**GIT :** World’s most popular version control system (VCS).

**VCS :** It is the software which is used to track and manage changes to the file over time.

**https://git-scm.com/**

We need to learn git because 90% projects use it and it is clear winner.

**What exactly GIT do for us?**

1. Track changes across multiple files.
2. Compare version of the project.
3. Time travel back to older versions.
4. Revert back to previous version.
5. Collaborate with other developers and share changes and merge those changes.

**Linus Torvalds developed GIT.**

**Who uses GIT?**

1. Engineers
2. Software Developers.
3. Tech Giants.
4. Governments etc.

**GIT VS GitHub**

|  |  |
| --- | --- |
| GIT | GITHUB |
| GIT is the VCS which runs locally on your machine. | GitHub is the service that hosts GIT repositories over the cloud and makes it easier to collaborate with other software developers. |
| GIT ≠ GitHub | GitHub is an online place to share work that is done using GIT. |

***To check the git version.***

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ git --version

git version 2.29.1.windows.1

------------------------------------------------------------------------­------------------------------

***To set username and useremail in git because it is needed for commit.***

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git config --global user.name "srikant"

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git config --global user.email "srikantvs26@gmail.com"

------------------------------------------------------------------------­------------------------------

***To check what the current username and email the git is configured.***

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ git config user.name

srikant

------------------------------------------------------------------------­------------------------------

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ git config user.email

[srikantvs26@gmail.com](mailto:srikantvs26@gmail.com)

------------------------------------------------------------------------­------------------------------

***To check the full configurations of git.***

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ git config --list

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager-core

credential.https://dev.azure.com.usehttppath=true

user.email=srikantvs26@gmail.com

user.name=srikant

core.repositoryformatversion=0

core.filemode=false

core.bare=false

core.logallrefupdates=true

core.symlinks=false

core.ignorecase=true

remote.origin.url=https://github.com/srikantvs/wallah.git

remote.origin.fetch=+refs/heads/\*:refs/remotes/origin/\*

branch.main.remote=origin

branch.main.merge=refs/heads/main

------------------------------------------------------------------------­------------------------------

***To open file explorer where this repository is located.***

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ start .

***To create multiple files in repository.***

srikantvs@srikantvs MINGW64 ~/Desktop/dogs/wallah (main)

$ touch one.txt two.txt three.txt last\_file.txt

------------------------------------------------------------------------­------------------------------

***To initialize a git repository.***

Check the folder is already a git repo by running git status command. If not Then…

srikantvs@srikantvs MINGW64 ~/Desktop/animals/my-new-repo

$ git init

Initialized empty Git repository in C:/Users/srikantvs/Desktop/animals/my-new-repo/.git/

------------------------------------------------------------------------­------------------------------

***To see the status of a git repository***

srikantvs@srikantvs MINGW64 ~/Desktop/animals/my-new-repo (master)

$ git status

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

------------------------------------------------------------------------­------------------------------

What is git repository(repo)?

A git repository is a folder in which GIT can do version control. It contains one .git folder.

We can create a git repository using this command : git init.

We create git repository one time per project. 😊

DO NOT CREATE REPOSITORY INSIDE OF ANOTHER REPOSITORY.

------------------------------------------------------------------------­------------------------------

What is git commit?

This is the changes that we have made eg. Line changes, file added, file removed, new folder created deleted etc. We club them together to what we call as commit, It is basically a set of changes we made.

------------------------------------------------------------------------­------------------------------

***To show the commit logs***

srikantvs@srikantvs MINGW64 /e/Saviour (master)

$ git log

commit 865eadf0f55b5894f110fca6b49f3aba07f53229 (HEAD -> master, origin/master, origin/HEAD)

Author: srikant <srikantvs26@gmail.com>

Date: Wed Nov 10 11:03:29 2021 +0530

usecase added for letter frequency demo

commit 922fec4cdc00c6d8c06912de5360974b30586ee7

Author: srikantvs <srikantvs26@gmail.com>

Date: Sun Oct 24 15:04:40 2021 +0530

comments added.

commit fd0697431ee6aea9a3eee630619835ecc92e975d

Author: srikantvs <srikantvs26@gmail.com>

Date: Sun Oct 24 15:01:14 2021 +0530

@ConfigurationProperties Spring Boot Example

------------------------------------------------------------------------­------------------------------

***To show the commit logs in single line.***

$ git log --oneline

865eadf (HEAD -> master, origin/master, origin/HEAD) usecase added for letter frequency demo

922fec4 comments added.

fd06974 @ConfigurationProperties Spring Boot Example

------------------------------------------------------------------------­------------------------------

***To show the last 3 commit logs in single line. When commit messages are huge it helps.***

srikantvs@srikantvs MINGW64 /e/Saviour (master)

$ git log -3 --oneline

865eadf (HEAD -> master, origin/master, origin/HEAD) usecase added for letter frequency demo

922fec4 comments added.

fd06974 @ConfigurationProperties Spring Boot Example

------------------------------------------------------------------------­------------------------------

***To show the last 3 commit logs in single line with given author name.***

$ git log --author="srikant" -3

commit 865eadf0f55b5894f110fca6b49f3aba07f53229 (HEAD -> master, origin/master, origin/HEAD)

Author: srikant <srikantvs26@gmail.com>

Date: Wed Nov 10 11:03:29 2021 +0530

usecase added for letter frequency demo

commit 922fec4cdc00c6d8c06912de5360974b30586ee7

Author: srikantvs <srikantvs26@gmail.com>

Date: Sun Oct 24 15:04:40 2021 +0530

comments added.

commit fd0697431ee6aea9a3eee630619835ecc92e975d

Author: srikantvs <srikantvs26@gmail.com>

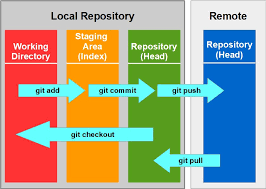
Date: Sun Oct 24 15:01:14 2021 +0530

@ConfigurationProperties Spring Boot Example

------------------------------------------------------------------------­------------------------------

.git folder.

**GIT WORKFLOW**



srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status

On branch master

nothing to commit, working tree clean

srikantvs@srikantvs MINGW64 /f/Test (master)

$ touch outline.txt new file created.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status check repo status.

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

outline.txt untracked file.

nothing added to commit but untracked files present (use "git add" to track)

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git add outline.txt add file to staging area

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status check repo status.

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: outline.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit -m "start work on outline and main characters" commit the changes with commit message.

[master b4ca29b] start work on outline and main characters

1 file changed, 1 insertion(+)

create mode 100644 outline.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status working tree is clean now.

On branch master

nothing to commit, working tree clean

**ATOMIC COMMITS**

When possible, a commit should contain a single feature, change or a fix, In other words, try to keep each commit focuses on a single thing.

This makes it much easier to undo or rollback changes later on. It also makes your code or project easier to review.

**WRITING COMMIT MESSAGES (should be in imperative mood, Present Tense)**

It should be like you are giving orders to the codebase to change its behaviour. Eg.

1. finish chapter 1
2. rename test to rest
3. fix typo

**CONFIGURING THE DEFAULT EDITOR IN GIT.**

<https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Setup-and-Config>

srikantvs@srikantvs MINGW64 /f/Test (master)

$ touch character.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git add character.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit now vim opened, I don’t want that.

Aborting commit due to empty commit message.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git config --global core.editor "code --wait" copied this from link.

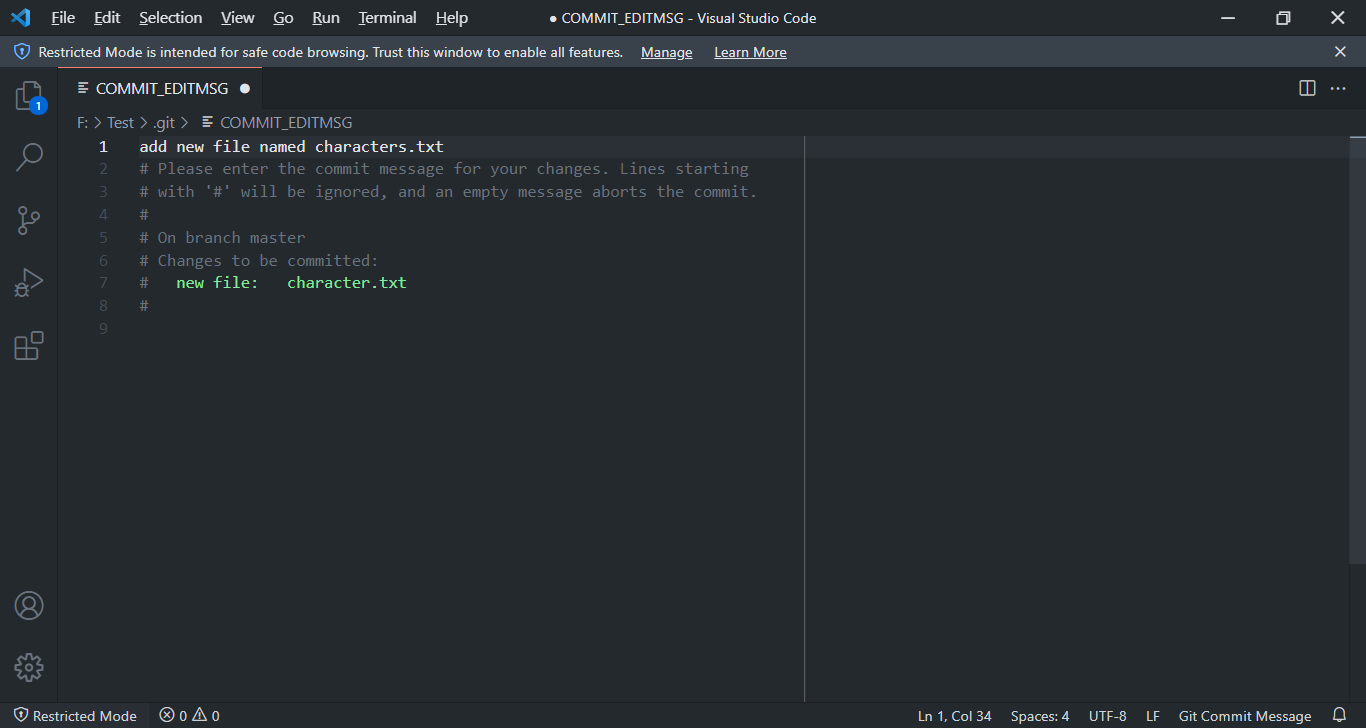
srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit now vscode opened.

[master fbd2739] add new file named characters.txt

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 character.txt



**AMMENDING COMMITS(Correcting ONLY the last commit ONLY).**

Suppose you just made a commit and then realized you forgot to include a file! Or, maybe you made a small typo in the commit message that you want to correct.

Rather than making a brand new separate commit, you can redo the previous commit using the –-amend option.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit you committed your changes.

[master fbd2739] add new file named characters.txt

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 character.txt

$ git status Oh No, one file missed.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git add forgotten\_file.txt add missed file.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit --amend amend the last change. Vscode will open

[master 7734744] add new file named characters.txt, forgotten\_file.txt as well.

Date: Sat Nov 20 13:57:40 2021 +0530

2 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 character.txt

create mode 100644 forgotten\_file.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git log -3 --oneline last commit is changed.

7734744 (HEAD -> master) add new file named characters.txt, forgotten\_file.txt as well.

b4ca29b start work on outline and main characters

ccfc81d deleted

**------------------------------------------------------------------------------------------------------**

**Correcting the Commit message ONLY.**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git log --oneline see the last commit message.

7734744 (HEAD -> master) add new file named characters.txt, forgotten\_file.txt as well.

b4ca29b start work on outline and main characters

ccfc81d deleted

aebd51d added new three files

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit --amend -m "this is the modified commit message without using vscode"

[master dc6d873] this is the modified commit message without using vscode

Date: Sat Nov 20 13:57:40 2021 +0530

2 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 character.txt

create mode 100644 forgotten\_file.txt

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git log --oneline

dc6d873 (HEAD -> master) this is the modified commit message without using vscode

b4ca29b start work on outline and main characters

ccfc81d deleted

aebd51d added new three files

**Committing using one command only.**

The file should be tracked by git o/w this will not work. So we have make the git track the file using git add <filename> Then this command will work.

srikantvs@srikantvs MINGW64 /f/Test (new\_branch\_from\_master)

$ vim myOldSongs.txt added few songs

srikantvs@srikantvs MINGW64 /f/Test (new\_branch\_from\_master)

$ git commit -a -m "added few old songs" shortcut brother.

warning: LF will be replaced by CRLF in myOldSongs.txt.

The file will have its original line endings in your working directory

[new\_branch\_from\_master b3291d1] added few old songs

1 file changed, 6 insertions(+)

**IGNORING FILES**

We can tell Git which files and directories to ignore in a given repository, using a .gitignore file. Those files or folders mentioned in that file will not be tracked by git. So, we can mention files like :

Secrets, API Keys, DB Credentials or any such file

Operating System Files

Log files

node\_modules

dependencies and packages

**How it works**

Create a file called .gitignore in the root of a repository. Inside the file, we can write patterns to tell Git which fils and folder to ignore.

folderName/ will ignore the directory

\*.log will ignore files with the .log extension

srikantvs@srikantvs MINGW64 /f/Test (master)

$ touch .gitignore

srikantvs@srikantvs MINGW64 /f/Test (master)

$ cat .gitignore what .gitignore contains.

\*.log

node\_modules/

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

nothing added to commit but untracked files present (use "git add" to track)

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git add . adding .gitignore file.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git commit -m "add .gitignore file" committing

[master c0e4414] add .gitignore file

1 file changed, 2 insertions(+)

create mode 100644 .gitignore

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git status now it is not tracking those files.it is ignoring them

On branch master

nothing to commit, working tree clean

[**https://www.toptal.com/developers/gitignore**](https://www.toptal.com/developers/gitignore)

With the above website we can create useful .gitignore files for our project. Very good.

**WORKING WITH BRANCHES**

Contexts

On large projects, we often work in multiple contexts:

* You're working on 2 different color scheme variations for your website at the same time, unsure of which you like best
* You're also trying to fix a horrible bug, but it's proving tough to solve.
* You need to really hunt around and toggle some code on and off to figure it out.
* A teammate is also working on adding a new chat widget to present at the next meeting. It's unclear if your company will end up using it.
* Another co-worker is updating the search bar autocomplete.
* Another developer is doing an experimental radical design overhaul of the entire layout to present next month.

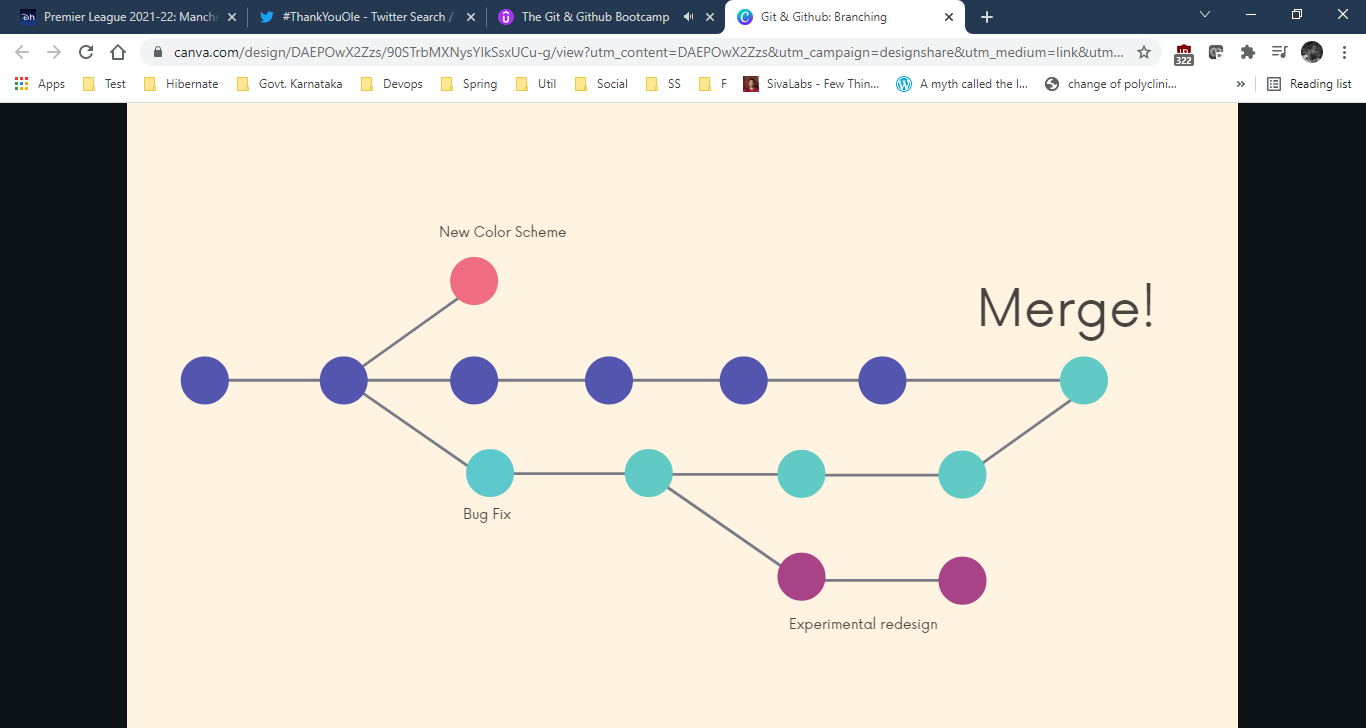
**Branches**

Branches are an essential part of Git!

Think of branches as alternative timelines for a project.

They enable us to create separate contexts where we can try new things, or even work on multiple ideas in parallel.

**If we make changes on one branch, they do not impact the other branches (unless we merge the changes)**



**MASTER BRANCH OR MAIN BRANCH**

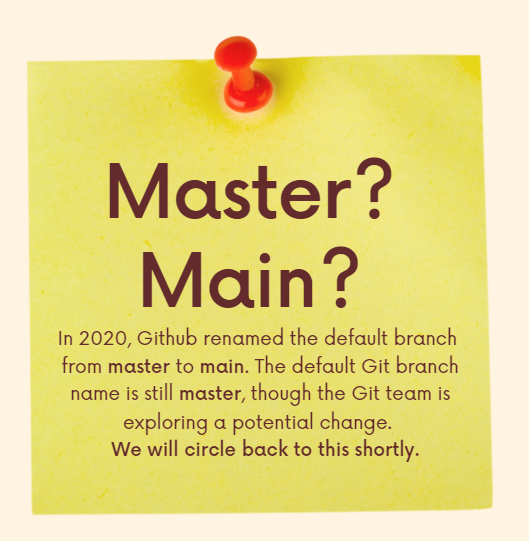
**Master Branch**: In git, we are always working on a branch**\*** The default branch name is **master**.

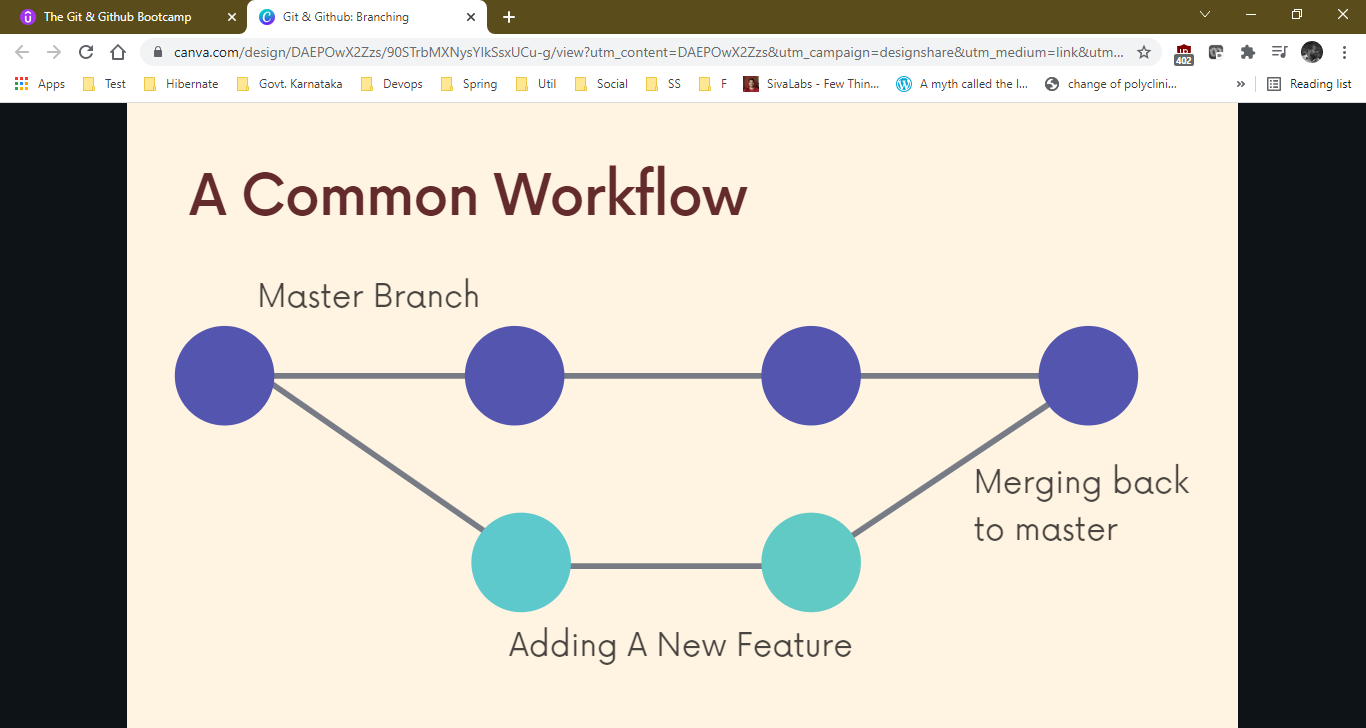
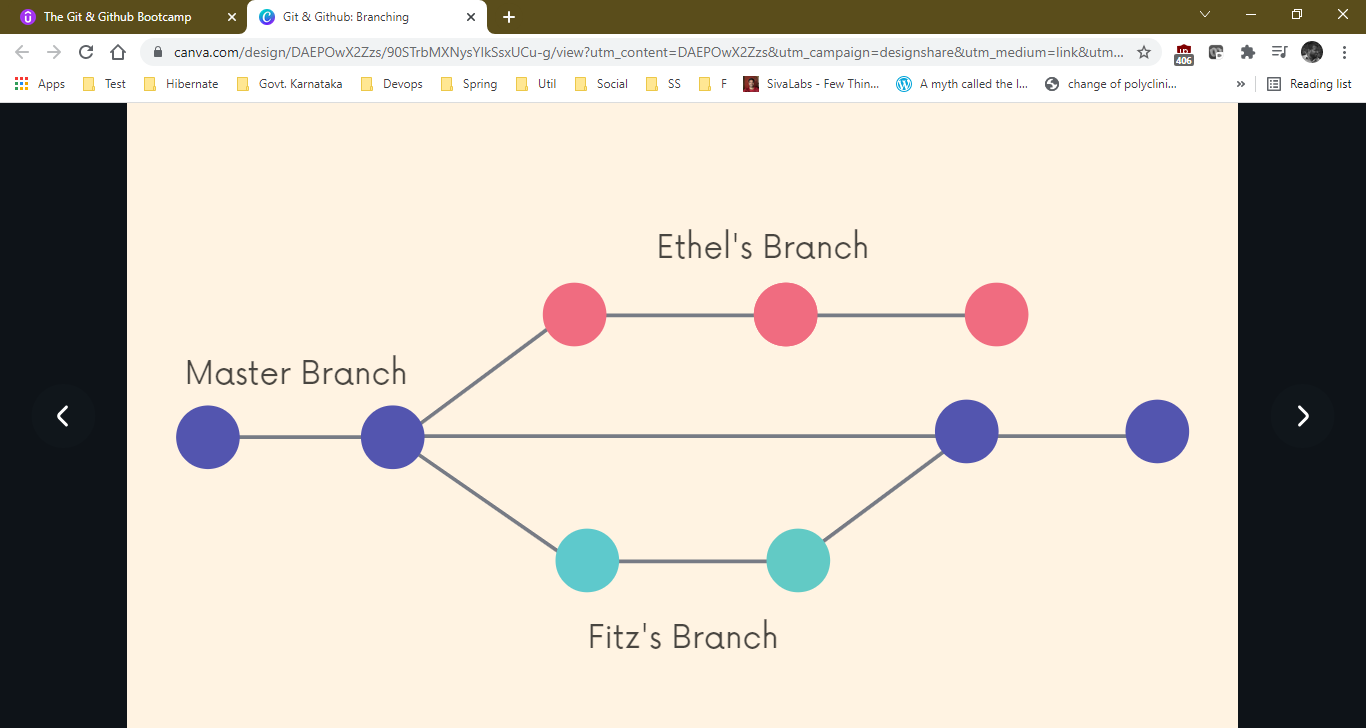
It doesn't do anything special or have fancy powers. It's just like any other branch.

**Master**

Many people designate the master branch as their "source of truth" or the "official branch" for their codebase, but that is left to you to decide.

From Git's perspective, the master branch is just like any other branch. It does not have to hold the "master copy" of your project.





**What is HEAD?**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git log -2 --oneline

c0e4414 (HEAD -> master) add .gitignore file

dc6d873 this is the modified commit message without using vscode

We can see HEAD is currently pointing to master branch

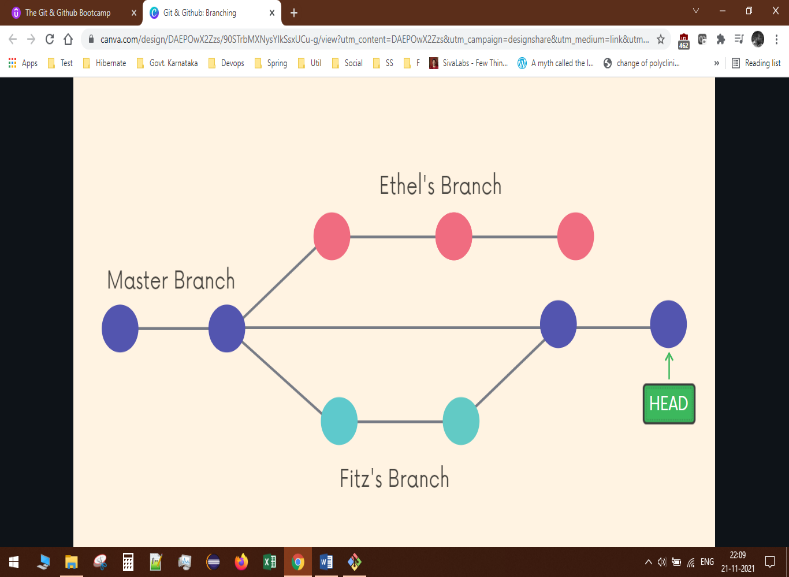
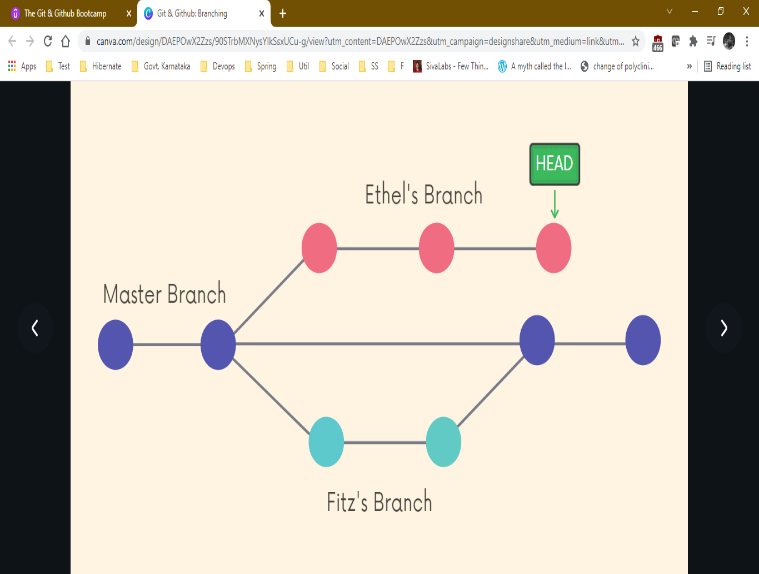
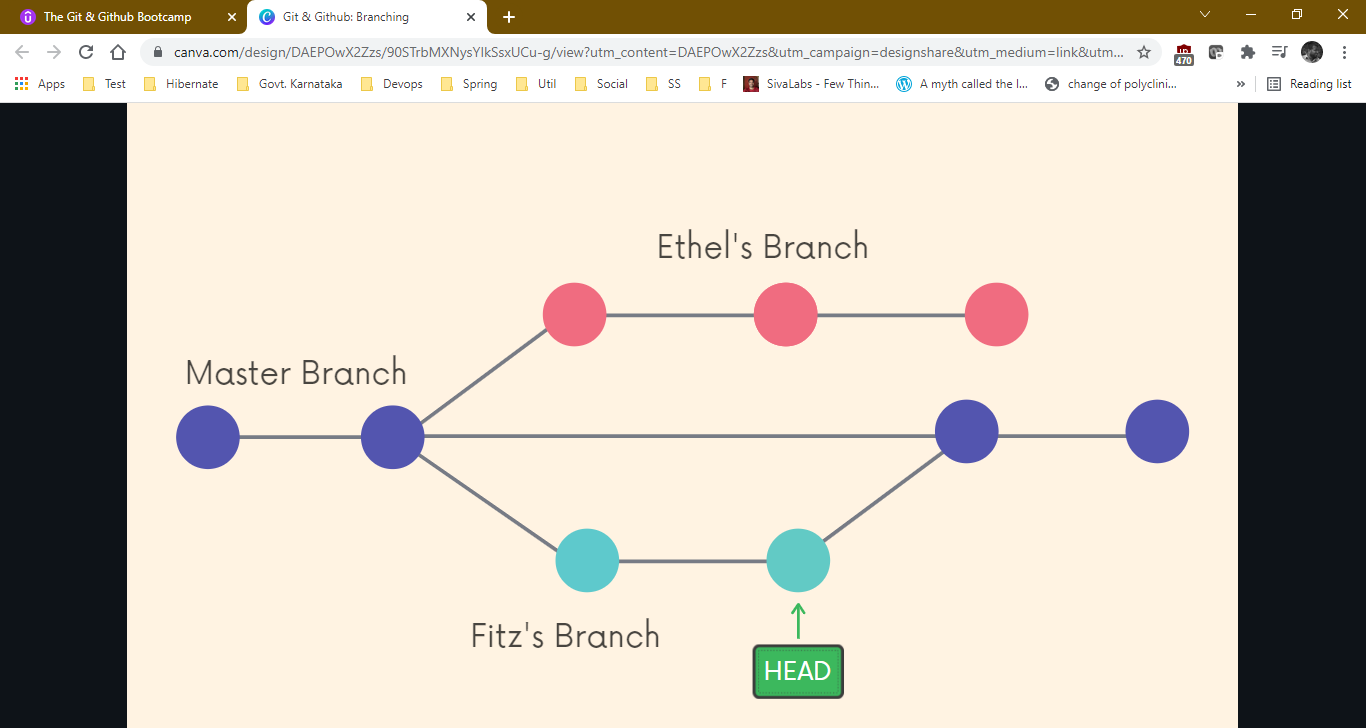
What is that HEAD -> master thing?

**HEAD**

We'll often come across the term HEAD in Git.

HEAD is simply a pointer that refers to the current "location" in your repository. It points to a particular branch reference.

So far, HEAD always points to the latest commit you made on the master branch, but soon we'll see that we can move around and HEAD will change!



**Viewing Branches (What all branches are there in the repository)**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch

\* master

**\*** is current branch on which you are.

----------------------------------------------------------------------------

**Creating Branches**

Use **git branch <branch-name>**to make a new branch based upon the current HEAD where it is right now pointing. We can check using git status command.

This just creates the branch. It does not switch you to that branch (the HEAD stays the same)

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch my-feature-1

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch

\* master

my-feature-1

Here I created new branch from master branch where HEAD was pointing to.

----------------------------------------------------------------------------

**Switching Branches** (To move between the branches)

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch

\* master

my-feature-1

test

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git checkout my-feature-1 this is older way but good.

Switched to branch 'my-feature-1'

srikantvs@srikantvs MINGW64 /f/Test (my-feature-1)

$ git switch test this is the new way.

Switched to branch 'test'

----------------------------------------------------------------------------

**Creating And Switching Branch In One Go.**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git checkout -b new\_branch\_from\_master

Switched to a new branch 'new\_branch\_from\_master'

srikantvs@srikantvs MINGW64 /f/Test (new\_branch\_from\_master) see its changed.

$

Another way.

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git switch -c my-new-branch -c : create

Switched to a new branch 'my-new-branch'

srikantvs@srikantvs MINGW64 /f/Test (my-new-branch)

$

**Git Checkout V/S Git Switch**

Historically, we used git checkout <branch-name> to switch branches. This still works.

The checkout command does a million additional things, so the decision was made to add a standalone switch command which is much simpler.

----------------------------------------------------------------------------

**We cannot switch branches with dirty working tree(branch).**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git switch my-feature-1

error: Your local changes to the following files would be overwritten by checkout:

character.txt

Please commit your changes or stash them before you switch branches.

Aborting

Still we can switch branches if we have created new file. It follows.

----------------------------------------------------------------------------

**To rename a branch**

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch

\* master

my-feature-1

my-new-branch

new\_branch\_from\_master

test

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git switch test

Switched to branch 'test'

srikantvs@srikantvs MINGW64 /f/Test (test)

$ git branch -m test test2

srikantvs@srikantvs MINGW64 /f/Test **(test2)**

$

----------------------------------------------------------------------------

**To delete a branch** (You can’t delete the branch on which you currently are).

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch -d test2

Deleted branch test2 (was c0e4414).

srikantvs@srikantvs MINGW64 /f/Test (master)

$ git branch

\* master

my-feature-1

my-new-branch

new\_branch\_from\_master