

Introduction to Git and GitHub Cheat Sheet

diff

`diff` is used to find differences between two files.

```
diff file1.ext file2.ext > file.diff
```

This could be the difference between your working environment and your staging area (`git diff` by itself), between your staging area and your last commit (`git diff --staged`), or between two commits (`git diff master branchB`)

diff -u

`diff -u` is used to compare two files, *line by line*, and have the differing lines compared side-by-side in the same output.

Patch

Patch is useful for applying file differences. See the below example, which compares two files. The comparison is saved as a `.diff` file, which is then patched to the original file!

```
patch my_file < file.diff
```

git add

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit.

git add -p

It separates all changes into hunks and lets you interactively choose which ones you want to stage and which ones you want to keep in the working directory.

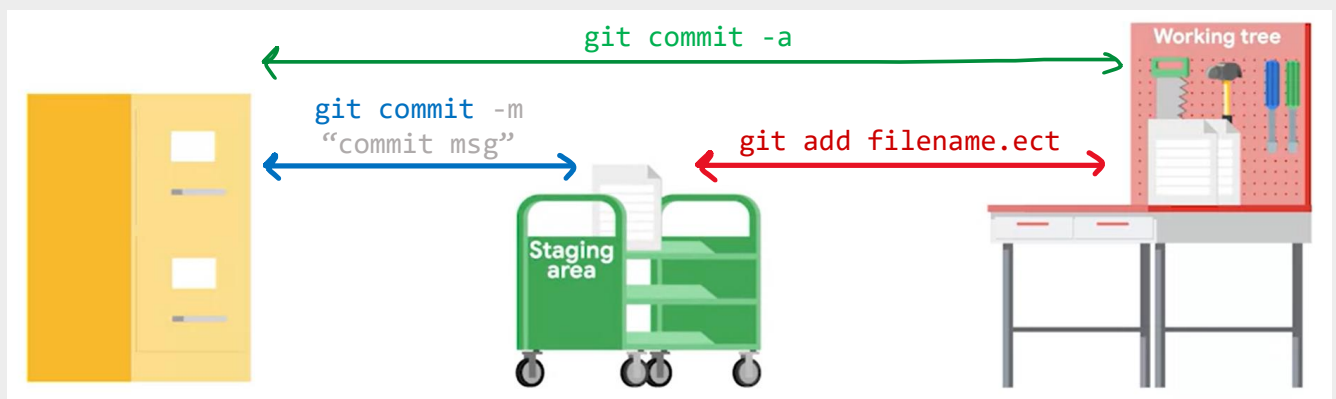
git commit

The `git commit` command takes all the file contents that have been staged with `git add` and records a new permanent snapshot in the database and then moves the branch pointer on the current branch up to it.

git commit -a

`git commit -a` is a shortcut to stage any changes to tracked files and commit them *in one step*.

- If the modified file has never been committed to the repo; we'll still need to use `git add` to track it first.



git log

`git log` shows us the list of commits made in the current Git repository.

git log -p

To look at the actual lines that changed in each commit we use `git log -p`. The `p` comes from ^{flag} patch, because using this flag gives us the patch that was created.

git log -number

e.g., `git log -2`: the -2 perimeter limits the output to the last two entries.

git log --stat

`git log --stat` this will cause git log to show some stats about the changes in the commit.

git log --graph --oneline

The `git log --graph` command creates a graphic overview of how a developer's various development pipelines have branched and merged over time. The `oneline` option is used to display the output as one commit per line.

git show

`git show commit-id` will give us the information about a specific commit.

git rm

`git rm` will stop the file from being tracked by git and remove it from the git directory.

git mv

`git mv` moves or renames a file, a directory, or a symlink.

git branch

Running `git branch` by itself will show you a list of all the branches in your repository.

`git branch new-branch-name` creates a new branch named `new-branch-name`.

git branch -d

used to delete a git branch from the local machine.

git branch -D

Shortcut for `--delete --force`.

⇒ `git branch -d` is for deleting branches that are fully merged in its upstream branch or to HEAD if you don't have an upstream for your branch. If the branch is not fully merged it will not perform the deletion. But `git branch -D` deletes the branch even if it's not merged.

git branch -r

Lists remote branches; can be combined with other branch arguments to manage remote branches like: lists or deletes (if used with `-d`) the remote-tracking branches. Combines with `--list` to match the optional pattern(s).

git checkout

`git checkout branch-name` to check out the latest snapshot for both files and for **branches**.
(We use it with branches to move to another branch)

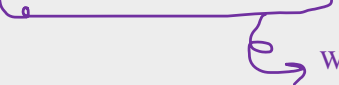
git checkout -b

`git checkout -b branch-name`: Git created the new branch and switched to it in just one command.

git merge

A typical workflow for managing branches in Git, is to create a separate branch for developing any new features or changes. Once the new features in good shape, *we merge the separate branch back into the main trunk of code*. Merging is the term that Git uses for combining branch data and history together.

Syntax: `git merge branch-name` to merge the `branch-name` branch into the master branch.



We write it in the master branch to merge all the branches with the master.

git merge --abort

`git merge --abort` command is an escape hatch. This will stop the merge and reset the files in your working tree back to the previous commit before the merge ever happened.

git revert

`git revert` reverts some existing commits. Given one or more existing commits, revert the changes that the related patches introduce, and record some new commits that record them.

Note: `git revert` is used to record some new commits to reverse the effect of some earlier commits (often only a faulty one).

git clone

Using the `git clone` command followed by the URL of the repo, we create a local copy of the repository.

git push

We can send our changes to that remote repository by using the `git push` command which will gather all the snapshots we've taken and send them to the remote repository.

git pull

we use? `Git pull` to update our local repository to reflect changes made in the remote repo.

git remote

`git remote` command manages set of tracked repositories. It manages the set of repositories ("remotes") whose branches we track.

`git remote -v` (in the directory of the repo) to see the URLs associated with the origin remote.

When we call a `git clone` to get a local copy of a remote repository, Git sets up that remote repository with the default origin name.

```
97059@Haya-PC MINGW64 ~/testing-repo (main)
```

```
$ git remote -v
```

```
origin https://github.com/HayaAbuRaed/testing-repo.git (fetch)
```

```
origin https://github.com/HayaAbuRaed/testing-repo.git (push)
```

used to fetch data from the remote repository

used to push data to that remote repo

git remote show <name>

Gives some information about the remote <name>.

git fetch

To sync the data, we use the `git fetch` command. This command copies the commits done in the remote repository to the remote branches, so we can see what other people have committed.

⇒ `git fetch` fetches remote updates but doesn't merge; `git pull` fetches remote updates and merges.

git remote update

Fetch updates for remotes or remote groups in the repository as defined by remotes.