**new to learn git cherry-pick command**

**git pull = git fetch + merge**

**Create a Repository**

Create a new local repository

$ git init [project name]

Clone a repository

$ git clone git\_url

Clone a repository into a specified directory

$ git clone git\_url my\_directory

**Make a change**

Show modified files in working directory, staged for your next commit

$ git status

Stages the file, ready for commit

$ git add [file]

Stage all changed files, ready for commit

$ git add .

Commit all staged files to versioned history

$ git commit -m "commit message"

Commit all your tracked files to versioned history

$ git commit -am "commit message"

Discard changes in working directory which is not staged

$ git restore [file]

Unstage a stagged file or file which is staged

$ git restore --staged [file]

Unstages file, keeping the file changes

$ git reset [file]

Revert everything to the last commit

$ git reset --hard

Diff of what is changed but not staged

$ git diff

Diff of what is staged but not yet commited

$ git diff --staged

Apply any commits of current branch ahead of specified one

$ git rebase [branch]

**Configuration:**

Set the name that will be attached to your commits and tags

$ git config --global user.name "name"

Set an email address that will be attached to your commits and tags

$ git config --global user.email "email"

Enable some colorization of Git output

$ git config --global color.ui auto

Edit the global configuration file in a text editor

git config --global –edit

**Observe your Repository:**

Show the commit history for the currently active branch

$ git log

Show the commits on branchA that are not on branchB

$ git log branchB..branchA

Show the commits that changed file, even across renames

$ git log --follow [file]

Show the diff of what is in branchA that is not in branchB

$ git diff branchB...branchA

Show any object in Git in human-readable format

$ git show [SHA]

**Synchronize:**

Fetch down all the branches from that Git remote

$ git fetch [alias]

Merge a remote branch into your current branch to bring it up to date

$ git merge [alias]/[branch]

# No fast-forward

$ git merge --no-ff [alias]/[branch]

# Only fast-forward

$ git merge --ff-only [alias]/[branch]

Transmit local branch commits to the remote repository branch

$ git push [alias] [branch]

Fetch and merge any commits from the tracking remote branch

$ git pull

Merge just one specific commit from another branch to your current branch

$ git cherry-pick [commit\_id]

**Remote:**

Add a git URL as an alias

$ git remote add [alias] [url]

Show the names of the remote repositories you've set up

$ git remote

Show the names and URLs of the remote repositories

$ git remote -v

Remove a remote repository

$ git remote rm [remote repo name]

Change the URL of the git repo

$ git remote set-url origin [git\_url]

**Temporary Commits**

Save modified and staged changes

$ git stash

List stack-order of stashed file changes

$ git stash list

Write working from top of stash stack

$ git stash pop

Discard the changes from top of stash stack

$ git stash drop

**Tracking path Changes**

Delete the file from project and stage the removal for commit

$ git rm [file]

Change an existing file path and stage the move

$ git mv [existing-path] [new-path]

Show all commit logs with indication of any paths that moved

$ git log --stat -M

**Ignoring Files**

/logs/\*

# "!" means don't ignore

!logs/.gitkeep

/# Ignore Mac system files

.DS\_store

# Ignore node\_modules folder

node\_modules

# Ignore SASS config files

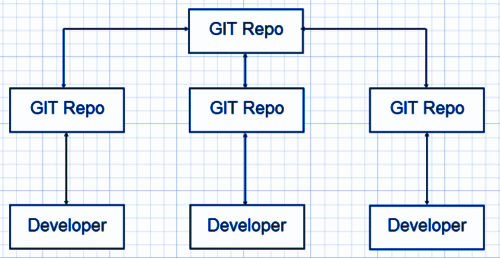
.sass-cache

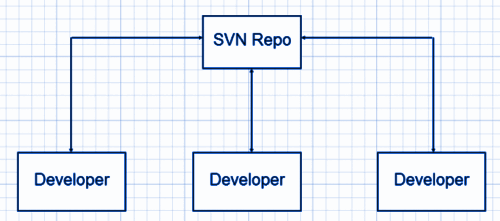
A .gitignore file specifies intentionally untracked files that Git should ignore

What is the meaning of the commands – git status, git log, git diff, git revert <commit>, git reset <file>?

|  |  |
| --- | --- |
| **Command** | **Meaning** |
| git status | Gives a list of which files are staged, unstaged, and untracked |
| git log | Illustrates the entire commit history by using the default format |
| git diff | Displays the unstaged changes between index and working directory |
| git revert <commit> | Undoes all the changes made in <commit> and applies it to the current branch by creating a new commit |
| git reset <file> | Removes <file> from a staging area without overwriting any changes by keeping the working directory unchanged |

**Git Workflow:**





State the difference between “git pull” and “git fetch.”

**Ans.** This is an important Git interview question. “git pull” and “git fetch” are used for downloading new data from a remote repository.

“git fetch – It downloads new data from the repository but does not support integrating this data into working files. It offers a fresh view of things that happened in the remote repository.

“git pull” – This command is used to update the current HEAD data branch with all the changes that occurred in the remote repository. Thus, it downloads the data and integrates it with existing working files.

How do you edit or fix the last commit message in Git?

Ans. If you forget to add anything in the commit message or committed a typo error, you can rectify it by using the –amend flag command.

$ git commit –amend -m “Sorry I missed an important update”

Note: –amend flag will only help in editing or fixing the last commit message.

How can you change any older commit messages?

Ans. To change an older commit the command is –

$ git rebase –interactive

What do you mean by the Gitflow workflow?

Ans. This is an important Git interview question. The Gitflow Workflow specifies a branching model for Git. It provides a framework for managing large projects and is mostly used for projects that have a scheduled release cycle. Gitflow assigns very specific roles to different branches and defines how and when they should interact:

Master: This branch is always ready to be released on LIVE. It releases when everything is fully tested and approved.

Develop: All feature branches are merged into this branch and all tests also are performed here. When everything is thoroughly checked, it can be merged into the master.

Feature: Each new feature should reside in its own branch, which can be pushed to develop branch as their parent branch.

Hotfix: These branches are used to quickly patch production releases. They are based on master instead of develop.

What is the difference between git remote and git clone?

Ans. With git remote, you can create, view, and delete connections to other repositories. It’s used to refer to a remote repository or a central repository.

|  |  |
| --- | --- |
| **git remote** | **git clone** |
| Allows you to create, view, and delete connections to other repositories. | Enables you to create a clone or copy of the target repository. |
| Targets a remote repository or a central repository. | It targets a different already existing repository. |

How will you find out if a branch has already been merged or not?

Ans. We use the following commands to find out if a branch has already been merged or not:

git branch –merged master – it will list all the branches that have been renamed into master.

git branch –merged – it lists the branches that have been merged into the current branch (HEAD).

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

What is the difference between revert and reset?

|  |  |
| --- | --- |
| **Revert** | **Reset** |
| It creates a new commit that undoes the changes made in the previous commit. | It undoes the local changes that have been made to a Git repository. |
| New history is added to the project and the existing history is not modified. | This command may alter existing history. The Reset command operates on the commit history, the staging index, and the working directory. |
| Command: git revert | Command: git reset |

What is git stash

When you are working in your working directory and done few changes so few items in staging, and now if you have to provide any hot fix based on the last commit on the remote repository then to do it the best way is save the current state of the working directory and the index/staging (something like on a stack), for that we use command 'git stash' or 'git stash save'. After using this command, the current working directory will be having the same match as on the remote last commit means locally no changes exist to stage or commit. Now you can do your stuff on the clean copy of the remote; git stash will not cover git ignored files and the newly created files but not 'staged' yet in current working directory.