads on different cloud infrastructures.

Multiple Choice Interview Questions

This section of questions will consist of multiple-choice interview questions, that are frequently asked in interviews.

Q1. What are minions in the Kubernetes cluster?

- a. They are components of the master node.
- b. They are the work-horse / worker node of the cluster.[Ans]
- c. They are monitoring engine used widely in kubernetes.
- d. They are docker container service.

Q2. Kubernetes cluster data is stored in which of the following?

- a. Kube-apiserver
- b. Kubelet
- c. Etcd[Ans]
- d. None of the above

Q3. Which of them is a Kubernetes Controller?

- a. ReplicaSet
- b. Deployment
- c. Rolling Updates
- d. Both ReplicaSet and Deployment[Ans]

Q4. Which of the following are core Kubernetes objects?

- a. Pods
- b. Services
- c. Volumes
- d. All of the above[Ans]

Q5. The Kubernetes Network proxy runs on which node?

- a. Master Node
- b. Worker Node
- c. All the nodes[Ans]
- d. None of the above

Q6. What are the responsibilities of a node controller?

- a. To assign a CIDR block to the nodes
- b. To maintain the list of nodes
- c. To monitor the health of the nodes
- d. All of the above[Ans]

Q7. What are the responsibilities of Replication Controller?

- a. Update or delete multiple pods with a single command
- b. Helps to achieve the desired state
- c. Creates a new pod, if the existing pod crashes
- d. All of the above[Ans]

Q8. How to define a service without a selector?

- a. Specify the external name[Ans]
- b. Specify an endpoint with IP Address and port
- c. Just by specifying the IP address
- d. Specifying the label and api-version

Q9. What did the 1.8 version of Kubernetes introduce?

- a. Taints and Tolerations[Ans]
- b. Cluster level Logging
- c. Secrets
- d. Federated Clusters

Q10. The handler invoked by Kubelet to check if a container's IP address is open or not is?

- a. HTTPGetAction
- b. ExecAction
- c. TCPSocketAction[Ans]
- d. None of the above

--DevOps—

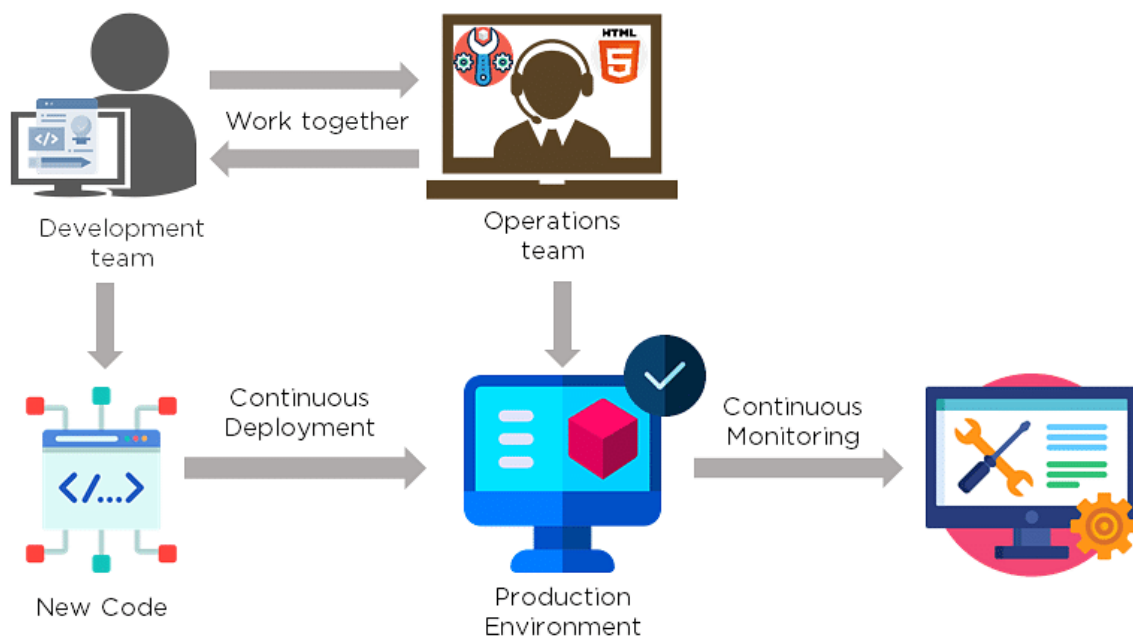
Basic

1. What do you know about DevOps?

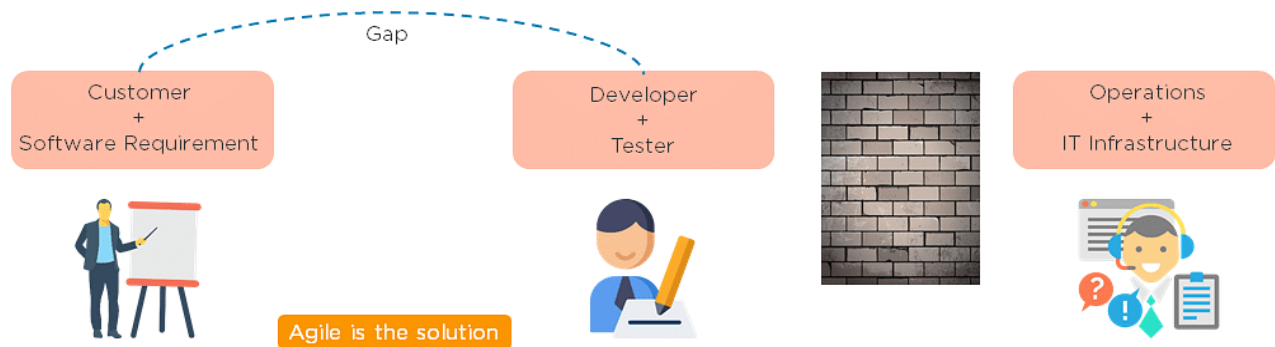
Your answer must be simple and straightforward. Begin by explaining the growing importance of DevOps in the [IT industry](#). Discuss how such an approach aims to synergize the efforts of the development and operations teams to accelerate the delivery of software products, with a minimal failure rate. Include how DevOps is a value-added practice, where development and operations engineers join hands throughout the product or service lifecycle, right from the design stage to the point of deployment.

2. How is DevOps different from agile methodology?

[DevOps is a culture](#) that allows the development and the operations team to work together. This results in [continuous development](#), testing, integration, deployment, and monitoring of the software throughout the lifecycle.



Agile is a [software development methodology](#) that focuses on iterative, incremental, small, and rapid releases of software, along with customer feedback. It addresses gaps and conflicts between the customer and developers.



DevOps addresses gaps and conflicts between the Developers and IT Operations.



3. Which are some of the most popular DevOps tools?

The most popular [DevOps tools](#) include:

1. [Selenium](#)
2. [Puppet](#)
3. [Chef](#)
4. [Git](#)
5. [Jenkins](#)

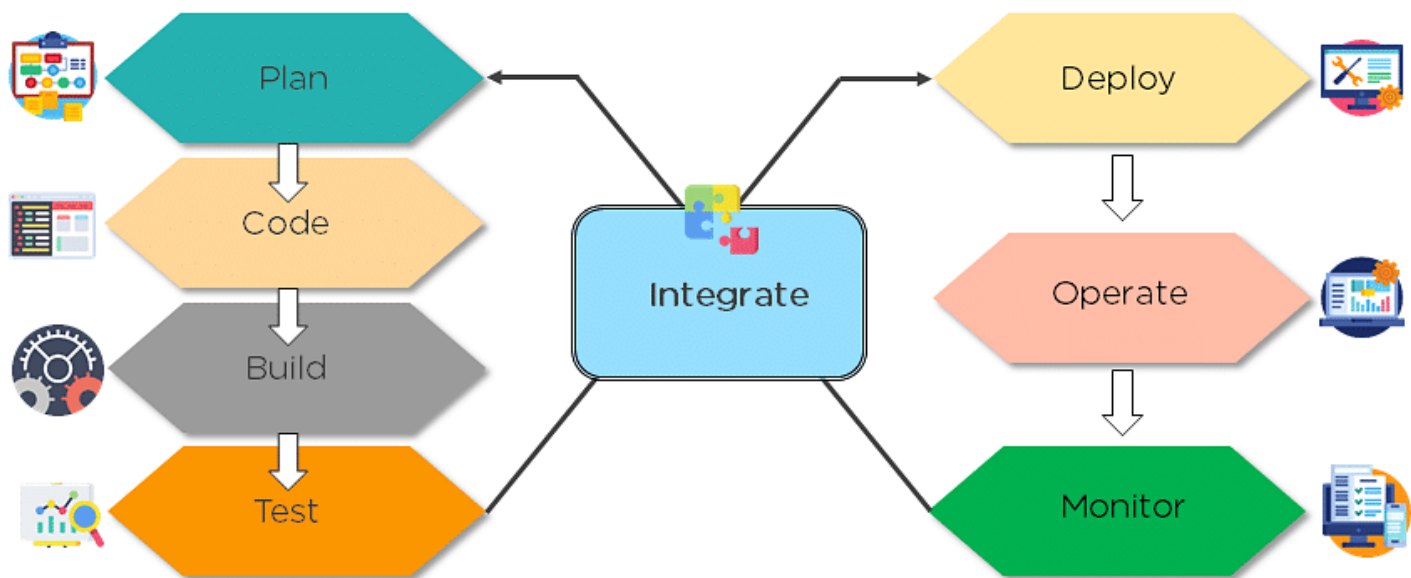
6. [Ansible](#)

7. [Docker](#)

4. What are the different phases in DevOps?

The various phases of the DevOps lifecycle are as follows:

- Plan - Initially, there should be a plan for the type of application that needs to be developed. Getting a rough picture of the development process is always a good idea.
- Code - The application is coded as per the end-user requirements.
- Build - Build the application by integrating various codes formed in the previous steps.
- Test - This is the most crucial step of the application development. Test the application and rebuild, if necessary.
- Integrate - Multiple codes from different programmers are integrated into one.
- Deploy - Code is deployed into a cloud environment for further usage. It is ensured that any new changes do not affect the functioning of a high traffic website.
- Operate - Operations are performed on the code if required.
- Monitor - Application performance is monitored. Changes are made to meet the end-user requirements.



The above figure indicates the DevOps lifecycle.

5. Mention some of the core benefits of DevOps.

The core benefits of DevOps are as follows:

Technical benefits

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

Business benefits

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

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

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

Stage 1

An assessment of the existing process and implementation for about two to three weeks to identify areas of improvement so that the team can create a road map for the implementation.

Stage 2

Create a proof of concept (PoC). Once it is accepted and approved, the team can start on the actual implementation and roll-out of the project plan.

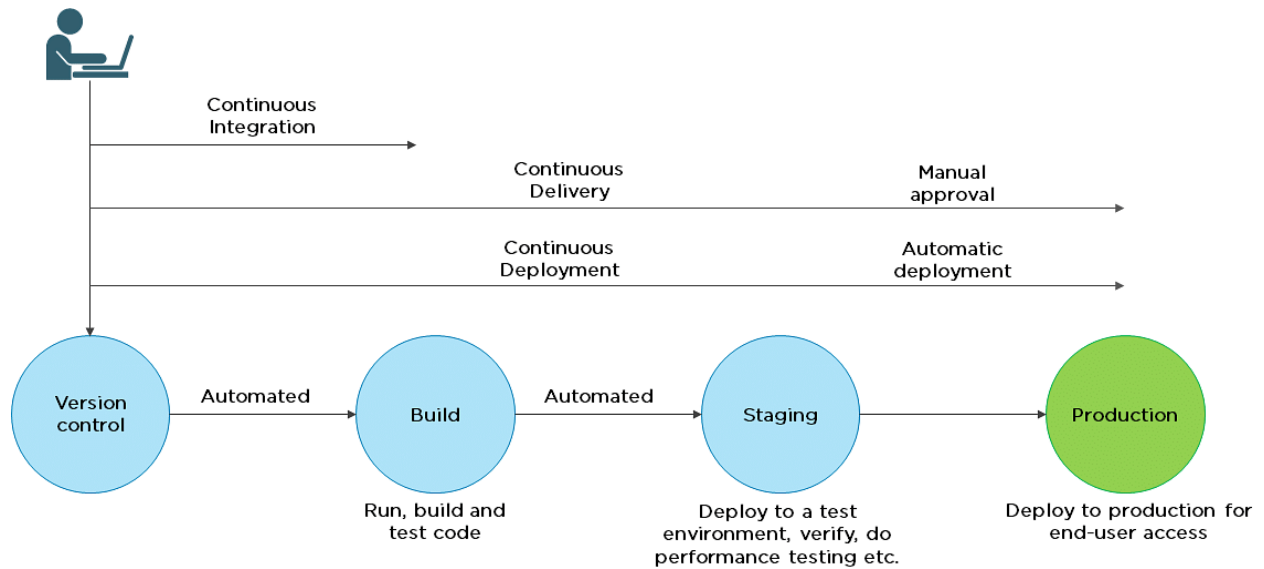
Stage 3

The project is now ready for implementing DevOps by using version control/integration/testing/deployment/delivery and monitoring followed step by step.

By following the proper steps for [version control](#), integration, testing, deployment, delivery, and monitoring, the project is now ready for DevOps implementation.

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

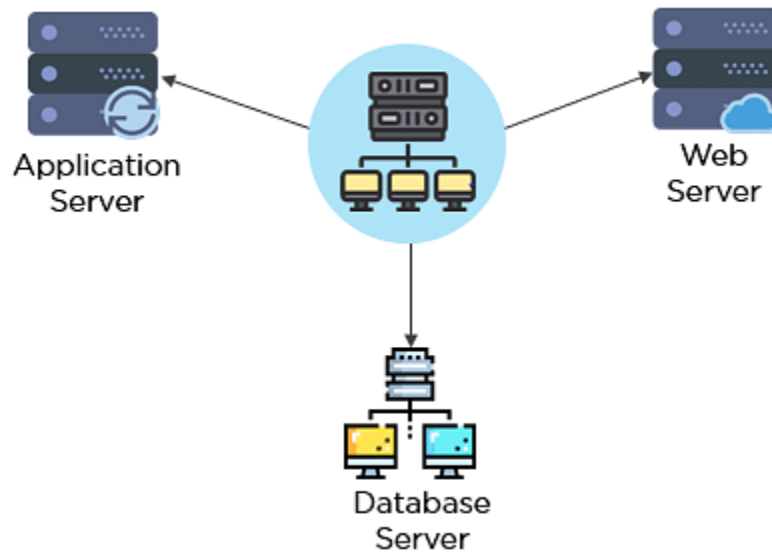
Continuous Delivery	Continuous Deployment
Ensures code can be safely deployed on to production	Every change that passes the automated tests is deployed to production automatically
Ensures business applications and services function as expected	Makes software development and the release process faster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



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

- Enables management of and changes to multiple systems.
- Standardizes resource configurations, which in turn, manage IT infrastructure.
- It helps with the administration and management of multiple servers and maintains the integrity of the entire infrastructure.

9. How does continuous monitoring help you maintain the entire architecture of the system?



Continuous monitoring in DevOps is a process of detecting, identifying, and reporting any faults or threats in the entire infrastructure of the system.

- Ensures that all services, applications, and resources are running on the servers properly.
- Monitors the status of servers and determines if applications are working correctly or not.
- Enables continuous audit, transaction inspection, and controlled monitoring.

10. What is the role of AWS in DevOps?

- Flexible services - Provides ready-to-use, flexible services without the need to install or set up the software.
- Built for scale - You can manage a single instance or scale to thousands using AWS services.
- Automation - AWS lets you automate tasks and processes, giving you more time to innovate
- Secure - Using AWS Identity and Access Management (IAM), you can set user permissions and policies.

- Large partner ecosystem - AWS supports a large ecosystem of partners that integrate with and extend AWS services.

11. Name three important DevOps KPIs.

The three important KPIs are as follows:

- Meantime to failure recovery - This is the average time taken to recover from a failure.
- Deployment frequency - The frequency in which the deployment occurs.
- Percentage of failed deployments - The number of times the deployment fails.

12. Explain the term "Infrastructure as Code" (IaC) as it relates to configuration management.

- Writing code to manage configuration, deployment, and automatic provisioning.
- Managing data centers with machine-readable definition files, rather than physical hardware configuration.
- Ensuring all your servers and other infrastructure components are provisioned consistently and effortlessly.
- Administering cloud computing environments, also known as [infrastructure as a service \(IaaS\)](#).

13. How is IaC implemented using AWS?

Start by talking about the age-old mechanisms of writing commands onto script files and testing them in a separate environment before deployment and how this approach is being replaced by IaC. Similar to the codes written for other services, with the help of AWS, IaC allows developers to write, test, and maintain infrastructure entities in a descriptive manner, using formats such as JSON or YAML. This enables easier development and faster deployment of infrastructure changes.

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

Before talking about the growing popularity of DevOps, discuss the current industry scenario. Begin with some examples of how big players such as [Netflix and Facebook](#) are investing in DevOps to automate and accelerate application deployment and how this has helped them grow their business. Using Facebook as an example, you would point to Facebook's continuous deployment and code ownership models and how these have helped it scale up but ensure the quality of experience at the same time. Hundreds of lines of code are implemented without affecting quality, stability, and security.

Your next use case should be Netflix. This streaming and on-demand video company follow similar practices with fully automated processes and systems. Mention the user base of these two organizations: Facebook has 2 billion users while Netflix streams online content to more than 100 million users worldwide.

These are great examples of how DevOps can help organizations to ensure higher success rates for releases, reduce the lead time between bug fixes, streamline and continuous delivery through automation, and an overall reduction in manpower costs.

We will now look into the next set of DevOps Interview Questions that includes - Git, Selenium, Jenkins.

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

The main differences between Agile and DevOps are summarized below:

Characteristics	Agile	DevOps
Work Scope	Only Agility	Automation needed along with Agility

Focus Area	Main priority is Time and deadlines	Quality and Time management are of equal priority
Feedback Source	The main source of feedback - customers	The main source of feedback - self (tools used for monitoring)
Practices or Processes	Practices like Agile Kanban, Scrum, etc., are followed.	Processes and practices like Continuous Development (CD),
Development Sprints or Release cycles	Release cycles are usually smaller.	Release cycles are smaller, along with immediate feedback.
Agility	Only development agility is present.	Both in operations and development, agility is followed.
followed		Continuous Integration (CI), etc., are followed.

16. Can you explain the “Shift left to reduce failure” concept in DevOps?

Shift left is a DevOps idea for improving security, performance, and other factors. Let us take an example: if we look at all of the processes in DevOps, we can state that security is tested prior to the deployment step. We can add security in the development phase, which is on the left, by employing the left shift method. [will be depicted in a diagram] We can integrate with all phases, including before development and during testing, not just development. This most likely raises the security level by detecting faults at an early stage.

17. What are the anti-patterns of DevOps?

Patterns are common practices that are usually followed by organizations. An anti-pattern is formed when an organization continues to blindly follow a pattern adopted by others but does not work for them. Some of the myths about DevOps include:

- Cannot perform DevOps → Have the wrong people

- DevOps ⇒ Production Management is done by developers
- The solution to all the organization's problems ⇒ DevOps
- DevOps == Process
- DevOps == Agile
- Cannot perform DevOps → Organization is unique
- A separate group needs to be made for DevOps

18. What are the benefits of using version control?

Here are the benefits of using Version Control:

- All team members are free to work on any file at any time with the Version Control System (VCS). Later on, VCS will allow the team to integrate all of the modifications into a single version.
- The VCS asks to provide a brief summary of what was changed every time we save a new version of the project. We also get to examine exactly what was modified in the content of the file. As a result, we will be able to see who made what changes to the project.
- Inside the VCS, all the previous variants and versions are properly stored. We will be able to request any version at any moment, and we will be able to retrieve a snapshot of the entire project at our fingertips.
- A VCS that is distributed, such as Git, lets all the team members retrieve a complete history of the project. This allows developers or other stakeholders to use the local Git repositories of any of the teammates even if the main server goes down at any point in time.

19. Describe the branching strategies you have used.

To test our knowledge of the purpose of branching and our experience of branching at a past job, this question is usually asked.

Below topics can help in answering this DevOps interview question -

- Release branching - We can clone the develop branch to create a Release branch once it has enough functionality for a release. This branch kicks off the next release cycle, thus no new features can be contributed beyond this point. The things that can be contributed are documentation generation, bug fixing, and other release-related tasks. The release is merged into master and given a version number once it is ready to ship. It should also be merged back into the development branch, which may have evolved since the initial release.
- Feature branching - This branching model maintains all modifications for a specific feature contained within a branch. The branch gets merged into master once the feature has been completely tested and approved by using tests that are automated.

Task branching - In this branching model, every task is implemented in its respective branch. The task key is mentioned in the branch name. We need to simply look at the task key in the branch name to discover which code implements which task.

20. What is the Blue/Green Deployment Pattern?

This is a method of continuous deployment that is commonly used to reduce downtime. This is where traffic is transferred from one instance to another. In order to include a fresh version of code, we must replace the old code with a new code version.

The new version exists in a green environment and the old version exists in a blue environment. After making changes to the previous version, we need a new instance from the old one to execute a newer version of the instance.

21. What is Continuous Testing?

Continuous Testing constitutes the running of automated tests as part of the software delivery pipeline to provide instant feedback on the business risks present in the most recent release. In order to prevent problems in step-switching in the Software delivery life-cycle and to allow Development teams to receive immediate feedback, every build is continually tested in this manner. This results in significant increase in speed in a developer's productivity as it eliminates the requirement for re-running all the tests after each update and project re-building.

22. What is Automation Testing?

Test automation or manual testing Automation is the process of automating a manual procedure in order to test an application or system. [Automation testing](#) entails the use of independent testing tools that allow you to develop test scripts that can be run repeatedly without the need for human interaction.

23. What are the benefits of Automation Testing?

Some of the advantages of Automation Testing are -

- Helps to save money and time.
- Unattended execution can be easily done.
- Huge test matrices can be easily tested.
- Parallel execution is enabled.
- Reduced human-generated errors, which results in improved accuracy.
- Repeated test tasks execution is supported.

24. How to automate Testing in the DevOps lifecycle?

Developers are obliged to commit all source code changes to a shared DevOps repository.

Every time a change is made in the code, Jenkins-like Continuous Integration tools will grab it from this common repository and deploy it for Continuous Testing, which is done by tools like Selenium.

25. Why is Continuous Testing important for DevOps?

Any modification to the code may be tested immediately with Continuous Testing. This prevents concerns like quality issues and release delays that might occur whenever big-bang testing is delayed until the end of the cycle. In this way, Continuous Testing allows for high-quality and more frequent releases.

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

Continuous Testing key elements are:

- Test Optimization - It guarantees that tests produce reliable results and actionable information. Test Data Management, Test Optimization Management, and Test Maintenance are examples of aspects.

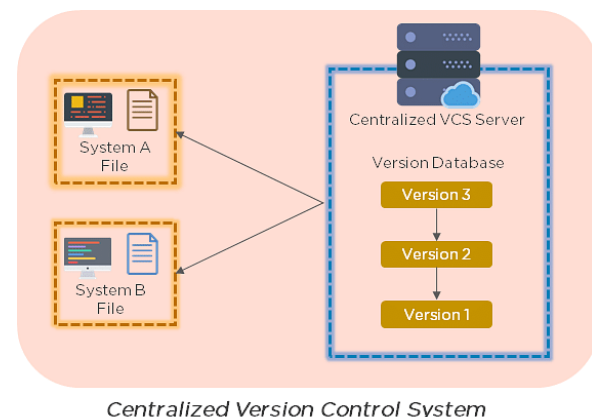
- **Advanced Analysis** - In order to avoid problems from occurring in the first place and to achieve more within each iteration, it employs automation in areas like scope assessment/prioritization, changes effect analysis, and static code analysis.
- **Policy Analysis** - It guarantees that all processes are in line with the organization's changing business needs and that all compliance requirements are met.
- **Risk Assessment** - Test coverage optimization, technical debt, risk mitigation duties, and quality evaluation are all covered to guarantee the build is ready to move on to the next stage.
- **Service Virtualization** - Ensures that real-world testing scenarios are available. Service visualisation provides access to a virtual representation of the needed testing phases, ensuring its availability and reducing the time spent setting up the test environment.
- **Requirements Traceability** - It guarantees that no rework is necessary and real criteria are met. To determine which needs require additional validation, are in jeopardy and performing as expected, an object evaluation is used.

DevOps Source Code Management — Git

27. Explain the difference between a centralized and distributed version control system (VCS).

Centralized Version Control System

- All file versions are stored on a central server
- No developer has a copy of all files on a local system

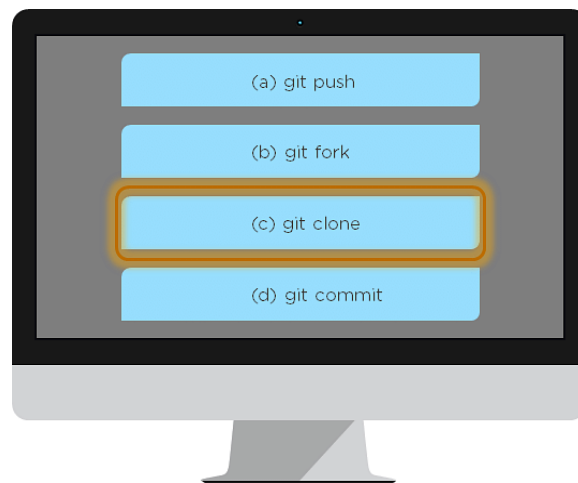


- If the central server crashes, all data from the project will be lost

Distributed Control System

- Every developer has a copy of all versions of the code on their systems
- Enables team members to work offline and does not rely on a single location for backups
- There is no threat, even if the server crashes

28. What is the git command that downloads any repository from GitHub to your computer?



The [git command](#) that downloads any repository from GitHub to your computer is git clone.

29. How do you push a file from your local system to the GitHub repository using Git?

First, connect the local repository to your remote repository:

```
git remote add origin [copied web address]
```

// Ex: git remote add origin <https://github.com/Simplilearn-github/test.git>

Second, push your file to the remote repository:

```
git push origin master
```

30. How is a bare repository different from the standard way of initializing a Git repository?

Using the standard method:

git init

- ✚ You create a working directory with git init
- ✚ A. git subfolder is created with all the git-related revision history Using the bare way

git init --bare

- ✚ It does not contain any working or checked out a copy of source files
- ✚ B. are repositories store git revision history in the root folder of your repository, instead of the .git subfolder

31. Which of the following CLI commands can be used to rename files?

1. git rm
2. git mv
3. git rm -r
4. None of the above

The correct answer is B) git mv

32. What is the process for reverting a commit that has already been pushed and made public?

There are two ways that you can revert a commit:

1. Remove or fix the bad file in a new commit and push it to the remote repository. Then commit it to the remote repository using:

git commit -m "commit message"
2. Create a new commit that undoes all the changes that were made in the bad commit. Use the following command:

git revert <commit id>

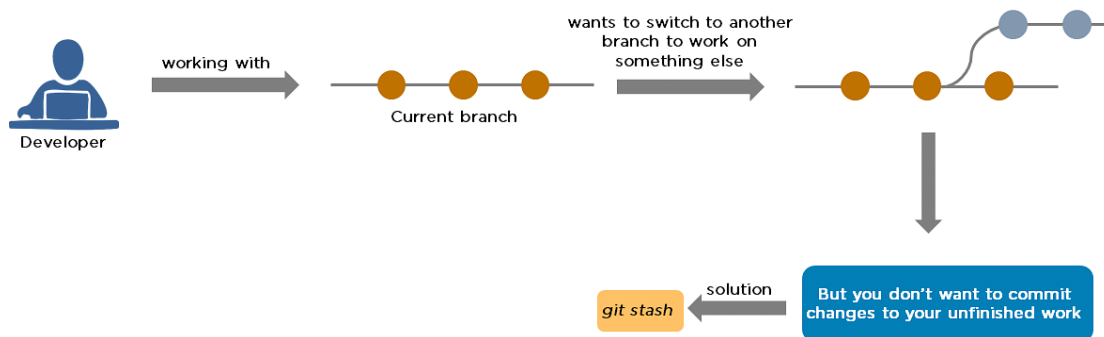
Example: git revert 56de0938f

33. Explain the difference between git fetch and git pull.

Git fetch	Git pull
Git fetch only downloads new data from a remote repository	Git pull updates the current HEAD branch with the latest changes from the remote server
Does not integrate any new data into your working files	Downloads new data and integrate it with the current working files
Users can run a Git fetch at any time to update the remote-tracking branches	Tries to merge remote changes with your local ones
Command - git fetch origin git fetch --all	Command - git pull origin master

34. What is Git stash?

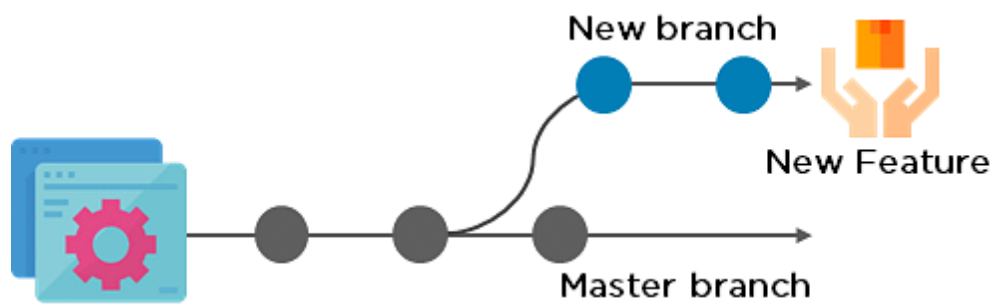
A developer working with a current branch wants to switch to another branch to work on something else, but the developer doesn't want to commit changes to your unfinished work. The solution to this issue is Git stash. Git stash takes your modified tracked files and saves them on a stack of unfinished changes that you can reapply at any time.



35. Explain the concept of branching in Git.

Suppose you are working on an application, and you want to add a new feature to the app. You can create a new branch and build the new feature on that branch.

- By default, you always work on the master branch
- The circles on the branch represent various commits made on the branch
- After you are done with all the changes, you can merge it with the master branch



37. How do you find a list of files that have been changed in a particular commit?

The command to get a list of files that have been changed in a particular commit is:

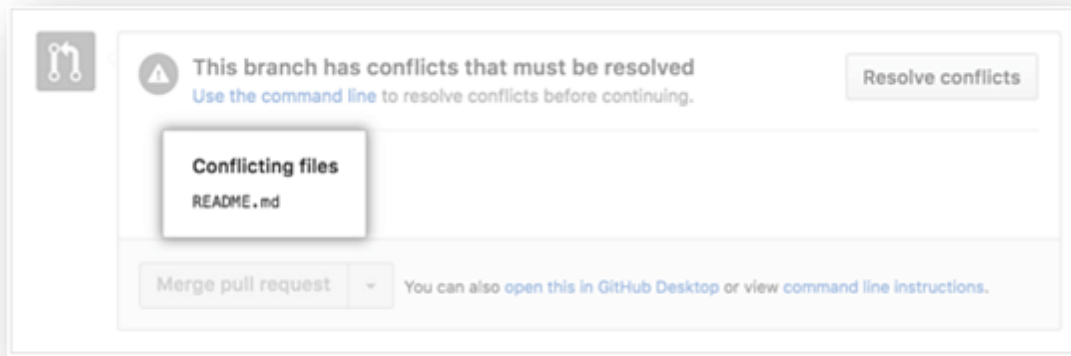
```
git diff-tree -r {commit hash}
```

Example: `git diff-tree -r 87e673f21b`

- -r flag instructs the command to list individual files
- commit hash will list all the files that were changed or added in that commit

38. What is a merge conflict in Git, and how can it be resolved?

A [Git merge conflict](#) happens when you have merge branches with competing for commits, and Git needs your help to decide which changes to incorporate in the final merge.

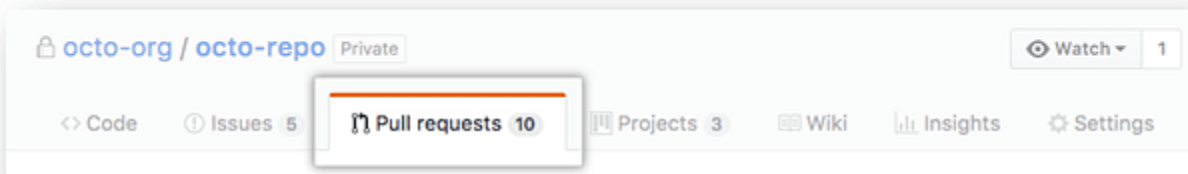


Manually edit the conflicted file to select the changes that you want to keep in the final merge.

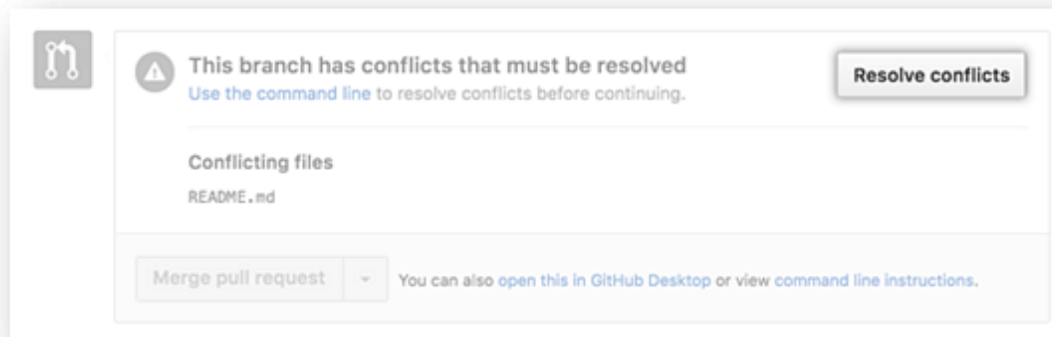
Resolve using GitHub conflict editor

This is done when a merge conflict is caused after competing for line changes. For example, this may occur when people make different changes to the same line of the same file on different branches in your Git repository.

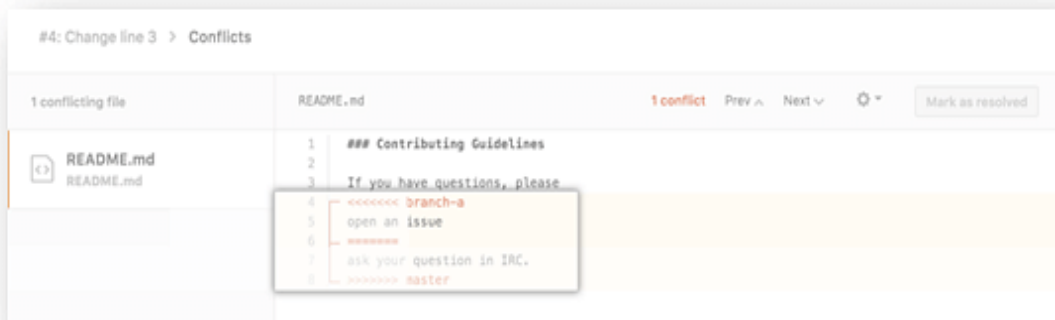
- Resolving a merge conflict using conflict editor:
- Under your repository name, click "Pull requests."



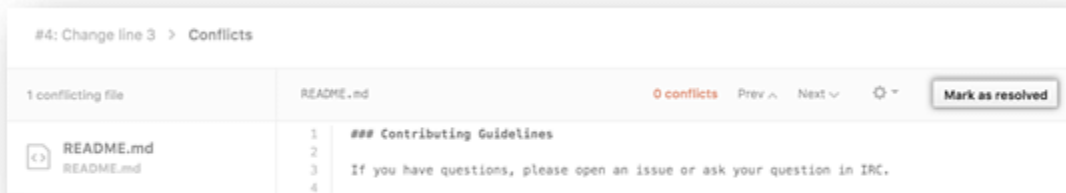
- In the "Pull requests" drop-down, click the pull request with a merge conflict that you'd like to resolve
- Near the bottom of your pull request, click "Resolve conflicts."



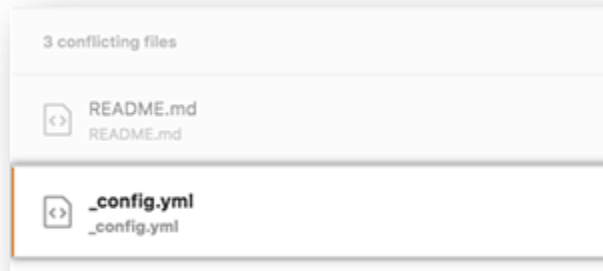
- Decide if you only want to keep your branch's changes, the other branch's changes, or make a brand new change, which may incorporate changes from both branches.
- Delete the conflict markers <<<<<<, =====, >>>>>> and make changes you want in the final merge.



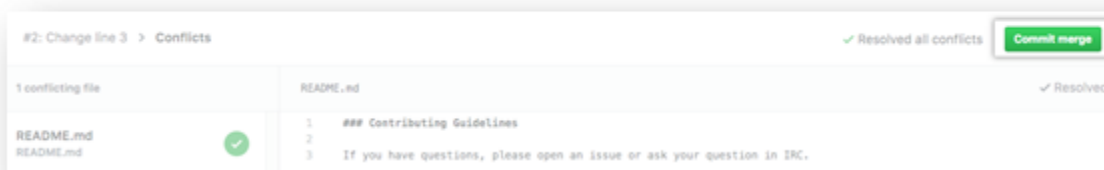
- If you have more than one merge conflict in your file, scroll down to the next set of conflict markers and repeat steps four and five to resolve your merge conflict.
- Once you have resolved all the conflicts in the file, click Mark as resolved.



- If you have more than one file with a conflict, select the next file you want to edit on the left side of the page under "conflicting files" and repeat steps four to seven until you've resolved all of your pull request's merge conflicts.



- Once you've resolved your merge conflicts, click Commit merge. This merges the entire base branch into your head branch.



- To merge your pull request, click Merge pull request.
- A merge conflict is resolved using the command line.
- Open Git Bash.
- Navigate into the local Git repository that contains the merge conflict.

```
cd REPOSITORY-NAME
```

- Generate a list of the files that the merge conflict affects. In this example, the file `styleguide.md` has a merge conflict.

```
$ git status
# On branch branch-b
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:   styleguide.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```

- Open any text editor, such as Sublime Text or Atom, and navigate to the file that has merge conflicts.
- To see the beginning of the merge conflict in your file, search the file for the conflict marker "`<<<<<<<`". Open it, and you'll see the changes from the base branch after the line "`<<<<<<< HEAD.`"
- Next, you'll see "`=====`", which divides your changes from the changes in the other branch, followed by "`>>>>>>> BRANCH-NAME.`".

```
If you have questions, please
<<<<<<< HEAD
open an issue
=====
ask your question in IRC.
>>>>>>> branch-a
```

- Decide if you only want to keep your branch's changes, the other branch's changes, or make a brand new change, which may incorporate changes from both branches.

- Delete the conflict markers "<<<<<<<", "=====", ">>>>>>>" and make the changes you want in the final merge.

In this example, both the changes are incorporated into the final merge:

```
If you have questions, please open an issue or ask in our IRC channel if it's mo
```

- Add or stage your changes.

```
$ git add .
```

- Commit your changes with a comment.

```
$ git commit -m "Resolved merge conflict by incorporating both suggestions."
```

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

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

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

Git bisect command -

```
git bisect <subcommand> <options>
```

The git bisect command is used in finding the bug performing commit in the project by using a binary search algorithm.

The bug occurring commit is called the “bad” commit, and the commit before the bug occurring one is called the “good” commit. We convey the same to the git bisect tool, and it picks a random commit between the two endpoints and prompts whether that one is the “good” or “bad” one. The process continues until the range is narrowed down and the exact commit that introduced the exact change is discovered.

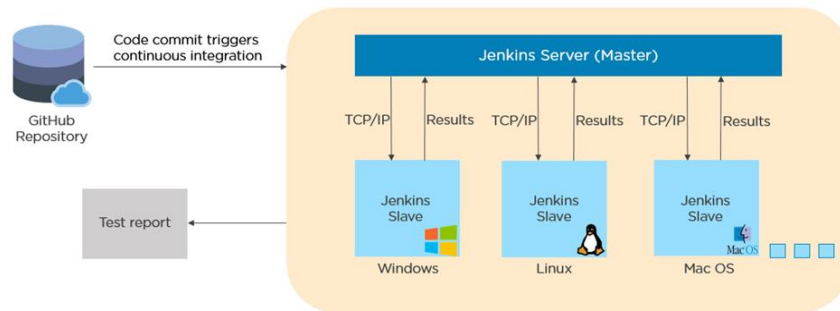
40. Explain some basic Git commands.

Some of the Basic Git Commands are summarized in the below table –

Command	Purpose
git init	Used to start a new repository.
git config - git config --global user.name "[name]" git config --global user.email "[email address]"	This helps to set the username and email to whom the commits belong to.
git clone <repository path>	Used to create a local copy of an existing repository.
git add - git add <file names separated by commas> git add .	Used to add one or more files to the staging area.
git commit - git commit -a git commit -m "<add commit message>"	Creates a snapshot or records of the file(s) that are in the staging area.
git diff - git diff [first branch] [second branch] git diff -staged	Used to show differences between the two mentioned branches/differences made in the files in the staging area vs current version.
git status	Lists out all the files that are to be committed.
git rm <file name(s)>	Used to delete a file(s) from the current working directory and also stages it.
git show <commit>	Shows the content changes and metadata of the mentioned commit.
git branch - git branch [branch name] git branch -d [branch name]	
git branch	The first one creates a brand new branch. The second is used to delete the mentioned branch. The last one lists out all the branches available and also highlights the branch we are in currently.

DevOps Continuous Integration - Jenkins

41. Explain the master-slave architecture of Jenkins.



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

42. What is Jenkins file?

Jenkins file contains the definition of a Jenkins pipeline and is checked into the source control repository. It is a text file.

- It allows code review and iteration on the pipeline.
- It permits an audit trail for the pipeline.
- There is a single source of truth for the pipeline, which can be viewed and edited.

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

1. `java -jar Jenkins.war`
2. `java -war Jenkins.jar`
3. `java -jar Jenkins.jar`
4. `java -war Jenkins.war`

The correct answer is A) `java -jar Jenkins.war`

44. What concepts are key aspects of the Jenkins pipeline?

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

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

1. build.xml
2. pom.xml
3. dependency.xml
4. Version.xml

The correct answer is B) pom.xml

46. Explain the two types of pipeline in Jenkins, along with their syntax.

Jenkins provides two ways of developing a pipeline code: Scripted and Declarative.

A. Scripted Pipeline: It is based on Groovy script as their Domain Specific Language. One or more node blocks do the core work throughout the entire pipeline.

1. Executes the pipeline or any of its stages on any available agent
2. Defines the build stage
3. Performs steps related to building stage
4. Defines the test stage
5. Performs steps related to the test stage
6. Defines the deploy stage
7. Performs steps related to the deploy stage

Jenkinsfile (Scripted Pipeline)

```
node {  
    1  
    stage('Build') {  
        2  
        // 3  
    }  
    stage('Test') {  
        4  
        // 5  
    }  
    stage('Deploy') {  
        6  
        // 7  
    }  
}
```

B. Declarative Pipeline: It provides a simple and friendly syntax to define a pipeline. Here, the pipeline block defines the work done throughout the pipeline.

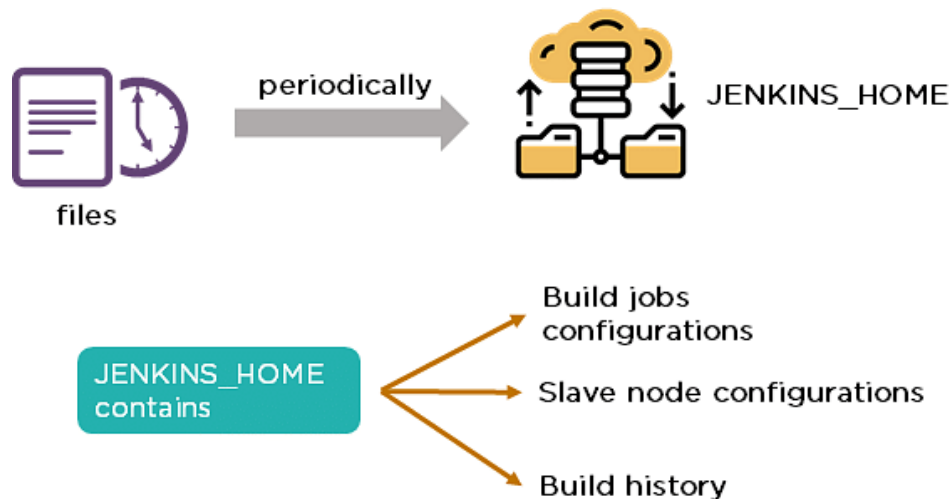
Syntax:

1. Executes the pipeline or any of its stages on any available agent
2. Defines the build stage
3. Performs steps related to building stage
4. Defines the test stage
5. Performs steps related to the test stage
6. Defines the deploy stage
7. Performs steps related to the deploy stage

```
Jenkinsfile (Declarative Pipeline)
pipeline {
  agent any ①
  stages {
    stage('Build') { ②
      steps {
        // ③
      }
    }
    stage('Test') { ④
      steps {
        // ⑤
      }
    }
    stage('Deploy') { ⑥
      steps {
        // ⑦
      }
    }
  }
}
```

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

In order to create a backup file, periodically back up your JENKINS_HOME directory.



In order to create a backup of Jenkins setup, copy the JENKINS_HOME directory. You can also copy a job directory to clone or replicate a job or rename the directory.

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

Copy the jobs directory from the old server to the new one



- Move the job from one Jenkins installation to another by copying the corresponding job directory.
- Create a copy of an existing job by making a clone of a job directory with a different name.
- Rename an existing job by renaming a directory.

49. Name three security mechanisms Jenkins uses to authenticate users.

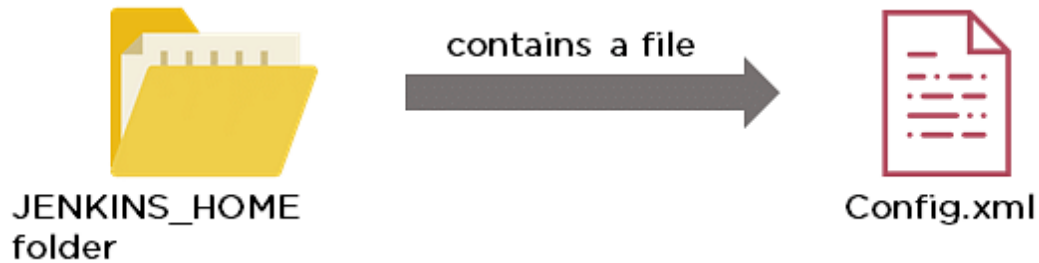
- Jenkins uses an internal database to store user data and credentials.
- Jenkins can use the Lightweight Directory Access Protocol (LDAP) server to authenticate users.
- Jenkins can be configured to employ the authentication mechanism that the deployed application server uses.

50. How is a custom build of a core plugin deployed?

Steps to deploy a custom build of a core plugin:

- Copy the .hpi file to \$JENKINS_HOME/plugins
- Remove the plugin's development directory
- Create an empty file called <plugin>.hpi.pinned
- Restart Jenkins and use your custom build of a core plugin

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



- When security is enabled, the Config file contains an XML element named `useSecurity` that will be set to `true`.
- By changing this setting to `false`, security will be disabled the next time Jenkins is restarted.

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

- By source code management commits.
- After completion of other builds.
- Scheduled to run at a specified time.
- Manual build requests.

53. What are the commands that you can use to restart Jenkins manually?

Two ways to manually restart Jenkins:

1. `(Jenkins_url)/restart` // Forces a restart without waiting for builds to complete
2. `(Jenkins_url)/safeRestart` // Allows all running builds to complete before it restarts

54. Explain how you can set up a Jenkins job?

To create a Jenkins Job, we go to the top page of Jenkins, choose the New Job option and then select Build a free-style software project.

The elements of this freestyle job are -

- Optional triggers for controlling when Jenkins builds.
- Optional steps for gathering data from the build, like collecting javadoc, testing results and/or archiving artifacts.
- A build script (ant, maven, shell script, batch file, etc.) that actually does the work.
- Optional source code management system (SCM), like Subversion or CVS.

DevOps Continuous Testing

55. What are the different Selenium components?

Selenium has the following components: Selenium Integrated Development Environment (IDE)

- It has a simple framework and should be used for prototyping.
- It has an easy-to-install Firefox plug-in.

Selenium Remote Control (RC)

- Testing framework for a developer to write code in any programming language (Java, PHP, Perl, C#, etc.).

[Selenium WebDriver](#)

- Applies a better approach to automate browser activities.
- It does not rely on JavaScript.

Selenium Grid

- Works with Selenium RC and runs tests on different nodes using browsers.

56. What are the different exceptions in Selenium WebDriver?

Exceptions are events that occur during the execution of a program and disrupt the normal flow of a program's instructions. Selenium has the following exceptions:

- Timeout Exception - It is thrown when a command performing an operation does not complete in the stipulated time.
- No Such Element Exception - It is thrown when an element with specific attributes is not found on the web page.
- Element Not Visible Exception - It is thrown when an element is present in Document Object Model (DOM) but is not visible. Ex: Hidden Elements defined in HTML using type="hidden".
- Session NotFoundException - The WebDriver is performing the action immediately after quitting the browser.

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

Selenium is capable of testing an application on an Android browser using an Android driver. You can use the Selendroid or Appium framework to test native apps or web apps in the Android browser. The following is a sample code:

```
@BeforeClass
public static void startSelendroidServer() throws Exception {
    if (selendroidServer != null) {
        selendroidServer.stopSelendroid();
    }
    SelendroidConfiguration config = new SelendroidConfiguration();
    config.addSupportedApp("selendroid-test-app-0.9.0.apk");
    selendroidServer = new SelendroidLauncher(config);
    selendroidServer.launchSelendroid();
    SelendroidCapabilities caps = new SelendroidCapabilities("io.selendroid.testapp:0.9.0");
    driver = new SelendroidDriver(caps);
}
```

58. What are the different test types that Selenium supports?

Functional - This is a type of black-box testing in which the test cases are based on the software specification.

Regression - This testing helps to find new errors, regressions, etc. in different functional and non-functional areas of code after the alteration.

Load Testing - This testing seeks to monitor the response of a device after putting a load on it. It is carried out to study the behavior of the system under certain conditions.

59. How can you access the text of a web element?

Get command is used to retrieve the text of a specified web element. The command does not return any parameter but returns a string value.

Used for:

- Verification of messages
- Labels
- Errors displayed on the web page

Syntax:

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

60. Which of these options is not a WebElement method?

1. `getText()`
2. `size()`
3. `getTagName()`
4. `sendKeys()`

The correct answer is B) `size()`

61. How can you handle keyboard and mouse actions using Selenium?

You can handle keyboard and mouse events with the advanced user interaction API. The advanced user interactions API contains actions and action classes.

Method	Description
clickAndHold()	Clicks without releasing the current mouse location
dragAndDrop()	Performs click-and-hold at the location of the source element
keyDown(modifier_key)	Performs a modifier key press (ctrl, shift, Fn, etc.)
keyUp(modifier_key)	Performs a key release

62. When do we use `findElement()` and `findElements()`?

A. `findElement()`

It finds the first element in the current web page that matches the specified locator value.

Syntax:

```
WebElement element=driver.findElement(By.xpath("//div[@id='example']/ul/li"));
```

B. `findElements()`

It finds all the elements in the current web page that matches the specified locator value.

Syntax:

```
List elementList=driver.findElements(By.xpath("//div[@id='example']/ul/li"));
```

63. What are driver.close() and driver.quit() in WebDriver?

These are two different methods used to close the browser session in Selenium WebDriver:

- driver.close() - This is used to close the current browser window on which the focus is set. In this case, there is only one browser open.
- driver.quit() - It closes all the browser windows and ends the WebDriver session using the driver.dispose method.

64. How can you submit a form using Selenium?

The following lines of code will let you submit a form using Selenium:

```
WebElement el = driver.findElement(By.id("ElementID"));

el.submit();
```

65. What are the Testing types supported by Selenium?

There are two types of testing that are primarily supported by Selenium:

Functional Testing - Individual testing of software functional points or features.

Regression Testing - Wherever a bug is fixed, a product is retested and this is called Regression Testing.

66. What is Selenium IDE?

Selenium integrated development environment (IDE) is an all-in-one Selenium script development environment. It may be used to debug tests, alter and record and is also available as a Firefox extension. Selenium IDE comes with the whole Selenium Core that allows us to rapidly and easily replay and record tests in the exact environment where they will be conducted.

Selenium IDE is the best environment for building Selenium tests, regardless of the style of testing we prefer, thanks to the ability to move instructions around rapidly and the autocomplete support.

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

The difference between Verify and Assert commands in Selenium are:

- The verify commands determine whether or not the provided condition is true. The program execution does not halt regardless of whether the condition is true or not, i.e., all test steps will be completed, and verification failure will not stop the execution.
- The assert command determines whether a condition is false or true. To know whether the supplied element is on the page or not, we do the following. The next test step will be performed by the program control, if the condition is true. However, no further tests will be run, and the execution will halt, if the condition is false.

68. How to launch Browser using WebDriver?

To launch Browser using WebDriver, following syntax is followed -

```
WebDriver driver = new InternetExplorerDriver();
```

```
WebDriver driver = new ChromeDriver();
```

```
WebDriver driver = new FirefoxDriver();
```

69. What is the difference between Asset Management and Configuration Management?

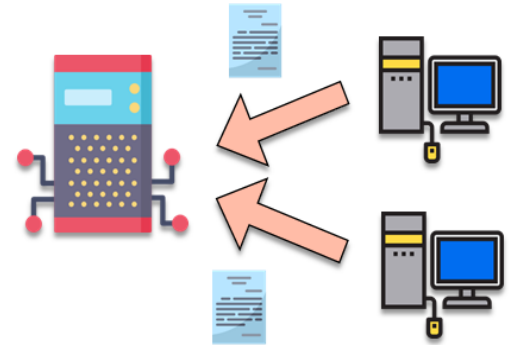
Differences between Configuration Management and Asset Management are:

Configuration Management	Asset Management
Operational Relationships.	Incidental relationships only.
Maintains troubleshooting data.	Maintains taxes data.
Everything we deploy is scope.	Everything we own is scope.
Deployment to retirement - lifecycle.	Purchase to disposal - lifecycle.
Operations - main concern.	Finances - main concern.
ITIL processes from interfacing	Leasing and purchasing from interfacing.

DevOps Configuration Management — Chef, Puppet, Ansible

69. Why are SSL certificates used in Chef?

- SSL certificates are used between the Chef server and the client to ensure that each node has access to the right data.
- Every node has a private and public key pair. The public key is stored at the Chef server.
- When an SSL certificate is sent to the server, it will contain the private key of the node.
- The server compares this against the public key in order to identify the node and give the node access to the required data.



70. Which of the following commands would you use to stop or disable the 'httpd' service when the system boots?

1. `# systemctl disable httpd.service`
2. `# system disable httpd.service`
3. `# system disable httpd`
4. `# systemctl disable httpd.service`

The correct answer is A) `# systemctl disable httpd.service`

71. What is Test Kitchen in Chef?

Test Kitchen is a command-line tool in Chef that spins up an instance and tests the cookbook on it before deploying it on the actual nodes.

Here are the most commonly used kitchen commands:

```
Command Prompt

$ kitchen create           //create instances
$ kitchen converge         //combines multiple instances
$ kitchen verify           //verify instances
$ kitchen destroy          //destroy instances
$ kitchen setup            //setup instances
```

72. How does chef-apply differ from chef-client?

- chef-apply is run on the client system.

chef-apply applies the recipe mentioned in the command on the client system.

```
$ chef-apply recipe_name.rb
```

- chef-client is also run on the client system.

chef-client applies all the cookbooks in your server's run list to the client system.

```
$ knife chef-client
```

73. What is the command to sign the requested certificates?

- For Puppet version 2.7:

```
# puppetca --sign hostname-of-agent
```

Example:

```
# puppetca --sign ChefAgent
```

```
# puppetca sign hostname-of-agent
```

Example:

```
# puppetca sign ChefAgent
```

- For Puppet version 2.7:

```
# puppetca --sign hostname-of-agent
```

Example:

```
# puppetca --sign ChefAgent
```

```
# puppetca sign hostname-of-agent
```

Example:

```
# puppetca sign ChefAgent
```

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

- Changes in the configuration are tracked using Jira, and further maintenance is done through internal procedures.
- Version control takes the support of Git and Puppet's code manager app.
- The changes are also passed through Jenkin's continuous integration pipeline.

75. What are the resources in Puppet?

- Resources are the basic units of any configuration management tool.
- These are the features of a node, like their software packages or services.
- A resource declaration, written in a catalog, describes the action to be performed on or with the resource.
- When the catalog is executed, it sets the node to the desired state.

76. What is a class in Puppet?

Classes are named blocks in your manifest that configure various functionalities of the node, such as services, files, and packages.

The classes are added to a node's catalog and are executed only when explicitly invoked.

```
Class apache (String $version = 'latest') {
```

```
  package{
```

```
    'httpd': ensure => $version,
```

```
    before => File['/etc/httpd.conf'],}
```

77. What is an Ansible role?

An Ansible role is an independent block of tasks, variables, files, and templates embedded inside a playbook.

This playbook installs tomcat on node1.

```
---
- hosts: node1
  roles
    - {role: install-
      tomcat}
```

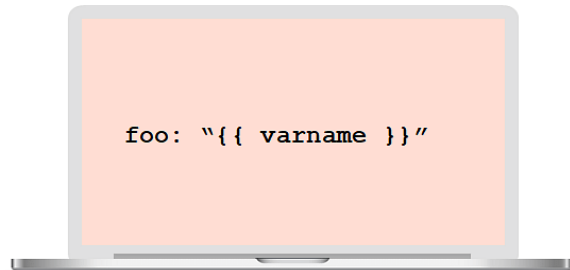
78. When should I use '{{ }}'?

Always use `{{}}` for variables, unless you have a conditional statement, such as "when: ...". This is because conditional statements are run through Jinja, which resolves the expressions.

For example:

```
echo "This prints the value of {{foo}}"
```

```
when : foo is defined
```



Using brackets makes it simpler to distinguish between strings and undefined variables.

This also ensures that Ansible doesn't recognize the line as a dictionary declaration.

79. What is the best way to make content reusable/redistributable?

There are three ways to make content reusable or redistributable in Ansible:

- Roles are used to managing tasks in a playbook. They can be easily shared via Ansible Galaxy.
- "include" is used to add a submodule or another file to a playbook. This means a code written once can be added to multiple playbooks.
- "import" is an improvement of "include," which ensures that a file is added only once. This is helpful when a line is run recursively.

80. How is Ansible different from Puppet?

Ansible	Puppet
Easy agentless installation	Agent-based installation
Based on Python	Based on Ruby
Configuration files are written in YAML	Configuration files are written in DSL
No support for Windows	Support for all popular OS's

DevOps Interview Questions on Containerization

81. Explain the architecture of Docker.?

[Docker](#) uses a client-server architecture.

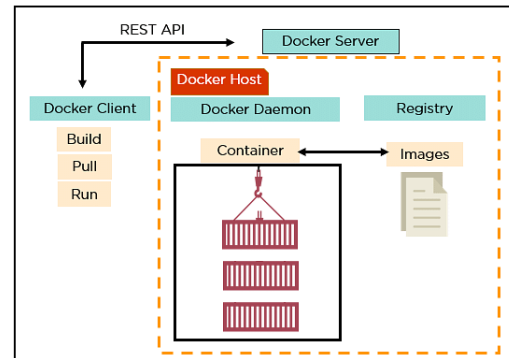
Docker Client is a service that runs a command. The command is translated using the REST API and is sent to the Docker Daemon (server).

Docker Daemon accepts the request and interacts with the operating system to build Docker images and run Docker containers.

A Docker image is a template of instructions, which is used to create containers.

[Docker container](#) is an executable package of an application and its dependencies together.

Docker registry is a service to host and distribute Docker images among users.



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

Criteria	Virtual Machine	Docker
Memory space	Occupies a lot of memory space	Docker containers occupy less space
Boot-up time	Long boot-up time	Short boot-up time
Performance	Running multiple virtual machines leads to unstable performance	Containers have a better performance, as they are hosted in a single Docker engine
Scaling	Difficult to scale up	Easy to scale up
Efficiency	Low efficiency	High efficiency
Portability	Compatibility issues while porting across different platforms	Easily portable across different platforms
Space allocation	Data volumes cannot be shared	Data volumes are shared and used again across multiple containers

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

- It is possible to share Docker containers on different nodes with [Docker Swarm](#).
- Docker Swarm is a tool that allows IT administrators and developers to create and manage a cluster of swarm nodes within the Docker platform.
- A swarm consists of two types of nodes: a manager node and a worker node.

84. What are the commands used to create a Docker swarm?

- Create a swarm where you want to run your manager node.

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

- Once you've created a swarm on your manager node, you can add worker nodes to your swarm.
- When a node is initialized as a manager, it immediately creates a token. In order to create a worker node, the following command (token) should be executed on the host machine of a worker node.

```
docker swarm join \ --token  
SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8v xv8rssmk743ojnwac  
r2e7c \ 192.168.99.100:2377
```

85. How do you run multiple containers using a single service?

- It is possible to run multiple containers as a single service with Docker Compose.
- Here, each container runs in isolation but can interact with each other.
- All Docker Compose files are YAML files



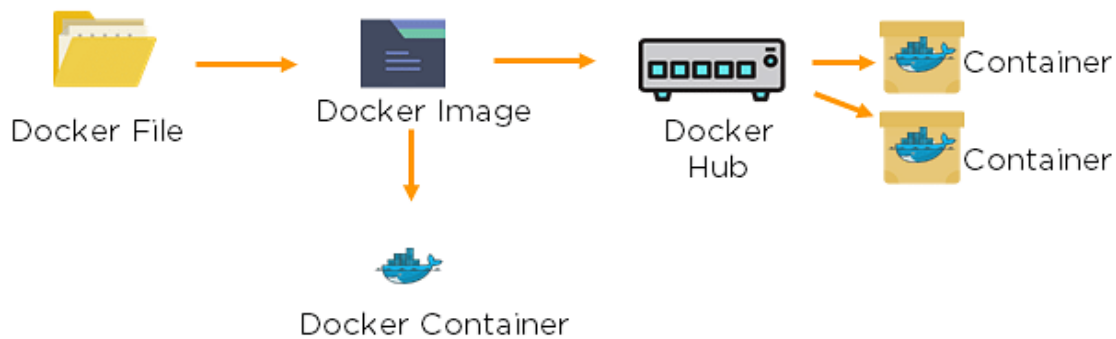
86. Instead of YAML, what can you use as an alternate file for building Docker compose?

To build a Docker compose, a user can use a JSON file instead of YAML. In case a user wants to use a JSON file, he/she should specify the filename as given:

Docker-compose -f Docker-compose.json up

87. What is a Dockerfile used for?

- A Dockerfile is used for creating Docker images using the build command.
- With a Docker image, any user can run the code to create Docker containers.
- Once a Docker image is built, it's uploaded in a Docker registry.
- From the Docker registry, users can get the Docker image and build new containers whenever they want.



88. Explain the differences between Docker images and Docker containers.

Docker Images	Docker Container
Docker images are templates of Docker containers	Containers are runtime instances of a Docker image
An image is built using a Docker file	Containers are created using Docker images
It is stored in a Docker repository or a Docker hub	They are stored in the Docker daemon
The image layer is a read-only filesystem	Every container layer is a read-write filesystem

89. How do you create a Docker container?

Task: Create a MySQL Docker container

A user can either build a Docker image or pull an existing Docker image (like MySQL) from Docker Hub.

Now, Docker creates a new container MySQL from the existing Docker image. Simultaneously, the container layer of the read-write filesystem is also created on top of the image layer.

- Command to create a Docker container: `Docker run -t -i MySQL`
- Command to list down the running containers: `Docker ps`

90. What is the difference between a registry and a repository?

Registry	Repository
A Docker registry is an open-source server-side service used for hosting and distributing Docker images	The repository is a collection of multiple versions of Docker images
In a registry, a user can distinguish between Docker images with their tag names	It is stored in a Docker registry
Docker also has its own default registry called Docker Hub	It has two types: public and private repositories

91. What are the cloud platforms that support Docker?

The following are the cloud platforms that Docker runs on:

- [Amazon Web Services](#)
- [Microsoft Azure](#)
- Google Cloud Platform
- Rackspace



92. What is the purpose of the expose and publish commands in Docker?

Expose

- Expose is an instruction used in Dockerfile.
- It is used to expose ports within a Docker network.
- It is a documenting instruction used at the time of building an image and running a container.
- Expose is the command used in Docker.
- Example: Expose 8080

Publish

- Publish is used in a Docker run command.
- It can be used outside a Docker environment.
- It is used to map a host port to a running container port.
- --publish or -p is the command used in Docker.
- Example: docker run -d -p 0.0.0.80:80

Now, let's have a look at the DevOps interview questions for continuous monitoring.

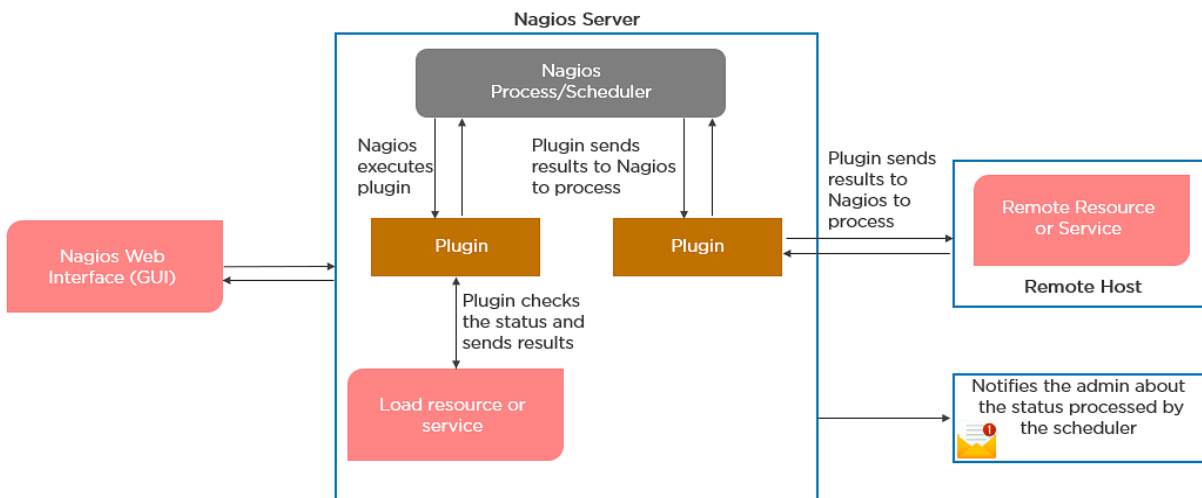
DevOps Continuous Monitoring

93. How does Nagios help in the continuous monitoring of systems, applications, and services?

Nagios enables server monitoring and the ability to check if they are sufficiently utilized or if any task failures need to be addressed.

- Verifies the status of the servers and services
- Inspects the health of your infrastructure
- Checks if applications are working correctly and web servers are reachable

94. How does Nagios help in the continuous monitoring of systems, applications, and services?



95. What do you mean by Nagios Remote Plugin Executor (NPRE) of Nagios?

Nagios Remote Plugin Executor (NPRE) enables you to execute Nagios plugins on Linux/Unix machines. You can monitor remote machine metrics (disk usage, CPU load, etc.)

- The check_npre plugin that resides on the local monitoring machine
- The NPRE daemon that runs on the remote Linux/Unix machine

96. What are the port numbers that Nagios uses for monitoring purposes?

Usually, Nagios uses the following port numbers for monitoring:

	5666
	5667
	5668

97. What are active and passive checks in Nagios?

Nagios is capable of monitoring hosts and services in two ways:

Actively

- Active checks are initiated as a result of the Nagios process
- Active checks are regularly scheduled

Passively

- Passive checks are initiated and performed through external applications/processes
- Passive checks results are submitted to Nagios for processing

98. What are active and passive checks in Nagios?

Active Checks:

The check logic in the Nagios daemon initiates active checks.

Nagios will execute a plugin and pass the information on what needs to be checked.

The plugin will then check the operational state of the host or service, and report results back to the Nagios daemon.

It will process the results of the host or service check and send notifications.

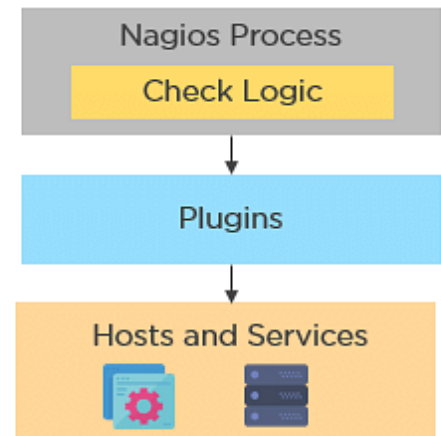
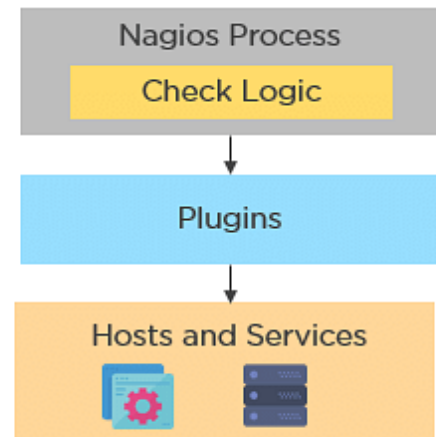
Passive Checks:

In passive checks, an external application checks the status of a host or service.

It writes the results of the check to the external command file.

Nagios reads the external command file and places the results of all passive checks into a queue for later processing.

Nagios may send out notifications, log alerts, etc. depending on the check result information.



99. Explain the main configuration file and its location in Nagios.

The main configuration file consists of several directives that affect how Nagios operates. The Nagios process and the CGIs read the config file.

A sample main configuration file will be placed into your settings directory:

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

100. What is the Nagios Network Analyzer?

- It provides an in-depth look at all network traffic sources and security threats.
- It provides a central view of your network traffic and bandwidth data.
- It allows system admins to gather high-level information on the health of the network.
- It enables you to be proactive in resolving outages, abnormal behavior, and threats before they affect critical business processes.

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

HTTP certificate monitoring

- Increased server, services, and application availability.
- Fast detection of network outages and protocol failures.
- Enables web transaction and web server performance monitoring.

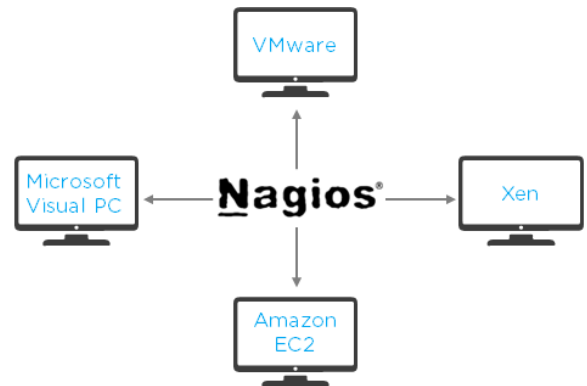
SSL certificate monitoring

- Increased website availability.
- Frequent application availability.
- It provides increased security.

102. Explain virtualization with Nagios.

Nagios can run on different virtualization platforms, like VMware, Microsoft Visual PC, Xen, Amazon EC2, etc.

- Provides the capabilities to monitor an assortment of metrics on different platforms
- Ensures quick detection of service and application failures
- Has the ability to monitor the following metrics:
- CPU Usage
- Memory
- Networking
- VM status
- Reduced administrative overhead



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

name - Template name that can be referenced in other object definitions so it can inherit the object's properties/variables.

use - Here, you specify the name of the template object that you

want to inherit properties/variables from.

register - This variable indicates whether or not the object definition

should be registered with Nagios.

define someobjecttype{

object-specific variables

name template_name

use name_of_template

register [0/1] }

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



Using the object configuration format, you can create object definitions that inherit properties from other object definitions. Hence, Nagios is known as object-oriented.

Types of Objects:

- Services
- Hosts
- Commands
- Time Periods

105. Explain what state stalking is in Nagios.

- State stalking is used for logging purposes in Nagios.
- When stalking is enabled for a particular host or service, Nagios will watch that host or service very carefully.
- It will log any changes it sees in the output of check results.
- This helps in the analysis of log files.