## What is Git?

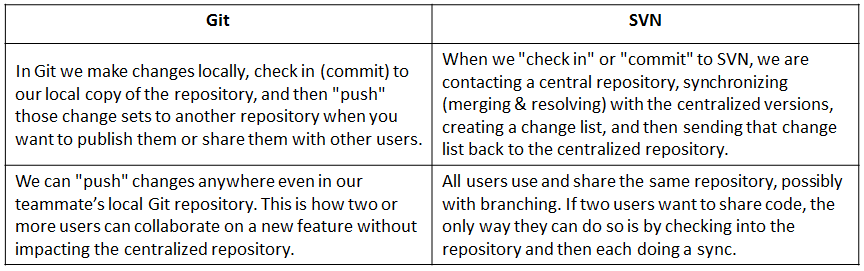
Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.

<https://www.atlassian.com/git/tutorials/what-is-git>

<https://www.visualstudio.com/learn/what-is-git/>

## What is the difference between Git and SVN?

The proper answer for this according to me will be the architectural differences between Git and SVN. So, the basic difference is that Git is distributed and SVN is centralized version control system.



Git is a distributed VCS; SVN is a non-distributed VCS.

Git has a centralized server and repository; SVN does not have a centralized server or repository.

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 is distributed under GNU, and its maintenance overseen by Junio Hamano; Apache Subversion, or SVN, is distributed under the open source license.

<http://www.differencebetween.net/technology/software-technology/difference-between-git-and-svn/>

<https://help.github.com/articles/what-are-the-differences-between-subversion-and-git/>

## What is the command to write a commit message in Git?

Command that is used to write a commit message is “git commit -a”.

Now explain about -a flag by saying -a on the command line instructs git to commit the new content of all tracked files that have been modified. Also mention you can use “git add<file>” before git commit -a if new files need to be committed for the first time.

## What is ‘bare repository’ in Git?

What is the difference between a repository created using the git init command and the git init --bare command?

Repositories created with the git init command are called working directories. In the top-level folder of the repository you will find two things:

* A .git subfolder with all the git related revision history of your repo
* A working tree, or checked out copies of your project files.

Repositories created with git init --bare are called bare repos. They are structured a bit differently from working directories. First off, they contain no working or checked out copy of your source files. And second, bare repos store git revision history of your repo in the root folder of your repository instead of in a .git subfolder.

Note : bare repositories are customarily given a .git extension.

<http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/>

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

There can be two answers to 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 push it to the remote repository. This is the most natural way to fix an error. Once you have made necessary changes to the file, commit it to the remote repository for that I will use

git commit -m “commit message”

* Create a new commit that undoes all changes that were made in the bad commit.to do this I will use a command

git revert <name of bad commit>

## What is the difference between git pull and git fetch?

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

<https://www.quora.com/Whats-the-difference-between-git-pulland-git-fetch>

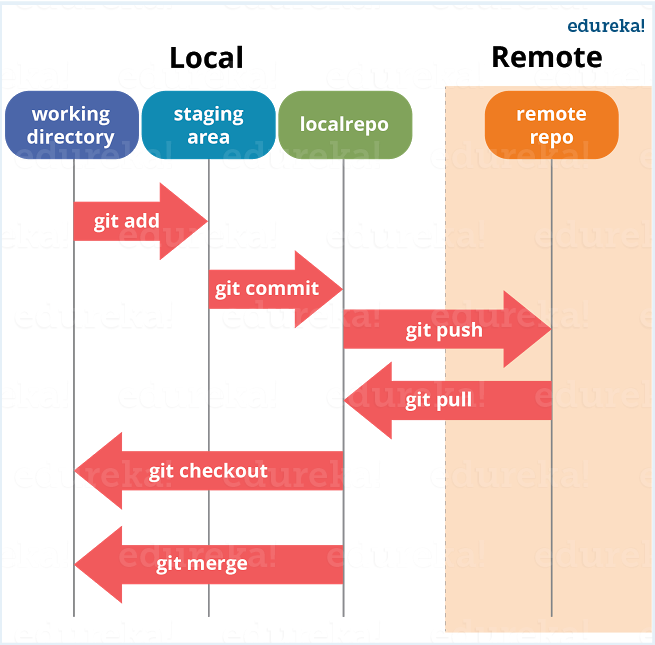
<https://longair.net/blog/2009/04/16/git-fetch-and-merge/>

<https://blog.mikepearce.net/2010/05/18/the-difference-between-git-pull-git-fetch-and-git-clone-and-git-rebase/>

## What is ‘staging area’ or ‘index’ in Git?

The staging area (aka index) is a container where git collects all changes which will be part of the next commit.

If you are editing a versioned file on your local machine, git recognizes that your file is modified - but it will not be automatically part of your next commit and is therfore unstaged. Staging the file will put the file into the staging area (index). The next git commit will transfer all items from staging are into your repository.



<https://softwareengineering.stackexchange.com/questions/119782/what-does-stage-mean-in-git>

<http://gitolite.com/uses-of-index.html>

## What is Git stash?

Often, when you’ve been working on part of your project, things are in a messy state and you want to switch branches for a bit 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 the git stash command.

Stashing takes the dirty state of 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.

<https://git-scm.com/book/en/v1/Git-Tools-Stashing>

<https://www.tutorialspoint.com/git/git_stash_operation.htm>

## What is Git stash drop?

Git ‘stash drop’ command is used to remove the stashed item. It will remove the last added stash item by default, and it can also remove a specific item if you include it as an argument.

If you want to remove a particular stash item from the list of stashed items you can use the below commands:

git stash list: It will display the list of stashed items like:

stash@{0}: WIP on master: 049d078 added the index file

stash@{1}: WIP on master: c264051 Revert “added file\_size”

stash@{2}: WIP on master: 21d80a5 added number to log

If you want to remove an item named stash@{0} use command git stash drop stash@{0}.

<https://www.edureka.co/blog/interview-questions/git-interview-questions/>

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

To get a list files that has changed in a particular commit use the below command:

git diff-tree -r {hash}

Given the commit hash, this will list all the files that were changed or added in that commit. The -r flag makes the command list individual files, rather than collapsing them into root directory names only. The output will also include some extra information, which can be easily suppressed by including two flags:

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

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