Git Introduction

Git Introduction

What is Git ?

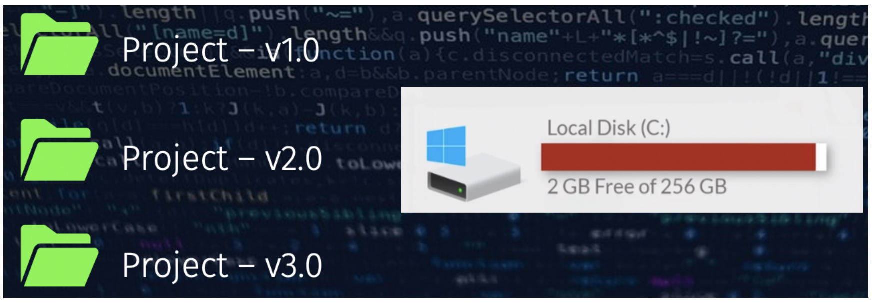
- Git is an distributed version control (source control) system.
- What is version control by the way?

  System that records changes of your file/project and then able to recall any specific version of project at later point in time.

  In other words – version control mechanism allows us to go back in time and get previous state of your project, also you can compare the changes over time.

Problem without version control

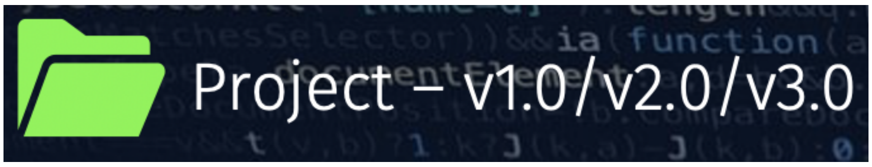
- Lets assume you are working on web-page creation project.
- You have done couple of things like creating [Header & footer etc for your web-page]
- You are going to save this contents into folder as an first version.
- For every new change in web-page, you might create new folder.



Note : Problem with this approach is - it would fill up your drive easily, when the project file grows exponentially.

Version control as an solution

- In order to avoid such painful and complexity in handling files for bigger projects version control system was created.
- Version control system allows you to make snapshot of current state of your project.
- Which means all of your different versions of project is going to be saved in one directory.



History of Git

- Now you might be clear about version control system.
- Git is one of the version control tool like any other.
- Other version control system tools available in market are..

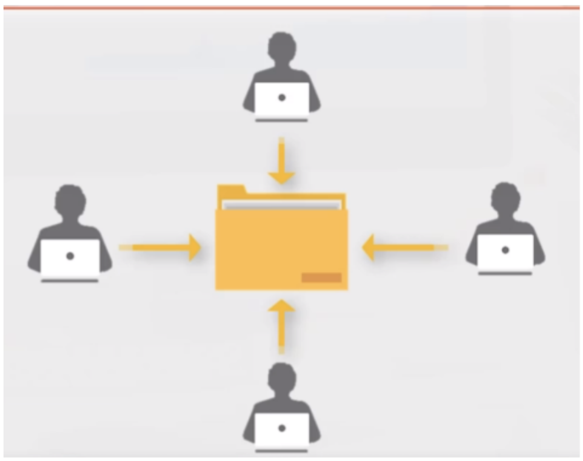
```
Subversion
Beanstalk
Mercurial
Perforce
```

- Git is created by “Linux Torvalds” The same person who created the Linux kernel.
- As like Linux, Git is also an open source tool.

Advantage of using Git

- Collaboration. (Different developers can work on the same project)
- For this we can use some solutions such as

```
GitHub
GitLab
Bitbucket
```



This solutions just provide some shareable place to store your code using Git.

Types of version control system

In General, there are 2 types of version control system.

Centralized

Downside :

- If any network failure you may not be able to reach your project files.
- If central server collapse we may loose the project files.

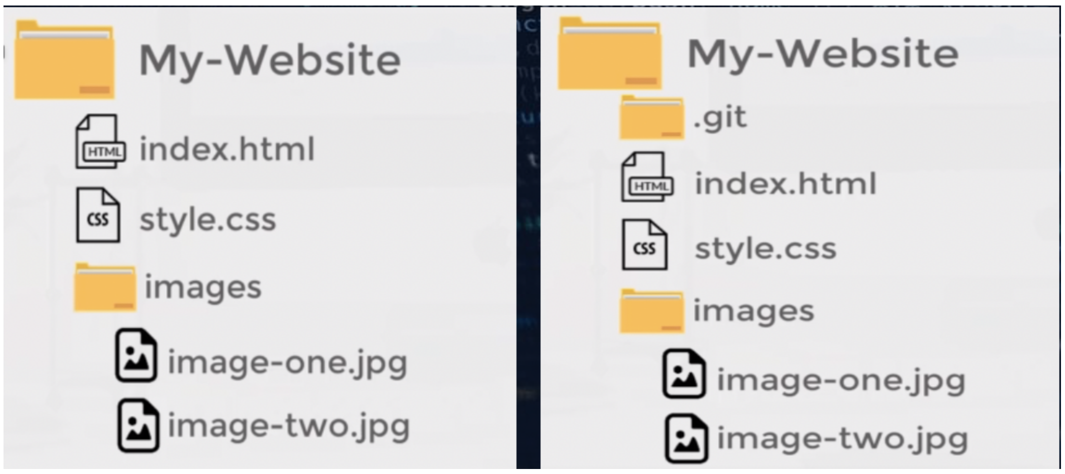
Distributed

- Versions of project files saved locally, in different computers & git server. Hence it has several copies.

How Git works ?

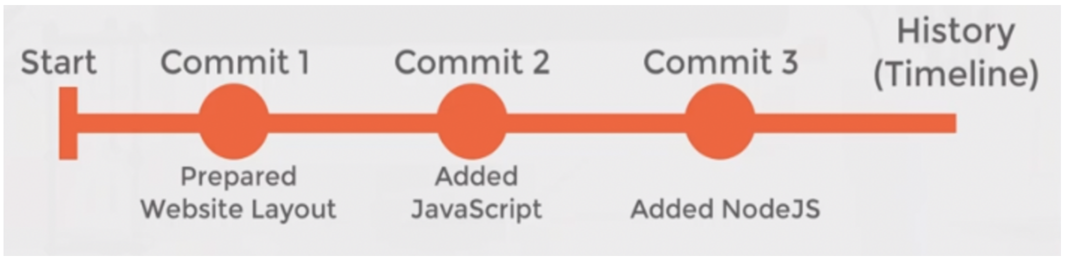
In order to use Git for your project, first of all you may need to initialize it.

- Lets assume you have project folder called My-Website.
- You can do git initialize in that folder which will create folder (.git)
- Once initialization is done the entire folder will be considered as an repository.



Lets see how its going to track all the files under the project folder.

- Commit 1 would be the first version of your project files.
- Commit 2 & Commit 3 is going to have updates files of your project.
- Also you can switch between the versions (also called snapshots) when and where required.



Git Installation

- To commit any file from your machine, you may need to install git first.
- You can visit official git link to download git.
- Git can be installed in Windows/Mac/All Linux Distributions.
- If you are going to use windows – "Git bash" tool would be very helpful to deal with git repositories.
- If you are going to use Linux all things can be done from the command line.

Let's get started with Git

Create Git repo in local machine

- To create a **"git repo"** in local machine, first you may need to create the project folder and copy all the relevant files into that folder.
- Then run command : **"git init"** which is going to create **".git"** folder in project folder. (.git folder may have all the relevant plugins to manage different snapshots of your files)
- Now we have just initialized the git into your project folder.

- Hence all the files available under your project folder may still be untracked, it can be confirmed using command **"git status"**

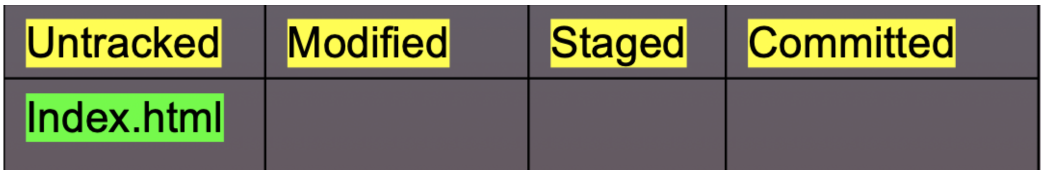
Making your first commit

- In general in your project working directory files can be in any of two states.

  Tracked : Files which are available in last snapshot (commit)

  Untracked : Files which are not available in last snapshot (commit)

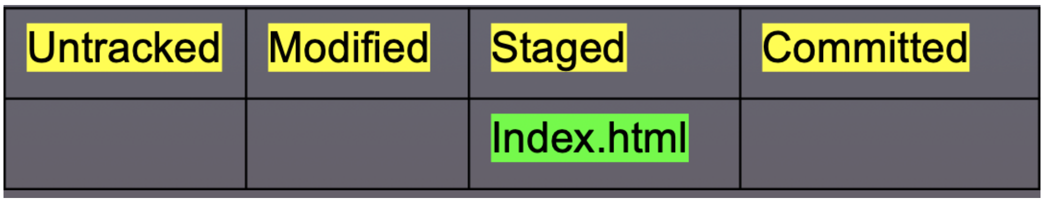
- As mentioned there are several phases in tracking your file.



Untracked	Modified	Staged	Committed
Index.html			

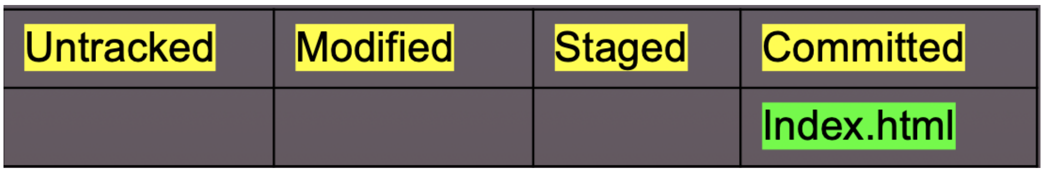
- To commit your files in git first stage it.

```
git add index.html
```



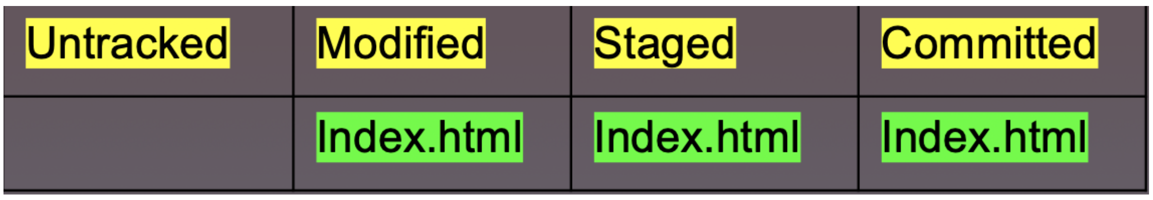
Untracked	Modified	Staged	Committed
		Index.html	

```
git commit –m “Initial commit”
```



Untracked	Modified	Staged	Committed
			Index.html

- If tracked file has been modified. You need to redo the same steps (first stage & then commit)



Untracked	Modified	Staged	Committed
	Index.html	Index.html	Index.html

Commands section

Untracked files – display in red

```
git status
```

This command helps you to stage the file, run git status again to check if files is ready to commit.

```
git add basic.sh
```

This time since the file is tracked – file is displayed in green

```
git status
```

You can also unstage the file if its required.

```
git rm --cached basic.sh
```

After unstage files should display in red again as it is untracked.

```
git status
```

Then commit your file with commit message.

```
git commit –m "Initial commit"
```

This command is going to ask me who I am ? Because git want to keep the track of developer who commits the code.

```
git config --global user.name ”vijay”
```

```
git config --global user.email vijay@gmail.com
```

Using this commands you can check your username & email address.

```
git config --global user.name
```

```
git config --global user.email
```

Final commit again after adding the user name & Email address.

```
git commit –m “Initial commit”
```

Finally running git status command to check everything is fine.

```
git status
```

History

- Let’s see how we can see the history of our files.
- For which we are going to modify the tracked file **(basic.sh)** again & commit.
- Once after tracked file is modified run

This command is going to show you the modified file details.

```
git status
```

We are going to stage the file again before commit using this command.

```
git add basic.sh
```

You can commit file once it is staged.

```
git commit –m “Second commit”
```

Now let’s how to check the history of commits whichever done using command

```
git log
```

You can also see the logs in more precise way with command

```
git log --oneline
```

Git Head

- You can see the “pointer” head from command “git log”
- This actually helps us to understand which commit we are using currently.
- By default head points to the last commit.
- If in-case we go back to any of the previous state of project then head will point to the relevant commit.
- **You can also learn more details about of head using command.**

  ```
  git show HEAD
  ```

- This command is going to provide more visibility about your commit and as well as commit difference (which we can see at later point)
- **Note : You can also view this output by providing head ID or array number.**

  ```
  git show 0d196fa
  ```

  ```
  git show HEAD~1
  ```

Git Restore

- Let’s say for example we are modifying the project file. (Adding some line into it)
- As a result of modification you can see file status as modified in **“git status”** command.
- **Now if in-case you need to restore the change back you can simply use command**

  ```
  git restore filename
  ```

- **If in-case you need to restore multiple file changes you can use commands**

  ```
  “git restore . (or) git restore *
  ```