**GIT Practice- document**

Git is distributed version control system (DVCS).

**Centralized VCS**

• One central repository

• Must be capable of connecting to repo

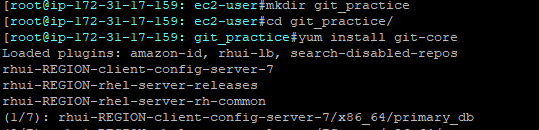
• Need to solve issues with group members making different changes on the same files

**Distributed VCS**

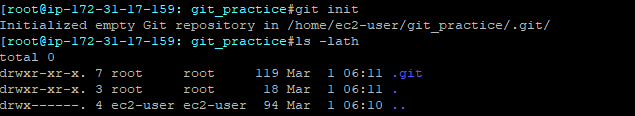
• Everyone has a working repo

• Faster & Connectionless

Create a new directory to install git



Initialize git in same directory, than .git directory(consisting all the meta-data) will be created

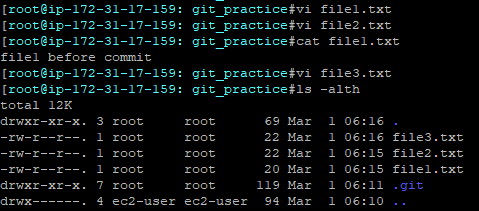


Connect to git server by adding the credentials



You need to do this only once if you pass the **--global** option, because **then Git will always use that information for anything you do on that system**. If you want to override this with a different name or email address for specific projects, you can run the command without the **--global** option when you’re in that project.

Create 3 sample files over which we will do our practice



Once the files are present in our local repository, once files are ready to push to git repository

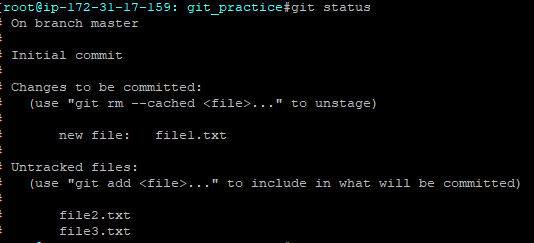
ADD the files, so that they will come to our local repository



Check the status of files, by using ‘git status’ command.

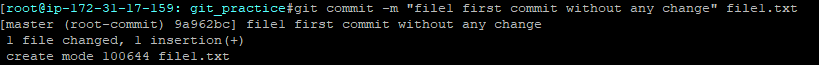
This will give you following information:

1. Branch on which file is added
2. Which file is to be committed after adding
3. Untracked or unadded files

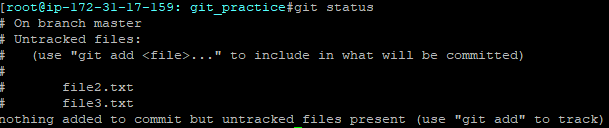


It’s recommended to commit the files, before pushing it to git repository and after adding so that we can check the status and changes done in file.

The "git commit" command only saves a new commit object in the local Git repository.



After commit, the git status also changes as below:



Difference between git add and git commit:

For example, you're working on two different things, and now your code looks like

//random code

//bug fix

//new feature

You can stage just the //bug fix line and commit that, and then stage the //new feature line and commit that.

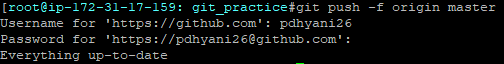
You now have two different commits for each. Later on in testing you realize the new feature commit broke something, you can delete the new feature commit, but don't have to recommit the bugfix.

Specify the project name to which you want to add files to remote git repository



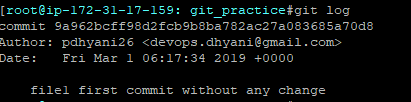
**Origin:**

If no file is added and committed than on pushing the files, we will get

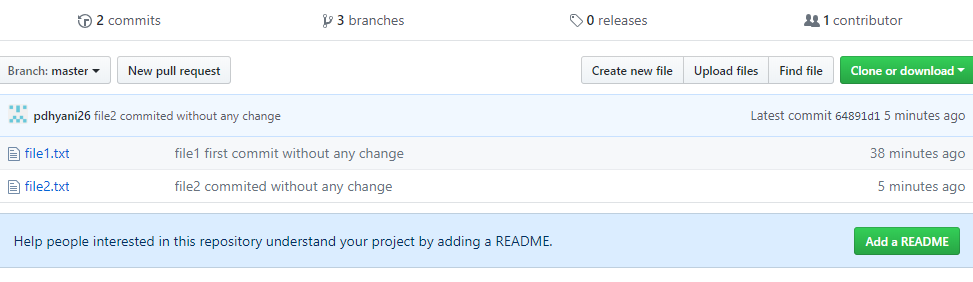
’

NOTE: If we add a file and push it without committing the, message would be same as “Everything is up to date”

‘git log’ command will provide the log of committed files, like user who committed and the timestamp



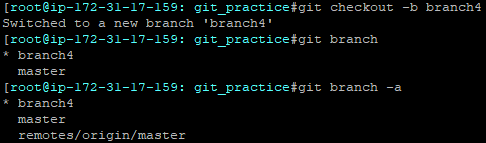
Files are showing successfully under branch- Master in GIT HUB



To know, all the branches present, use below command. ‘\*’ depicts the current branch you are working on



To create a new branch and switch to it



To switch to another branch



* **Revert ALL THE THINGS we changed:**

git checkout -f

* **If we already committed it:**

git revert HEAD # Revert last commit (HEAD)

1. What exactly is merge branch
2. When we provide an existing project having branch, why they don’t reflect in git branch command, due to that we are unable to merge current branch with those
3. Git stash, fork
4. .gitignore file : If you create a file in your repository named *.gitignore*, Git uses it to determine which files and directories to ignore, before you make a commit.

A *.gitignore* file should be committed into your repository, in order to share the ignore rules with any other users that clone the repository.

Ignored files are usually build artefacts and machine generated files that can be derived from your repository source or should otherwise not be committed. Some common examples are:

Dependency caches, such as the contents of /node\_modulesor /packages

compiled code, such as .o, .pyc, and .class files

build output directories, such as /bin, /out, or /target

files generated at runtime, such as .log, .lock, or .tmp

hidden system files, such as .DS\_Store or Thumbs.db

personal IDE config files, such as .idea/workspace.xml

**to make a global gitignore file**

$ touch ~/.gitignore

$ git config --global core.excludesFile ~/.gitignore

**Ignoring a previously committed file**

$ echo debug.log >> .gitignore

$ git rm --cached debug.log

rm 'debug.log'

$ git commit -m "Start ignoring debug.log"

1. README file : includes the information about the project that you want to enter

If you try to get ‘git log’ without committing any file just after initializing



**READ GIT CHEAT SHEET DOWNLOADED**