**1. What is Git? What is the difference between Git and GitHub?**

**Answer:**Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.

**2. What is the advantage of using GitHub for Selenium?**

**Answer:**GitHub is a cloud-based hosting service that lets you manage Git repositories; it helps to have a backup code in case of physical failures.

* github supports branching, so we can have multiple versions of code.
* github supports project cloning, so it helps in easily distribution of project across multiple teams and multiple locations
* github supports code pull so anyone with access rights can pull code in local machine. This can also be integrated with jenkins.
* github supports code push so anyone with access rights can checkin code in github central repository.

**3. How to handle git conflicts?**

**Answer:**Git can handle on its own most merges by using its automatic merging features. There arises a conflict when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other. Conflicts are most likely to happen when working in a team environment.

* Identify the files that have caused the conflict.
* Make the necessary changes in the files so that conflict does not arise again.
* Add these files by the command git add.
* Finally to commit the changed file using the command git commit

Please refer below YouTube video link

<https://www.youtube.com/watch?v=CQqdTDBlopg&feature=youtu.be>

**4. Explain different Git commands?**

**Answer:**Below are the most git commands

**1. Initialize a repo**

Create an empty git repo or re-initialize an existing one

$ git init [repository path]

**2. git remote add [variable name] [Remote Server Link]**

This command is used to connect your local repository to the remote server.

Example:

git remote add origin https://github.com/hverma22/Test5

**3. git clone**

This command is used to obtain a repository from an existing URL.

git clone [url]

Example: navigate to your repo path where you want to clone and write below command in cmd

git clone https://github.com/hverma22/Test2

**4. How to Create a New Branch in Git**

To create a new branch use:

$ git checkout -b <new\_branch\_name>

**5. How to List Branches in Git**

$ git branch

Example output:

develop

my\_feature

master

**6. git log**

git log

This command is used to list the version history for the current branch.

Example:

git log --oneline

**7. git merge**

git merge [branch name]

This command merges the specified branch’s history into the current branch.

**8. How to Switch Branches in Git**

When you create a new branch then Git automatically switches to the new branch.

If you have multiple branches, then you can easily switch between branches with git checkout:

$ git checkout master

$ git checkout develop

$ git checkout my\_feature

You can get the specific previous version as well.

Command: git checkout <ChangeID> <filePath with extenion>

Example:

git checkout 6475fgh5 pom.xml

**9. How to Delete Branches in Git**

To delete a local branch:

$ git branch -d <local\_branch>

To delete a remote branch on origin:

$ git push origin :<remote\_branch>

**10. Git Stage Files**

To stage or simply add files, you need to use git add command. You can stage individual files:

$ git add foo.js

or all files at once:

$ git add .

**11. git diff**

This command shows the file differences which are not yet staged.

Example:

git diff –-staged

git diff [first branch] [second branch]

**12.Git Unstage Changes**

If you want to remove a certain file from the stage:

$ git reset HEAD foo.js

Or remove all staged files:

$ git reset HEAD .

**13. Git Status**

If you want to see what files have been created, modified or deleted, Git status will show you a report.

$ git status

**14. git rm**

This command deletes the file from your working directory and stages the deletion.

git rm [file]

**15.git commit**

git commit -m “[commit message]”

This command records or snapshots the file permanently in the version history.

Example:

git commit -m “First Commit”

**16. git show**

git show [commit]

This command shows the metadata and content changes of the specified commit.

Command: git show <ChangeID>:<FilePath>

Example:

git show 45dhfg56:/src/test/newtest.xml

**17. Undoing Commits**

The following command will undo your most recent commit and put those changes back into staging, so you don’t lose any work:

$ git reset --soft HEAD~1

To completely delete the commit and throw away any changes use:

$ git reset --hard HEAD~1

**18. git push** - After you have committed your changes, next is to push to a remote repository.

git push [variable name] master

This command sends the committed changes of master branch to your remote repository.

Example:

Push a local branch for the first time:

git push origin master

git push origin master --force

After that, then you can just use

$ git push

**19. To push a local branch to a different remote branch**, you can use:

$ git push origin <local\_branch>:<remote\_branch>

**20. Undo Last Push**

If you have to undo your last push, you can use:

$ git reset --hard HEAD~1 && git push -f origin master

**21.git config**

This command sets the author name and email address respectively to be used with your commits.

git config –global user.name “[name]”

git config –global user.email “[email address]”

Example:

git config user.name "Hitendra Kuamar Verma"

git config user.email "Hitendra@Hitendra-PC"

**22. git pull**

git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.

Example:

git pull https://github.com/hverma22/Test2.git

**5. What is the version control tool you are using and tell me the steps what you follow and how will you resolve conflicts?**

**Answer:**Please refer below YouTube video link

<https://www.youtube.com/watch?v=CQqdTDBlopg&feature=youtu.be>

**6. What is the difference between SVN & GIT?**

**Answer:**Below are the differences -

* Git is a distributed VCS; SVN is a non-distributed VCS.
* Git uses multiple repositories including a centralized repository and server, as well as some local repositories; SVN is a centralized version control system.
* The content in Git is stored as metadata; SVN stores files of content.
* Git branches are easier to work with than SVN branches.
* Git does not have the global revision number feature like SVN has.
* Git has better content protection than SVN.
* Git was developed for Linux kernel by Linus Torvalds; SVN was developed by CollabNet, Inc.
* Git belongs to the 3rd generation of Version Control tools; SVN belongs to the 2nd generation of Version Control tools

**7. How do you maintain source code in GIT?**

**Answer: Please refer below link –**

<https://www.automationtestinginsider.com/2020/06/selenium-integration-with-git-and-github.html>

**8. Suppose there are 10 classes & I want to push only 5 classes, how do you do that?**

**Answer:**Normally we commit to git, all files are going to git but in your scenario push only single file git. For this, you have to run specific command to push the only single file to git.

$ git commit -m "Message goes here" filename

**Example to push to single file to git**

$ git commit -m "Pushing Only Single file to git" config/file1.txt

Let’s take look how to push one or two or three files to git in a single commit.

$ git commit -m "Message goes here" file1 file2 file3

**Example to push to three files to git**

$ git commit -m "Pushing Only three files to git" config/file1.txt  config/file2.txt config/file3.txt

**9. Mention the various Git repository hosting functions.**

**Answer:**

* Github
* Gitlab
* Bitbucket
* SourceForge
* GitEnterprise

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

**Answer:** There can be two approaches to tackle this question and make sure that you include both because any of the below options can be used depending on the situation:

Remove or fix the bad file in a new commit and then push it to the remote repository. This is the most obvious way to fix an error. Once you have made necessary changes to the file, then commit it to the remote repository using the command: git commit -m “commit message”

Also, you can create a new commit that undoes all changes that were made in the bad commit. To do this use the command

git revert <name of bad commit>

**11. What is the difference between git pull and git fetch?**

**Answer:** Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

Git fetch is also used for the same purpose but it works in a slightly different way. When you perform a git fetch, it pulls all new commits from the desired branch and stores it in a new branch in your local repository. If you want to reflect these changes in your target branch, git fetch must be followed with a git merge. Your target branch will only be updated after merging the target branch and fetched branch. Just to make it easy for you, remember the equation below:

Git pull = git fetch + git merge

**12. What is git stash?**

**Answer:**Often, when you’ve been working on part of your project, things are in a messy state and you want to switch branches for some time to work on something else. The problem is, you don’t want to do a commit of half-done work just so you can get back to this point later. The answer to this issue is Git stash.

Stashing takes your working directory that is, your modified tracked files and staged changes and saves it on a stack of unfinished changes that you can reapply at any time.

$ git status

  modified:   index.php

  modified:   css/styles.css

Apply Git Stash

$ git stash

Saved working directory and index state WIP on master:

   2dfe283 Implement the new login box

HEAD is now at 2dfe283 Implement the new login box

Git's Stash is meant as a temporary storage. When you're ready to continue where you left off, you can restore the saved state easily:

$ git stash pop

**13. What is the function of ‘git stash apply’?**

**Answer:** If you want to continue working where you had left your work then ‘git stash apply‘command is used to bring back the saved changes onto your current working directory.

git stash apply n

To get list of stashes:

git stash list

**14. What is the function of ‘git config’?**

**Answer:** Git uses your username to associate commits with an identity. The git config command can be used to change your Git configuration, including your username.

Now explain with an example.

Suppose you want to give a username and email id to associate a commit with an identity so that you can know who has made a particular commit. For that I will use:

git config –global user.name “Your Name”: This command will add a username.

git config –global user.email “Your E-mail Address”: This command will add an email id.

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

**Answer:**  To know if a branch has been merged into master or not you can use the below commands:

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

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

**16. Can you explain the Gitflow workflow?**

**Answer:**To record the history of the project, Gitflow workflow employs two parallel long-running branches – master and develop:

Master – this branch is always ready to be released on LIVE, with everything fully tested and approved (production-ready).

Hotfix – these branches are used to quickly patch production releases. These branches are a lot like release branches and feature branches except they’re based on master instead of develop.

Develop – this is the branch to which all feature branches are merged and where all tests are performed. Only when everything’s been thoroughly checked and fixed it can be merged to the master.

Feature – each new feature should reside in its own branch, which can be pushed to the develop branch as their parent one.