**Git and GitHub**

* **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. These versions are recorded in a repository and can be recalled from the same. There are local, centralized and distributed VCS.

* **Why Version Control System?**
  + Collaboration – shared workspace and real-time updates
  + Manage versions – all versions of code are preserved.
  + Rollbacks – easy rollback from current version.
  + Reduce downtime – reverse faulty update and save time.
  + Analyse project – analyse and compare versions
* **What is a repository?**

A repository (or a repo) is a directory or storage space where your projects can live. It can be local to a folder on your computer or it can be a storage space on another online host (such as GitHub). You can keep code files, text files, image files you name it inside a repository.

* **Centralized vs Distributed:**
* We are working on our local system and we have a single file that only you are working on in that case it makes very little sense to use a VCS.
* When multiple people working on the same file in that case VCS becomes very important because person A works on that file and makes changes. Person B also works on the same file and makes changes. We have to track all the changes made by all the people on that particular file. So, the VCS plays important role in that case.

Two types of VCS:

**Centralized VCS**



* The repository is located on remote server.
* The people who are working on this repository have to be connected always and have to be online to work on their systems because this repository is placed on a server.
* They work on this repository online and make their changes committed and so they can collaborate.

**Disadvantage**

A single repository is placed on a server. So, if anything goes wrong it will be difficult for us to maintain the backup and bring the repository to the normal.

**Distributed VCS**



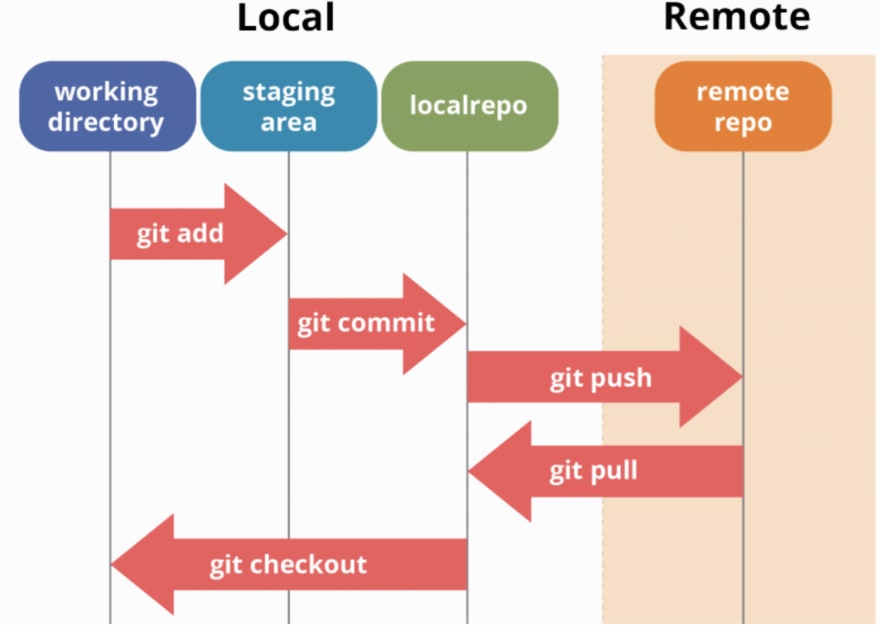
* Here every person can pull and create his local repository. So, you can see workstation, one has a local repository workstation, two has his own copy workstation.

**Advantage**

* If anything goes wrong here on the server we can backup the repository from any local system because these are the complete copy of the repository and also do not need to be online. Every time you can pull the repository take your local copy and keep on working on it in offline mode.
* **What is GIT?**
* **Git** is a widely-used and powerful version control system. It was created by Linus Torvalds in 2005 to manage the Linux kernel development.
* There are two types of Version Controlling: Centralized and Distributed.
* Git is a Distributed Version Control System, which means that each developer has a copy of the entire codebase and its history on their local machine. This allows for offline development and efficient branching and merging.
* It supports non-linear workflows by providing data assurance for developing quality software.
* It let’s you and your team of developers work together on the same project from anywhere.
* Team members can work on files and easily merge their changes into one source.
* Key benefits:
* To track changes in files / folders
* To collaborate in teams
* It is free and open source.

**GIT Features**

* Economical – released under GPL’s license. It is free and open-source.
* Non-linear – supports non-linear development of software.
* Snapshots – records changes made to a file rather than file itself.
* Distributed – every user has his own copy of the repository data stored locally.
* Speed – speed offered by git is lightening fast compared to other VCs.
* Robustness – Nearly every task in git is undo-able.
* Integrity – No changes can be made without git recording it.
* Branching – Every collaborator’s working directory is a branch by itself.
* **Workflow of GIT:**



**Git installation and setup:**

Install GitBash for the windows

* Log on to <https://git-scm.com/downloads>
* Click Download for windows button.
* Run the downloaded executable file and follow through till you finally install git.

Install git for linux:

* sudo apt update
* sudo apt install git
* git –version

**GIT HUB** – a website to upload your repositories.

Signup and create a account on GitHub [https://github.com/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbklHQlFILWRiVXp3N1BhelBrTEgtR3BnT3RTd3xBQ3Jtc0tuSjBLZF8zMWtCanFfbXZ4YWk1bm1aTnFWdDJ1X0xFZXJyaFEzQjRrTTc2WlN6WVBHdTZneGpwVWJCT1hRR0RqUVBsUG96MFI0c05ZQVRYZ3NNSm96ZHgweEp1TXk1cWFNN3RyOEw5WU4tMWh3X2dkRQ&q=https%3A%2F%2Fgithub.com%2F&v=0Icla6TVNNo)

* It provides backup, visual interface to your repository. It makes collaboration easier.
* Git and GitHub is not same. Git is a VCS whereas GitHub is a website to upload our files/folders.

**GIT Operations and Commands:**

* git init – creating repositories
* making changes – status, add, commit
* parallel development – branch, merge, rebase
* syncing repositories – origin, push, pull

Create repository: git init, git clone, git fork

* git init – create new git repository
* git clone – copy of original repository on your local machine
* git fork – create a copy of original repository on your github account.

Sync repository:

* git origin -add a remote repository.
* git pull – copy all the files from the master branch of remote repository to your local repository.
* git push – push your local changes into central repository.
* git remote add origin <repository link>
* git pull origin master
* git push origin master

**A simple workflow:**

Git has three types of files

* Untracked files -- working directory
* Staged files -- staging area
* Committed files – Local Repository

Untracked to Staging area -- git add filename (or) git add .

Staging area to Local repository -- git commit -m “msg”

Staging area to untracked section -- git restore --staged filename

$ git log (lists commits, date and time, username)

$ git log --oneline (lists only commits)

**Git Setup on Local System:**

**Step 1**: check if git is already installed.

git --version

**Step 2:** download Git installer from [https://git-scm.com/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmtrdVFJMVNZRmtXTlNXUm1BQWJJS19abERlZ3xBQ3Jtc0tuMGxNUFJXc2h4XzE0TW9yWUd3VmMycnhVcWxMQmduRE5TbXN6SXRYc0dsR0ViQl94RlpGSTVRclB5RGZCc0Z5enE0SzFMQXZsU25Rbnd5QVB5cDJ5U2NrUUVJcG5SUGt5YnNFWl81YVZOWHZtM212RQ&q=https%3A%2F%2Fgit-scm.com%2F&v=B8NurikxEo8)

**Step 3:** Signup and create a account on GitHub [https://github.com/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbklHQlFILWRiVXp3N1BhelBrTEgtR3BnT3RTd3xBQ3Jtc0tuSjBLZF8zMWtCanFfbXZ4YWk1bm1aTnFWdDJ1X0xFZXJyaFEzQjRrTTc2WlN6WVBHdTZneGpwVWJCT1hRR0RqUVBsUG96MFI0c05ZQVRYZ3NNSm96ZHgweEp1TXk1cWFNN3RyOEw5WU4tMWh3X2dkRQ&q=https%3A%2F%2Fgithub.com%2F&v=0Icla6TVNNo)

**Step 4:** Add your github email and username to git

$ git config --global user.email "yourGitHub@email.com"

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

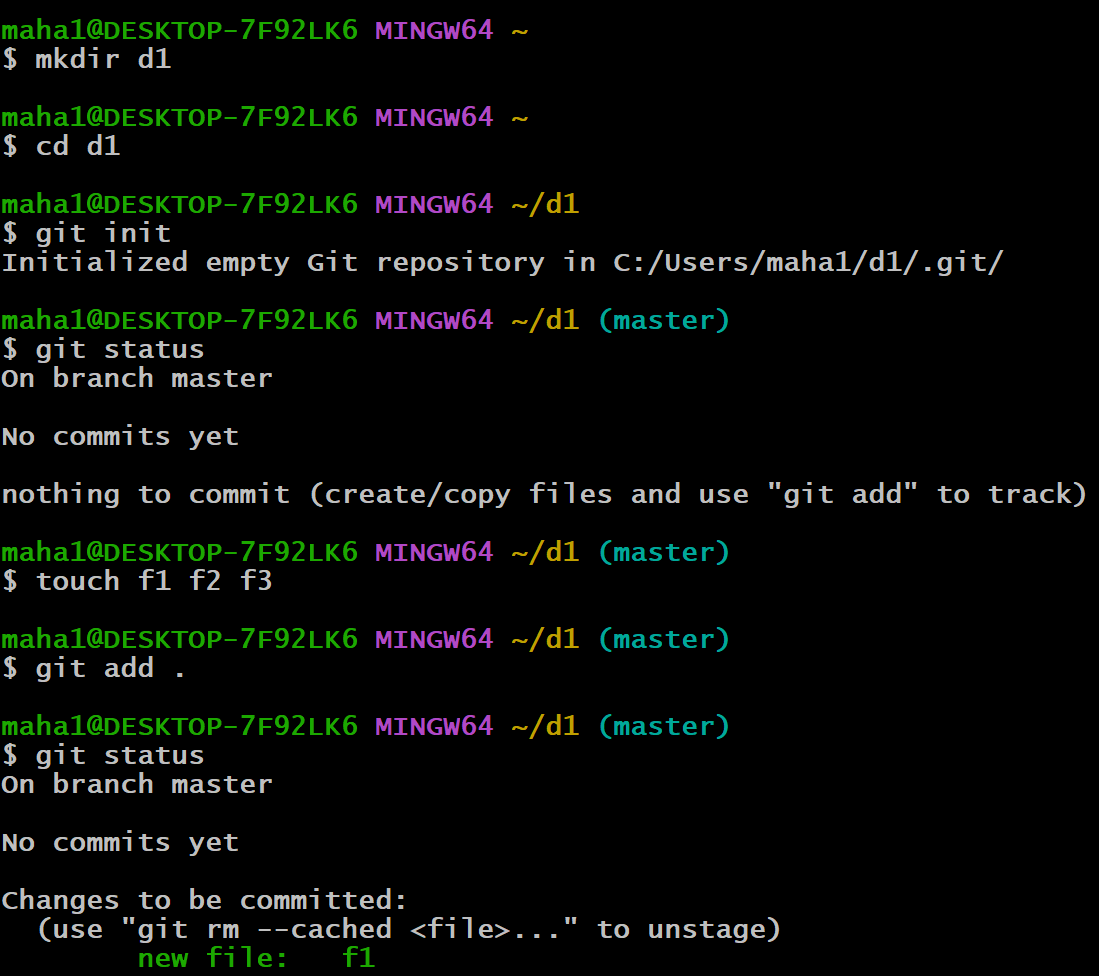
To check the configurations:

$ git config --global --list

**Step 5:** Add file/folders to git - tracking

**Step 6:** Commands - terminal - goto the location of the folder/project

**Hands-on practice:**



How to make working directory as git repository

$ git init

-Observation -

++++++++++++++++

$ git status

+++++++++++++++++

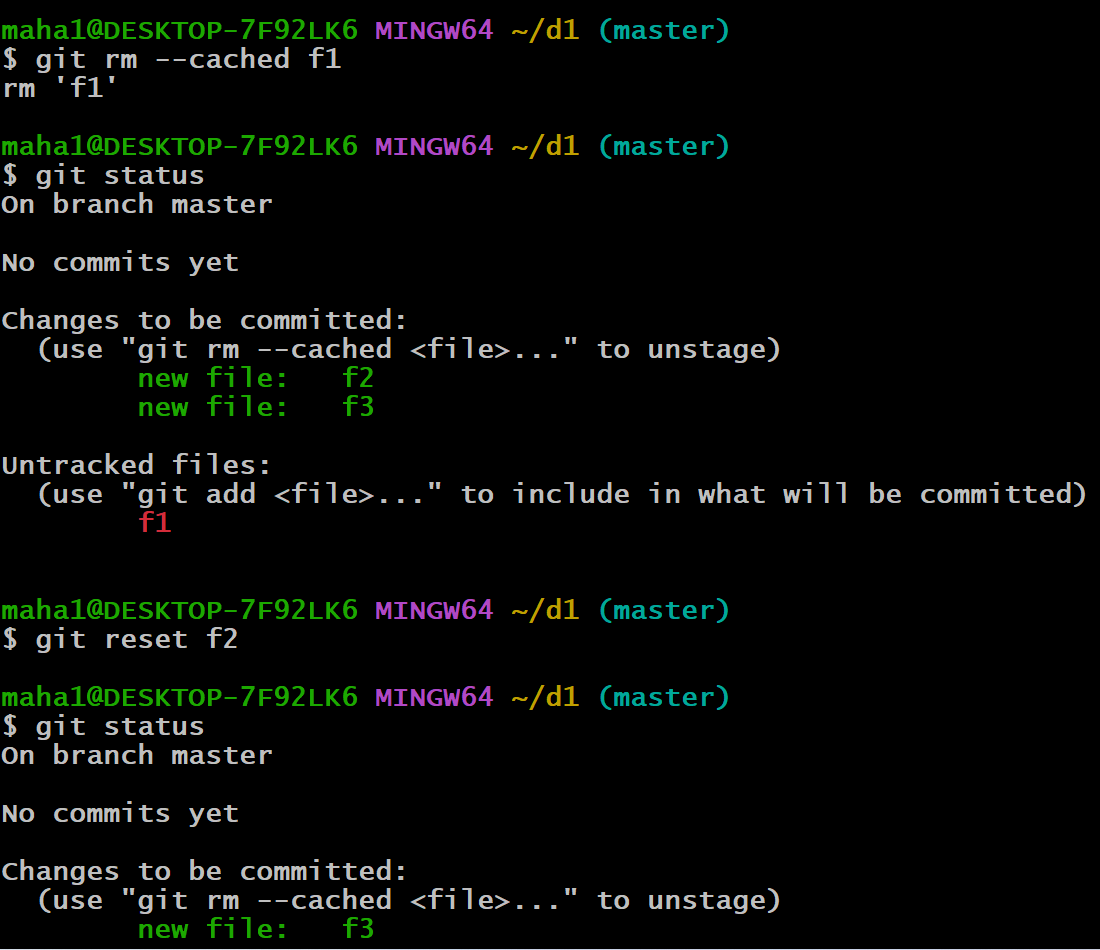
To move file to staging area

$ git add f1

$ git add f2 f3

$ git add .

++++++++++++++++



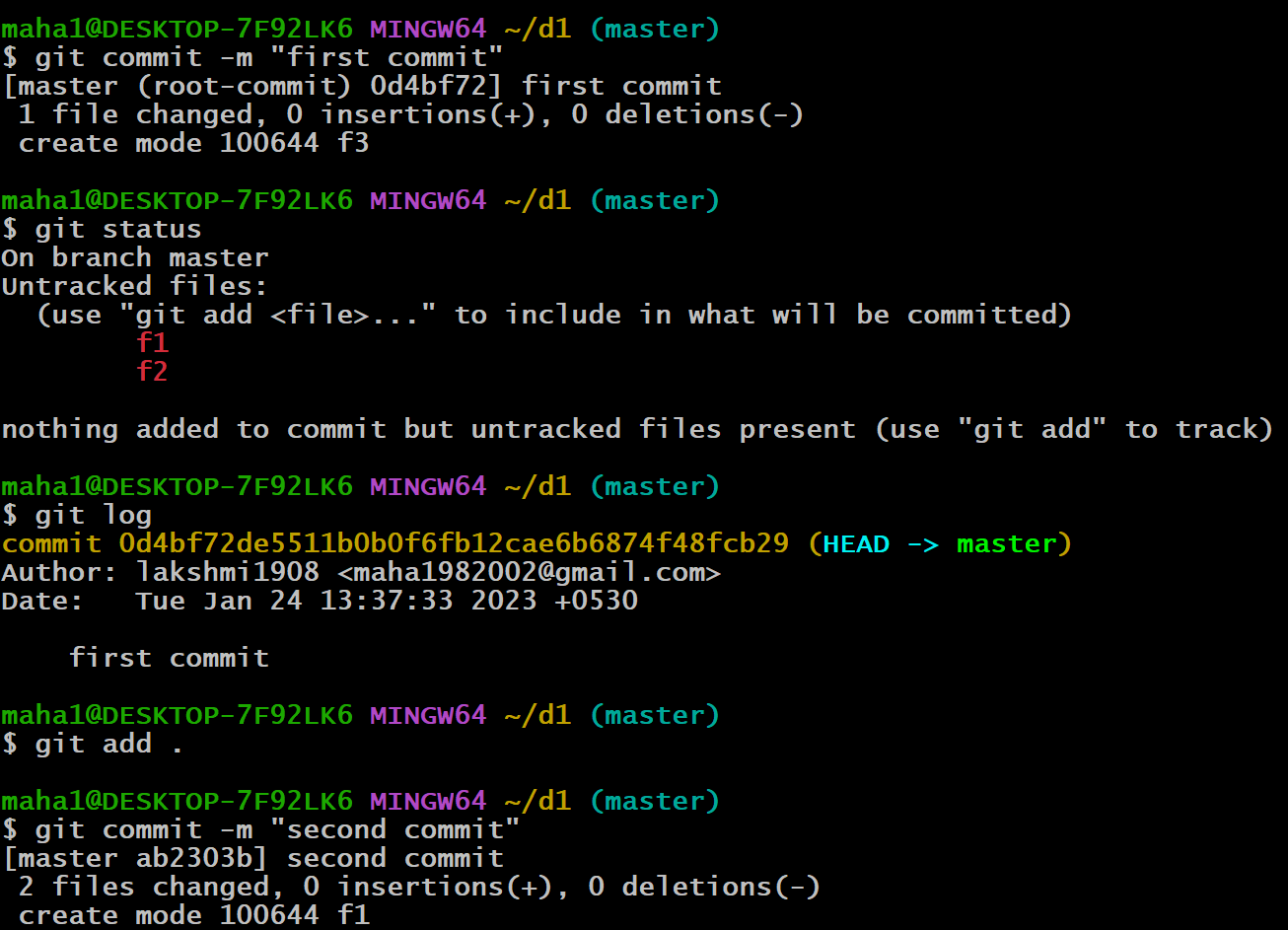
Bring file back to untracked section

$ git rm --cached f1

or

$ git reset f2

++++++++++++++++++++



To move the files from staging area to Local Repository

$ git commit -m "first commit"

$ git status

+++++++++++++++++++



To see the list of commit

$ git log

+++++++++++++++++++

remaining files I want to move to Local Repository.

$ git add .

$ git commit -m "second commit"

$ git log

(or)

git log --oneline

$ git status

**Parallel Development:**

* Branching is an integral part of any version control system.
* Unlike other vcs, git does not create a copy of existing files for new branch.
* It points to snapshot of the changes you have made in the system.

**Branch:** This feature is created in git, so that developers can create code related to different functionalities on separate branches. This helps the development team in creating the code in an uncluttered way. Later this code can be merged with master branch. Default branch of git is “master”.

Branch is a feature where we work on multiple things at the same time. Branches are master, hotfix, release, develop, feature.

**Branching and Merging**

**Step 1:** Create branch **git branch “branch name”**

**Step 2:** Checkout branch **git checkout “branch name”**

**Step 3:** Merge new branch in master branch **git merge “branch name”**

**Step 4:** Delete branch **git branch -d “branch name”** — delete from local git push origin —delete “branch name” — delete from remote

1) **git branch** to check the current branch we are in

2) **git branch -A** to check all the branches

3) **git branch -a** short information about current branch

4) **git branch <branchname>** to create a branch

5) **git checkout <branchname**> to switch over to that branch.

6) **git checkout -b <branchname>** both create and switch over to that branch

7) **git merge <old> <new>** to merge the branches

8) **git branch -d <branchname>** to delete a branch

**Git ignore**

touch .bak .exe .c

{vi .gitignore

.bak

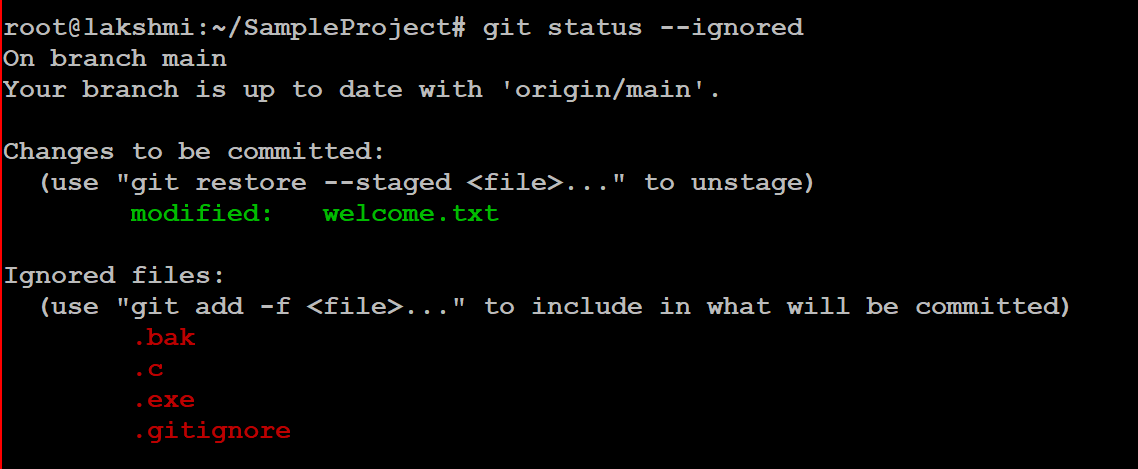
.exe

.c

.gitignore}

git add .

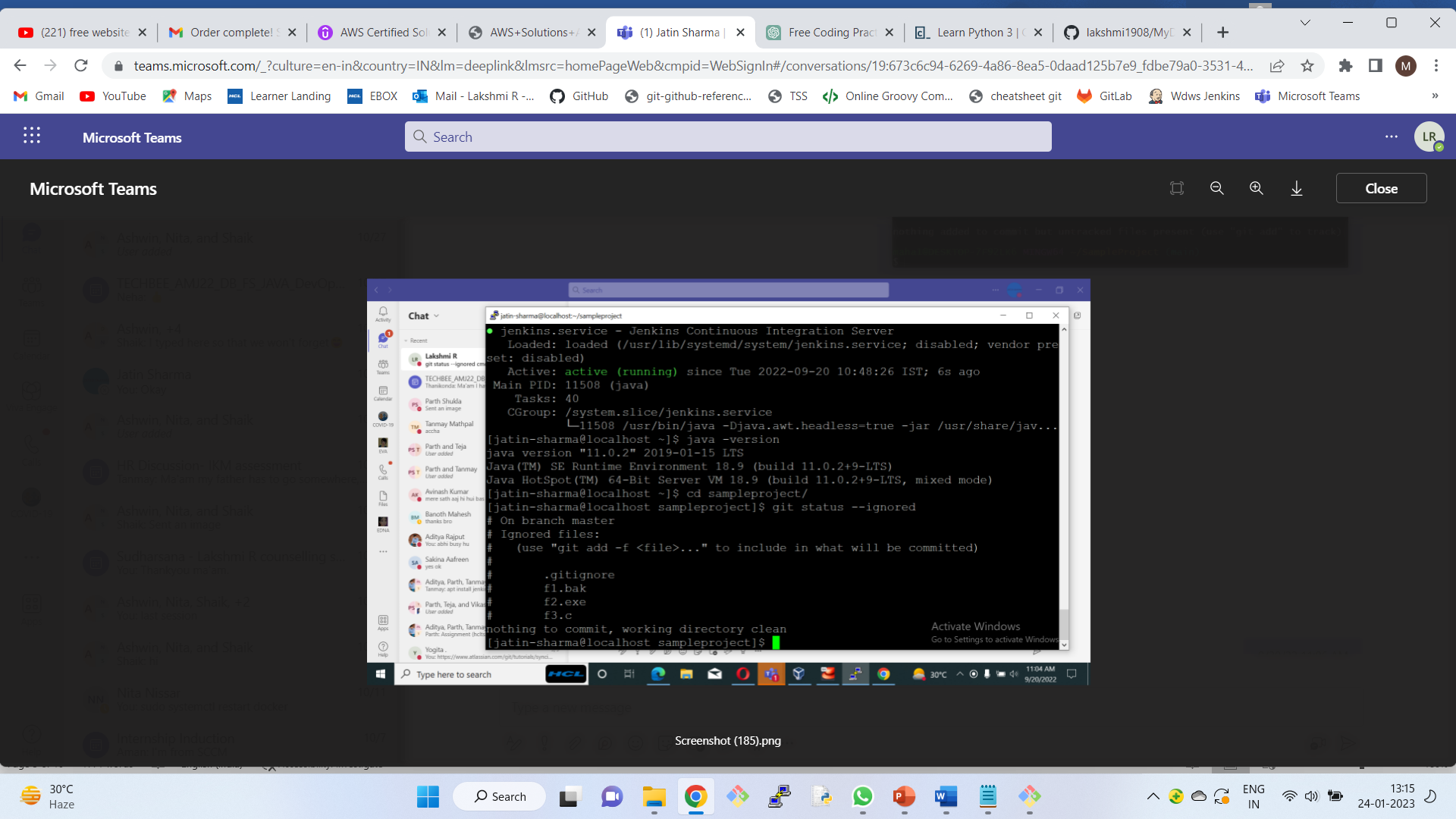
git status –ignored



Understanding .gitignore file

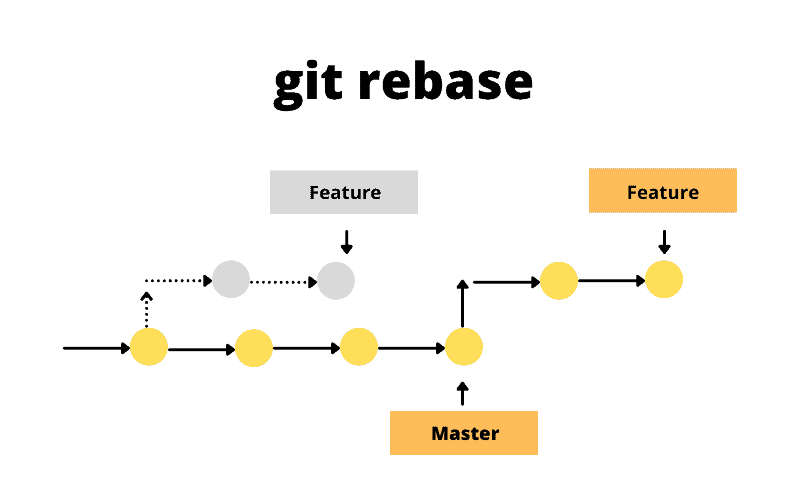
vi .gitignore

\*.class



**git rebase**

* This is used when changes made in one branch needs to be reflected in another branch.
* This is called as fastforward merge.
* The commits from the child branch are added to the top of the master branch.
* This is helpful when we want code from a branch to be reflected as the latest working version on master.



- Moves the entire feature branch to begin on the tip of the master branch

- Re-writes the project history

- We get much cleaner and linear project history.

**git merge:**

Take the independent lines of development created by git branch and integrate them into a single branch.