**Git log**

The advantage of a version control system is that it records changes. These records allow us to retrieve the data like commits, figuring out bugs, updates. But, all of this history will be useless if we cannot navigate it. At this point, we need the git log command.

Git log is a utility tool to review and read a history of everything that happens to a repository. Multiple options can be used with a git log to make history more specific.

Generally, the git log is a record of commits. A git log contains the following data:

* A commit hash, which is a 40-character checksum data generated by SHA (Secure Hash Algorithm) algorithm. It is a unique number.
* Commit Author metadata: The information of authors such as author name and email.
* Commit Date metadata: It's a date timestamp for the time of the commit.
* Commit title/message: It is the overview of the commit given in the commit message.

**Basic Git log**

Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as:

$ git log

The above command is listing all the recent commits. Each commit contains some unique sha-id, which is generated by the SHA algorithm. It also includes the date, time, author, and some additional details.

**Git Log Stat**

The log command displays the files that have been modified. It also shows the number of lines and a summary line of the total records that have been updated.

Generally, we can say that the stat option is used to display

* the modified files,
* The number of lines that have been added or removed
* A summary line of the total number of records changed
* The lines that have been added or removed.

It will be used as follows:

$ git log --stat

Git log P or Patch

The git log patch command displays the files that have been modified. It also shows the location of the added, removed, and updated lines.

It will be used as:

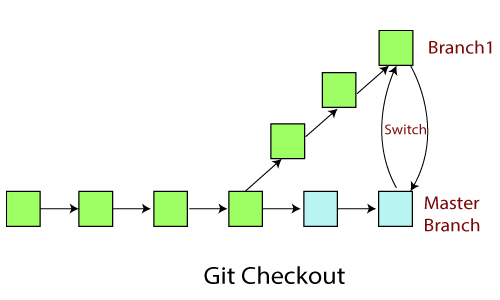
$ git log --patch **Or** $ git log -p

Generally, we can say that the --patch flag is used to display:

* Modified files
* The location of the lines that you added or removed
* Specific changes that have been made.

**Git Checkout**

In Git, the term checkout is used for the act of switching between different versions of a target entity. The git checkout command is used to switch between branches in a repository. Be careful with your staged files and commits when switching between branches.



The git checkout command operates upon three different entities which are files, commits, and branches. Sometimes this command can be dangerous because there is no undo option available on this command.

It checks the branches and updates the files in the working directory to match the version already available in that branch, and it forwards the updates to Git to save all new commit in that branch.

**Operations on Git Checkout**

We can perform many operations by git checkout command like the switch to a specific branch, create a new branch, checkout a remote branch, and more. The git branch and git checkout commands can be integrated.

**Checkout Branch**

To switch between branches, use the below command.

$ git checkout <branchname>

Create and Switch Branch

The git checkout commands let you create and switch to a new branch. You can not only create a new branch but also switch it simultaneously by a single command. The git checkout -b option is a convenience flag that performs run git branch <new-branch>operation before running git checkout <new-branch>.

$ git checkout -b <branchname>

Checkout Remote Branch

Git allows you to check out a remote branch by git checkout command. It is a way for a programmer to access the work of a colleague or collaborator for review and collaboration. Each remote repository contains its own set of branches. So, to check out a remote branch, you have first to fetch the contents of the branch.

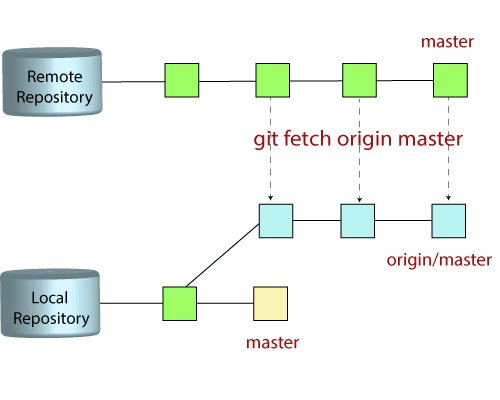
$ git checkout <remotebranch>

Collaborating in git git refers to sharing on code among several developers. We will look at code sharing is handled in Git.We will discuss major collaboration operations in Git.  Our main focus will be on Git fetch, Git pull and Git Push.

**Git Fetch**

Git "fetch" Downloads commits, objects and refs from another repository. It fetches branches and tags from one or more repositories. It holds repositories along with the objects that are necessary to complete their histories to keep updated remote-tracking branches.

The "git fetch" command is used to pull the updates from remote-tracking branches. Additionally, we can get the updates that have been pushed to our remote branches to our local machines. As we know, a branch is a variation of our repositories main code, so the remote-tracking branches are branches that have been set up to pull and push from remote repository.



We can fetch the complete repository with the help of fetch command from a repository URL like a pull command does. See the below output:

$ git fetch< repository Url>

The git command that allows to fetch a specific branch from a repository. It will only access the element from a specific branch.

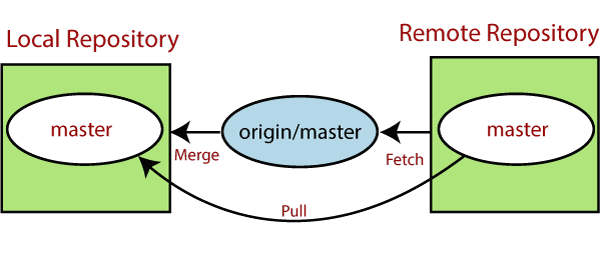
$ git fetch <branch URL><branch name>

The git fetch command allows to fetch all branches simultaneously from a remote repository.

$ git fetch -all

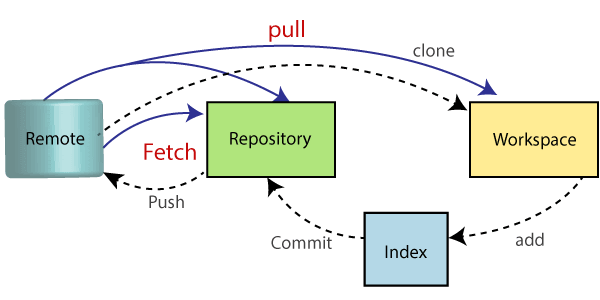
**Git Pull / Pull Request**

The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The git pull command is used to pull a repository.



Pull request is a process for a developer to notify team members that they have completed a feature. Once their feature branch is ready, the developer files a pull request via their remote server account. Pull request announces all the team members that they need to review the code and merge it into the master branch.

The below figure demonstrates how pull acts between different locations and how it is similar or dissimilar to other related commands.



**The "git pull" command**

The pull command is used to access the changes (commits)from a remote repository to the local repository. It updates the local branches with the remote-tracking branches. Remote tracking branches are branches that have been set up to push and pull from the remote repository. Generally, it is a collection of the fetch and merges command. First, it fetches the changes from remote and combined them with the local repository.

The git pull command is syntax is:

$ git pull <option> [<repository URL><refspec>...]

In which:

<option>: Options are the commands; these commands are used as an additional option in a particular command. Options can be -q (quiet), -v (verbose), -e(edit) and more.

<repository URL>: Repository URL is your remote repository's URL where you have stored your original repositories like GitHub or any other git service.

**Git Pull Remote Branch**

Git allows fetching a particular branch. Fetching a remote branch is a similar process, as mentioned above, in git pull command.

$ git pull <remote branch URL>

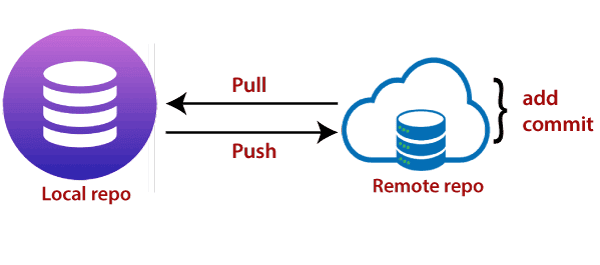
**Git Pull Origin Master**

There is another way to pull the repository. We can pull the repository by using the git pull command. The syntax is given below:

$ git pull origin master

**Git Push**

The push term refers to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.



Moreover, we can say the push updates the remote refs with local refs. Every time you push into the repository, it is updated with some interesting changes that you made. If we do not specify the location of a repository, then it will push to default location at origin master.

The "git push" command is used to push into the repository. The push command can be considered as a tool to transfer commits between local and remote repositories. The basic syntax is given below:

$ git push <option> [<Remote URL><branch name><refspec>...]

Push command supports many additional options. Some options are as follows under push tags.

**Git Push Tags**

<repository>: The repository is the destination of a push operation. It can be either a URL or the name of a remote repository.

<refspec>: It specifies the destination ref to update source object.

--all: The word "all" stands for all branches. It pushes all branches.

--prune: It removes the remote branches that do not have a local counterpart. Means, if you have a remote branch say demo, if this branch does not exist locally, then it will be removed.

--mirror: It is used to mirror the repository to the remote. Updated or Newly created local refs will be pushed to the remote end. It can be force updated on the remote end. The deleted refs will be removed from the remote end.

--dry-run: Dry run tests the commands. It does all this except originally update the repository.

--tags: It pushes all local tags.

--delete: It deletes the specified branch.

-u: It creates an upstream tracking connection. It is very useful if you are going to push the branch for the first time.

Git Push Origin Master

Git push origin master is a special command-line utility that specifies the remote branch and directory. When you have multiple branches and directory, then this command assists you in determining your main branch and repository.

Generally, the term origin stands for the remote repository, and master is considered as the main branch. So, the entire statement "git push origin master" pushed the local content on the master branch of the remote location.

$ git push origin master

**Git Force Push**

The git force push allows you to push local repository to remote without dealing with conflicts. It is used as follows:

$ git push <remote><branch> -f **OR** $ git push <remote><branch> -force

How to Safe Force Push Repository:

There are several consequences of force pushing a repository like it may replace the work you want to keep. Force pushing with a lease option is capable of making fail to push if there are new commits on the remote that you didn't expect. If we say in terms of git, then we can say it will make it fail if remote contains untracked commit. It can be executed as:

$git push <remote><branch> --force-with-lease

Delete a Remote Branch

We can delete a remote branch using git push. It allows removing a remote branch from the command line. To delete a remote branch, perform below command:

$ git push origin -delete edited