GIT Interview Questions

1. Why should you use Jenkins?

Answer:

- Jenkins is a highly configurable system by itself
- Integration with several plugins provides more flexibility. It's easy to write plugins.
- By combining Jenkins with Ant, Gradle, or other Build Automation tools, the possibilities are limitless
- There is a large support community and thorough documentation
- Complete build history
- Easy integration of Version Control Software, Build, Reporting and Deployment Tools
- Customization and extension via custom scripts and plugins
- Easy tracking of bugs at early stage in development environment than production

2. You are asked to backup Jenkins. How would you approach this task?

Answer: Use the backup plugin. To use this, determine if you want to back up all data or just a subset. Alternatively, script a physical copy using shell commands/Java CLI for Jenkins.

3. You are required to run 2 jobs in parallel while a 3rd job is to run after these have completed. How will you achieve this in Jenkins?

Answer: Use the Pipeline plugin. (TIP: You may be grilled in more depth on knowledge of Groovy in doing this task). Alternatively, you can use the Build Flow Plugin (TIP: If you answer this question, be prepared to answer questions on the DSL syntax, including passing parameters and error handling. Just as a FYI, the pipeline plugin is motivated by the Build Flow Plugin – you do not need both)

4. Name some common plugins you have used.

Answer:

- Email Ext
- GIT
- Build Pipelines
- Pipeline
- Flow
- Deployment Plugin
- OAuth

TIP: Mention the plugins you have used, not just what is in this list. Make sure you answer at-least 5 non-trivial plugins (The Green Balls plugin is trivial) and be sure to know all configuration details for these plugins

5. How will you revert a commit already made?

Answer: One or more commits can be reverted using git revert. This command creates a new commit with patches that cancel out the changes introduced in specific commits. Alternatively, one can always checkout the state of a commit from the past, and commit it anew.

6. I do not want to change repository history with git revert? What do I do?

Answer: Running the following command will revert the last two commits: git revert HEAD~2..HEAD

7. What is squashing? Where is it used? How do you use it?

Answer: Squashing is the process of combining multiple commits into one. It is usually done in feature branches.

E.g. The following command will squash last N commits into one:

git rebase -I HEAD~{N}

The editor will open. Each of the commits will begin with pick. Replace pick with squash (s) for all lines except the first one, save and exit.

8. Every time someone commits a change, I want some program to run. How do you do this?

Answer: Write pre-commit, update or post-commit hooks.

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

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

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

Answer: git bisect allows the user to perform binary search on the entire history of a repository. By issuing the command git bisect start, the repository enters bisect mode. After this, all you must do is identify a bad and a good commit:

- git bisect bad #marks the current version as bad
- git bisect good {hash or tag} #marks the given hash or tag as good, ideally of some earlier commit

Once this is done, GIT will then have a range of commits that it needs to explore. At every step, it will checkout a certain commit from this range, and require you to identify it as good or bad. After which the range will be effectively halved, and the whole search will require a lot less number of steps than the actual number of commits involved in the range. Once the first bad commit has been found, or the bisect mode needs to be ended, the following command can be used to exit the mode and reset the bisection state:

git bisect reset

10. What are the different ways you can refer to a commit?

Answer: In GIT, each commit is given a unique hash. These hashes can be used to identify the corresponding commits in various scenarios (such as while trying to checkout a state of the code using the git checkout {hash} command).

Additionally, GIT also maintains several aliases to certain commits, known as refs. Also, every tag that you create in the repository effectively becomes a ref (and that is exactly why you can use tags instead of commit hashes in various GIT commands). GIT also maintains several special aliases that change based on the state of the repository, such as HEAD, FETCH_HEAD, MERGE_HEAD, etc.

GIT also allows commits to be referred as relative to one another. For example, HEAD~1 refers to the commit parent to HEAD, HEAD~2 refers to the grandparent of HEAD, and so on. In case of merge commits, where the commit has two parents, ^ can be used to select one of the two parents, e.g. HEAD^2 can be used to follow the second parent.

Refspecs are used to map local and remote branches together. However, these can be used to refer to commits that reside on remote branches allowing one to control and manipulate them from a local GIT environment.

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

Answer: In simple words, git rebase allows one to move the first commit of a branch to a new starting location. For example, if a feature branch was created from master, and since then the master branch has received new commits, git rebase can be used to move the feature branch to the tip of master. The command effectively will replay the changes made in the feature branch at the tip of master, allowing conflicts to be resolved in the process. When done with care, this will allow the feature branch to be merged into master with relative ease and sometimes as a simple fast-forward operation.

12. What is a staging area or Index in GIT?

Answer: Before completing the commits, it can be formatted and reviewed in an intermediate area known as 'Staging Area' or 'Index'.

13. What is git stash?

Answer: git stash takes the current state of the working directory and index and puts in on the stack for later use and gives you a clean working directory. So, in case if you are in the middle of something and need to jump over to the other job, and at the same time you don't want to lose your current edits then you can use git stash.

14. What is git stash drop?

Answer: When you are done with the stashed item or want to remove it from the list, run the git stash drop command. It will remove the last added stash item by default, and it can also remove a specific item if you include as an argument.

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

Answer:

git branch --merged lists the branches that have been merged into the current branch

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

16. What is the function of git clone?

Answer: The git clone command creates a copy of an existing GIT repository. To get the copy of a central repository, 'cloning' is the most common way used by programmers.

17. What is the function of git config?

Answer: The git config command is a convenient way to set configuration options for your GIT installation. Behavior of a repository, user info, preferences etc. can be defined through this command.

18. What does commit object contain?

Answer:

a) A set of files, representing the state of a project at a given point of time

b) Reference to parent commit objects

c) An SHA-1 name, a string that uniquely identifies the commit object.

19. How can you create a repository in GIT?

Answer: In GIT, to create a repository, create a directory for the project if it does not exist, and then run git init command. By running this command, .git directory will be created in the project directory, the directory does not need to be empty.

20. What is 'head' in GIT and how many heads can be created in a repository?

Answer: A 'head' is simply a reference to a commit object. In every repository, there is a default head referred as "Master". A repository can contain any number of heads.

21. What is the purpose of branching in GIT?

Answer: The purpose of branching in GIT is that you can create your own branch and jump between those branches. It will allow you to go to your previous work keeping your recent work intact.

22. What is the function of git diff/status in GIT?

Answer: git diff shows the changes between commits, working tree etc. git status shows you the difference between the working directory and the index, it clarifies GIT more comprehensively.

23. What is the function of git checkout in GIT?

Answer: A git checkout command is used to update directories or specific files in your working tree with those from another branch without merging it in the whole branch.

24. What is the function of git rm?

Answer: To remove the file from the staging area and off your disk git rm is used.

25. What is the function of git stash apply?

Answer: When you want to continue working where you have left your work, git stash apply command is used to bring back the saved changes onto the working directory.

26. What is the use of git log?

Answer: To find specific commits in your project history by author, date, content or history git log is used.

27. What is git add used for?

Answer: git add adds file changes in your existing directory to your index.

28. What is the function of git reset?

Answer: The function of git reset is to reset your index as well as the working directory to the state of your last commit.

29. What is git ls-tree?

Answer: git ls-tree represents a tree object including the mode and the name of each item and the SHA-1 value of the blob or the tree.

Note: Much of the questions are about testing your technical ability. The more challenging questions relate to specific branching and merging strategies, especially related to the longevity of branches, difference between tags and branches and pull v/s rebase questions. Such questions must be answered as far as possible in context of what is being practiced currently in your organization.