# Git Tutorial and Basic Commands

**What is Version Control?**

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. So ideally, we can place any file in the computer on version control.

**Here's Why:**

A Version Control System (VCS) 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. And sometimes you just want to know "who wrote this crap", and having access to that information is worthwhile ?

**So What is Git?**

**Git** is a **version-control system for tracking changes** in computer files and coordinating work on those files among multiple people. Git is a ***Distributed Version Control System***. So Git does not necessarily rely on a central server to store all the versions of a project's files. Instead, every user "clones" a copy of a repository (a collection of files) and has the *full* history of the project on their own hard drive. This clone has *all* of the metadata of the original while the original itself is stored on a self-hosted server or a third party hosting service like GitHub.

**Git** helps you ***keep track of the changes*** you make to your code. It is basically the history tab for your code editor(With no incognito mode ?). 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 Workflow:**

Before we start working with Git commands, it is necessary that you understand what it represents.

**What is a Repository ?**

A repository a.k.a. repo is nothing but a collection of source code.

**There are four fundamental elements in the Git Workflow:**

- **Working Directory**
- **Staging Area**
- **Local Repository**
- **Remote Repository.**

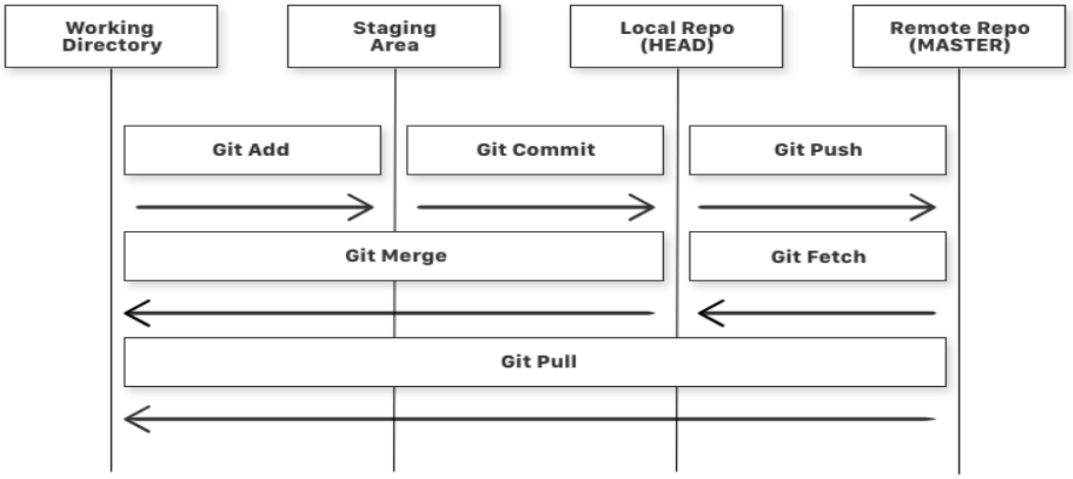


Diagram of a simple Git Workflow

**If you consider a file in your Working Directory, it can be in three possible states.**

1. **It can be staged.** Which means the files with the updated changes are marked to be committed to the local repository but not yet committed.
2. **It can be modified.** Which means the files with the updated changes are not yet stored in the local repository.
3. **It can be committed.** Which means that the changes you made to your file are safely stored in the local repository.

- **git add** is a command used to add a file that is in the working directory to the staging area.
- **git commit** is a command used to add all files that are staged to the local repository.

- **git push** is a command used to add all committed files in the local repository to the remote repository. So in the remote repository, all files and changes will be visible to anyone with access to the remote repository.
- **git fetch** is a command used to get files from the remote repository to the local repository but not into the working directory.
- **git merge** is a command used to get the files from the local repository into the working directory.
- **git pull** is command used to get files from the remote repository directly into the working directory. It is equivalent to a **git fetch** and a **git merge .**

## Now that we know what Git is and it's basic terminologies, let's see how we can place a file under git.

**Step 0: Make a GitHub Account. Duh.**
If you don't already have one, you can make one here.

**Step 1: Make sure you have Git installed on your machine.**
If you are on a Windows, fire up the terminal and enter the following command:

```
$ git --version
```

**Step 2: Tell Git who you are.**
Introduce yourself. Slide in. Seriously, 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
```

**Step 4: Let's Git**
Create a new repository on GitHub. Follow this link.

Now, locate to the folder you want to place under git in your terminal.

```
$ cd Desktop/File_Name
```

**Initialize Git:**
And to place it under git, enter:

```
$ git init          # Initiates an empty git repository
```

**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 " " is given so that the other users can read the message and see what changes you made
```

**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 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     #lists 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.

```
$ 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
```