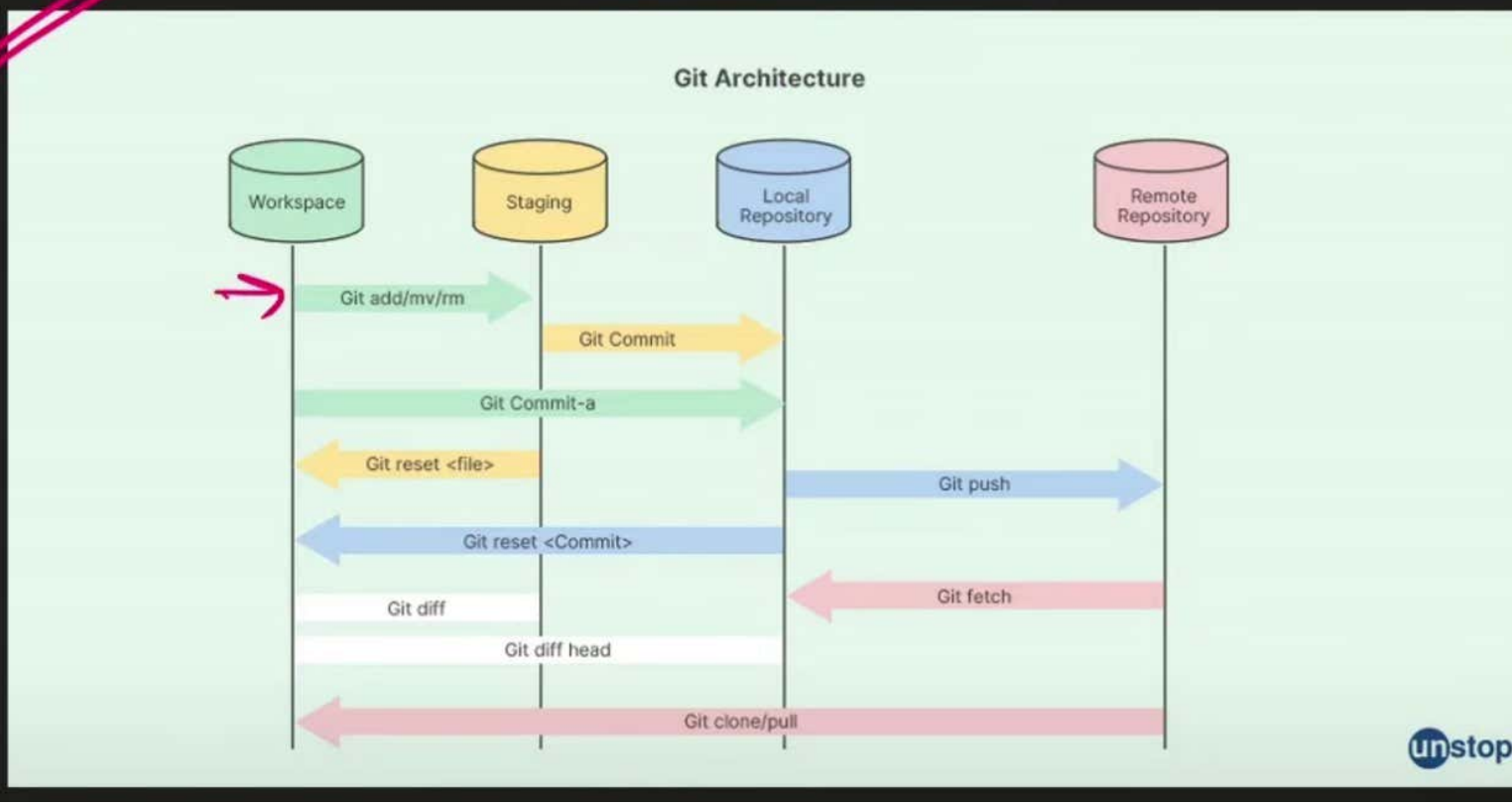
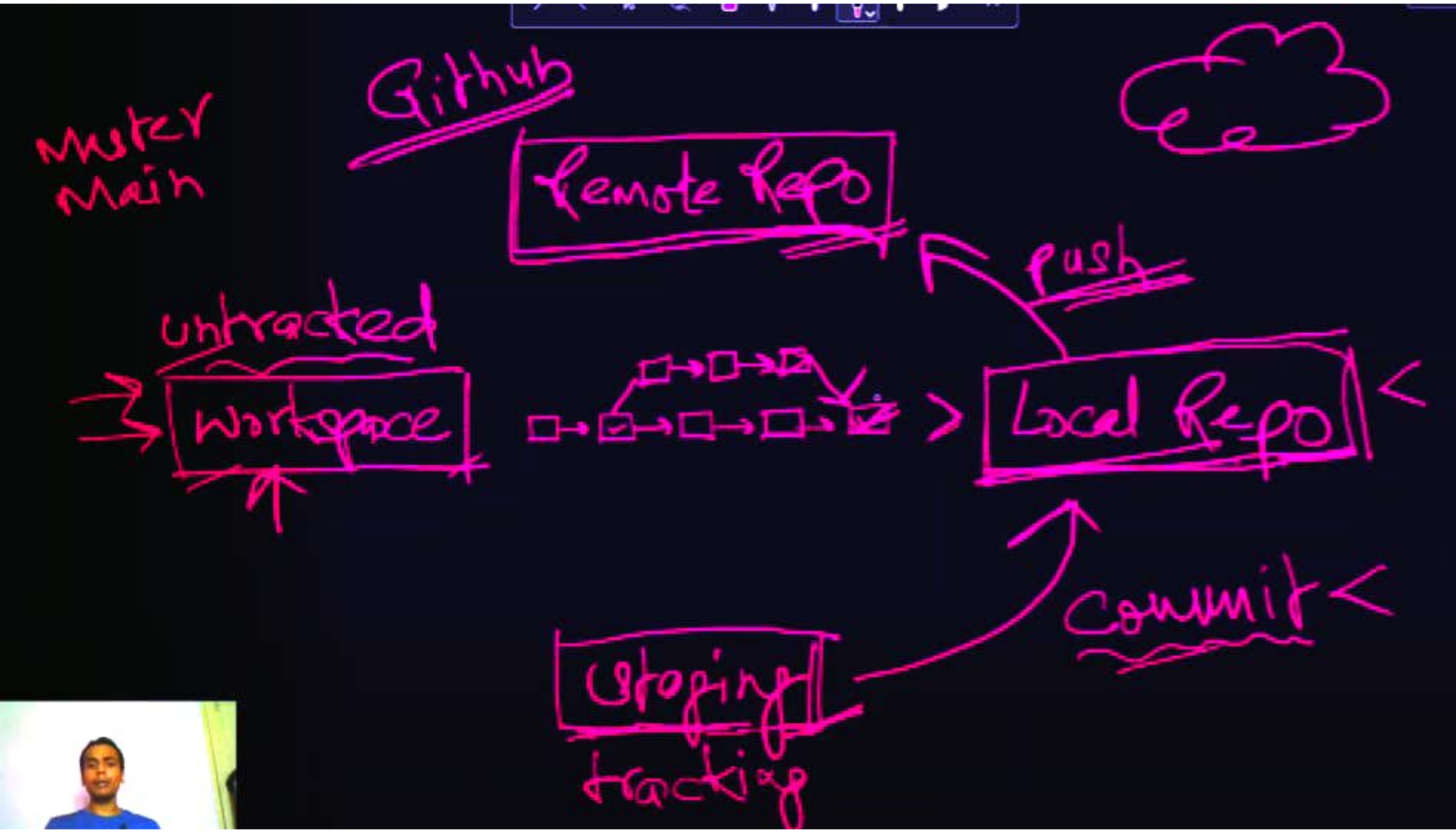
Source: https://unstop.com/blog/what-is-git




Github
Master
Main
Remote Repo
push
Untracted
Workspace
Local Repo
commit
Staging
tracking

GitHub Commands & Actions

Visit --> Upcoming pages

BASIC COMMANDS

1. git init (to initialize a git repo)
2. git status (tells you current status of git repo)
3. git add filename.py | git add . (move work to staging area)
4. git commit -m "commit message"
5. Create a .gitignore file and add files/folders that doesn't need tracking.

SEEING COMMITS

1. git log (see details for all the commits)
2. git log --oneline (1 liner details for all commits)
3. git log --stat (details of changes on commit level)
4. git log -p (see code changes on commit level)
5. git show <6 digit sha id> (see changes on specific commit)
6. git diff (changes made within staging area before commit)

BRANCHING AND MERGING

1. git branch <name> <sha id>(creates a new branch)
2. git branch (lists all branches)
3. git checkout <branch> (switch to another branch)
4. git log --oneline --all (see commits of all branches)
5. git log --oneline --all --graph (graph of commits from all branches)
6. git branch -d/-D <branch> (deletes a branch)
7. git merge <branch> (run from master/main)

WORKING WITH REMOTE REPO

1. git remote add origin <url> (declaring remote repo)
2. git push origin master/main (sends local repo to origin)
3. git pull origin master (pull code from remote repo)
4. git clone <url> (download/copy remote repo)

UNDERSTANDING SOFTWARE VERSIONS -> V.1.0.0

1. git tag -a v1.0.0 <sha(optional)>
   Vim editor:
   Fn+insert – write a message
   :wq – save and quit
2. git log (check if tagging was done successfully)
3. git tag -d v1.0.0 (delete the mentioned tag)

Git

→ Git is a VCS/SCM i.e, Version control System & Source code manager respectively

→ Download from git-scm.com

Some commands

→ To open VS code from your local pc workspace.
So, Left or right click to open bar, then either select VS code or come to cmd / bash and enter "code ." to open it in VS code.

workspace → At this stage git has no access to work. work is in local place.

So, in initial stage, as no git is involved. so when

I/p ⇒ git status O/p ⇒ not a git repo.

I/p ⇒ git init O/p ⇒ Initialized empty git repo.
This step is done to enable git to access and supervise the work.

I/p ⇒ git status O/p ⇒ on branch master.
No commits
Untracked files (x.py, y.py, etc)

I/p ⇒ git add.
add all untracked files to staging phase → This helps git to track the works & progress.

O/p ⇒ No commits yet (so, now we are ready to commit all the changes / work to a local repo as workspace is now empty and all files are in staging phase ready to be committed locally in our system.

I/p ⇒ git commit -m "EDA done".
This message is written as in future, our work may fail and we want to return to our previous work and again continue from there, so this commit message helps to track that work and Extract history to resume from there.

Now, if we change anything like add or modify code in file x.py. So again "git add first.py" or used "git add ." if multiple files are modified or added. Now commit again. Now if we want to roll back to original work in our workspace as if we have made changes unknowingly then

" git restore --staged x.py " - It will discard changes

→ whenever "U" symbol in front of files in left bar of VS code, it shows that these are untracked or unstaged files.

→ If we have some details which we not want to push in public repo or not want git to track it, so, make .gitignore file and add names of files such as delete.txt or etc in which useless / sensitive info is stored and add & commit .gitignore file.

→ git assigns a unique id called as "sha-id" to each commit and using this id, we can move back to any commit-(version of work)

→ git checkout "sha id" → to roll back at version x

→ git checkout master → to again come back to same step.