Git Tutorial

Tags	Tools
Created by	
Date	@September 30, 2022

A tutorial introduction to Git

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2500c4c7-62b4-4e3e-b983-7406fd252599/git-cheat-sheet-education.pdf

Introduction:

Git is free and open source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Configure git:

It is a good idea to introduce yourself to Git with your name and public email address before doing any operation.

```
git config --global user.name "your_name"
git config --global user.email "sample@gmail.com"
```

Open the terminal in your project and execute the below command

```
git init
```

You’ve now initialized the working directory—you may notice a new directory created, named ".git".

Next, tell Git to take a snapshot of the contents of all files under the current directory

```
git add .
```

This snapshot is now stored in a temporary staging area which Git calls the "index". You can permanently store the contents of the index using the below command.

```
git commit -m "commit message"
```

Adding specific files to the staging

```
git add file_name
```

Adding multiple files to stagging

```
git add file_name1 file_name2
```

To list changes that you’ve made but not yet added in stagging or committed

```
git status
```

Alternatively, instead of running ***git add***
beforehand, you can use

```
git commit -a
```

which will automatically notice any modified (but not new) files, add them to the index, and commit, all in one step.

A note on commit messages: Though not required, it’s a good idea to begin the commit message with a single short (less than 50 characters) line summarizing the change, followed by a blank line and then a more thorough description.

Viewing project history

```
git log
```

If you also want to see complete diffs at each step, use

```
git log -p
```

To list the overview of the change

```
git log --stat --summary
```

Managing branches

A single Git repository can maintain multiple branches of development. To create a new branch named "branch_name", use

```
git branch branch_name
```

To list all branches

```
git branch
```

Output: Default branch name “master” and newly created branch will be visible

To start working on another branch you can switch branches using the below command

```
git switch branch_name
```

Now make any changes in your repo, add changes to staging, and finally commit your changes then move to the default master branch

Now you will observe that changes made by you will no longer be available in the default “master” branch

To bring changes into the master branch you have to merge your created branch with the master using the below command

```
git merge branch_name
```

To see all commit history using nice graph representation

```
git log --graph
```

To delete the branch

```
git branch -d branch_name
```

Note: This command ensures that the changes in the experimental branch are already in the current branch.

If you have experimented with something in a branch and you don’t want to include those changes in the main branch then

```
git branch -D branch_name
```

Using Git for collaboration

Clone a repo

```
git clone repo_url
```

Now change your directory to the newly cloned repo

```
cd path_to_cloned_repo
```

Now make any changes and commit it

```
git commit -a -m "commit_message"
```

Push a repo

```
git push origin "branch_name"
```

If more than one person is working in same repo then you might get error during push because other developer might have made changes in github repo which is not available in your local repo

to fix that issue you can use **git fetch** or **git pull** command

git pull is equivalent git fetch and git merge

git pull bring changes from remote repo into local and merge the change into your local repo

```
git pull
```

OR

git fetch bring changes from remote repo and allow us to review and resolve any conflict and after that we can manually commit the new changes

```
git fetch
git merge
```

Now you can push your changes to github

How to contribute to open source or someone else repo

Fork repo

Fork a repo - GitHub Docs

Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting point for your own

https://docs.github.com/en/get-started/quickstart/fork-a-repo

Creating Pull request

Creating a pull request from a fork - GitHub Docs

If your pull request compares your topic branch with a branch in the upstream repository as the base branch, then your topic branch is also called the compare branch of the pull request. For

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request-from-a-fork

Github

Create a repo - GitHub Docs

You can store a variety of projects in GitHub repositories, including open source projects. With open source projects, you can share code to make better, more reliable software. You can

https://docs.github.com/en/get-started/quickstart/create-a-repo

Advance Github - Workflows

A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run

when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.

Workflows are defined in the `.github/workflows` directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have one workflow to build and test pull requests, another workflow to deploy your application every time a release is created, and still another workflow that adds a label every time someone opens a new issue