Overview of GIT

What is version control?

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

A Version Control System allows you to revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also means that if you screw things up or lose files, you can generally recover easily.

Git basics

Git is a free and open source version control system, originally created by Linus Torvalds in 2005. Unlike older centralized version control systems such as SVN and CVS, Git is distributed: every developer has the full history of their code repository locally. Git helps you keep track of the changes you make to your code. It is basically the history tab for your code editor. If at any point while coding you hit a fatal error and don't know what's causing it you can always revert back to the stable state. So it is very helpful for debugging. Or you can simply see what changes you made to your code over time.

Git also has excellent support for branching, merging, and rewriting repository history, which has led to many innovative and powerful workflows and tools. Pull requests are one such popular tool that allow teams to collaborate on Git branches and efficiently review each other's code. Git is the most widely used version control system in the world today and is considered the modern standard for software development.

What is a Repository ?

A repository / repo is nothing but a collection of source code

How Git works

 # Make sure you have Git installed on your machine.

If you are on a **Mac**, fire up the terminal and enter the following command:

```
$ git --version
```

 # Tell Git who you are.

Introduce yourself. Mention your Git username and email address, since every Git commit will use this information to identify you as the author.

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

```
$ git config --global user.email "im_satoshi@musk.com"
```

```
$ git config --global --list # To check the info you just provided
```

 # Clone a git repo.

```
$ git clone https://github.com/example/example.git
```

 # Add files to the Staging Area for commit:

Now to add the files to the git repository for commit:

```
$ git add .  # Adds all the files in the local repository and stages them for commit. OR if you want to add a specific file
```

```
$ git add README.md # To add a specific file
```

 # Before we commit let's see what files are staged:

```
$ git status # Lists all new or modified files to be committed
```

 # Commit Changes you made to your Git Repo:
Now to commit files you added to your git repo:

```
$ git commit -m "First commit"# The message in the " "
```

 # Uncommit Changes you just made to your Git Repo:
Now suppose you just made some error in your code or placed an unwanted file inside the repository, you can unstage the files you just added using:

```
$ git reset --hard HEAD~1# Remove the most recent commit# Commit again!
```

 # Add a remote origin and Push:
Now each time you make changes in your files and save it, it won't be automatically updated on GitHub. All the changes we made in the file are updated in the local repository. Now to update the changes to the master:

```
$ git remote add origin remote_repository_URL# sets the new remote
```

The **git remote** command lets you create, view, and delete connections to other repositories.

```
$ git remote -v# List the remote connections you have to other repositories.
```

The **git remote -v** command lists the URLs of the remote connections you have to other repositories.

 # Push Code

```
$ git push -u origin master # pushes changes to origin
```

Now the **git push** command pushes the changes in your local repository up to the remote repository you specified as the origin.

 # View Commit History:
You can use the **git log** command to see the history of commit you made to your files:

```
$ git log
```

 # Pull update from remote:
So to make sure those changes are reflected on my local copy of the repo:

```
$ git pull origin master
```

 # Pull update from remote:
So to make sure those changes are reflected on my local copy of the repo:

```
$ git pull origin master
```

 # Create Branch

```
$ git branch branch_name
```

 # Checkout to another Branch

```
$ git checkout branch_name
```

 # Merge two Branches

```
$ git merge branch_name
```

One More Thing:

`.gitignore`

`.gitignore` tells git which files (or patterns) it should ignore. It's usually used to avoid committing transient files from your working directory that aren't useful to other collaborators, such as compilation products, temporary files IDEs create, etc.