Date :

Coursera Courses.
Version Control System

VCS To rollback code.
Complex project that kept changing directions, keeping historical copies

- lets you see progress of changes over time

- These files can be anything code, config or images.

Diffing files 2 copies of same code and see what difference is between two?
Soln - We can use diff. from commandline
diff → when we call this we see different lines in both files

1) diff (file1, file2).
By highlighting what's changed, it helps us understand the changes and see how the files have been modified.

2) wdiff - highlights, changed words
Eg more meld, kdiff3

Date :

Applying Changes diff -u oldfile newf > change.diff.

change.diff is a file that contains these changes b/w old file & newfile and additional content needed to apply changes.

patch - command that applies those differences made to original file.

PRACTICAL EXAMPLE DIFF & PATCH. ✓

VCS We can make edits to multiple files and treat that collection of edits as a single change which is commonly called as COMMIT.

Also author of commit to tell why commit was made?

VCS also stores history of code and config of your files

Date :

What is GIT?

2005. By Linux Torvas.

open source. Git has distributed Architecture. You can use Git with or without network connection. Distributed VCS means each contributor has a full copy of a repository, they can interact with tracked files without needing to coordinate server.

git-scm.com → acronym for VCS → Source control mgt.

Other Version Control: Mercurial, Subversion

Linux apt install git ? for using from commandline.

git config --global user.name "muskan"
git config --global user.email "_____"

git init - When we run this, we initialize an empty git repository in current directory .git

We can check this with ls -la command
ls -la .

Date :

ls -l .git/ command to see what's inside .git file (DB for your project)

Whenever you clone your project, this git directory is copied to your project.

The files outside git is a working tree that acts as a sandbox where we can edit the current versions of the files.

Staging area (Index) ← git add file to

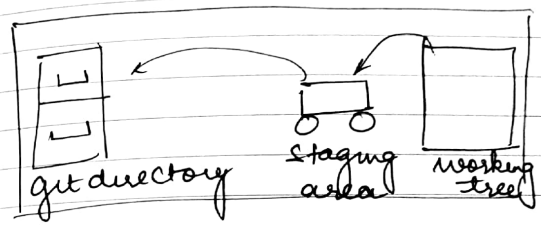
A file maintained by Git that contains all of the information about what files and changes that are going to your next commit

git status - To get info about current working tree and pending changes.

git commit → enter commit msg.

Date :

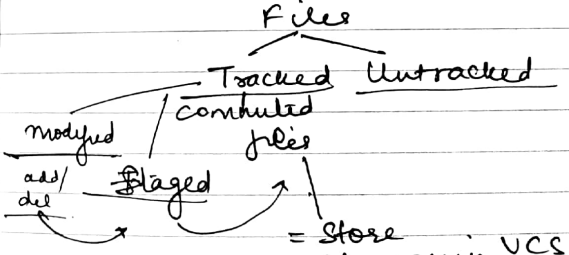
Tracking Files



git directory - history of all files and changes

working - current state of project including any changes we've made.

staging area - changes that have been marked to be included in next commit -



git commit -m 'Add ____'
↑ msg.

git config -l - see all your info. useful when you have a lot of people and it tells you have made change

Date :

GIT WORKFLOW

1) Make changes to our files
2) Stage them with git add
3) And commit that

ANATOMY OF A COMMIT MSG

1) Write better description.
Keep audience in mind. Sections
First line - Short Summary of Commit, followed by blank line
— BLANK LINE — each line
Full description of changes (72 chars).

git log - gives info about author of each commit, timestamp and each commit message.

Using Git Locally

1) Skipping the Staging area
We can (commit -a) - A shortcut to stage any changes to tracked files and commit in 1 step.

Best reserved for making small changes
commit -a -m "msg".

When you git log head will the top most thing. head is used to indicate what is currently marked snapshot

Getting More Info About Changes

git log -p. — exit with q.
git show (info about commit & associated patch
git Log : —
--stat → which files were changed and what changes are made

To remember everything you did b/f commit git gives git diff command

Tells what changes have been there

git add -p - when we use this flag, git will show us the change being added and ask us if we want to stage it or not. This way we can

Date :

detects if there is any change we dont wanna commit.

Deleting / Renaming Files

you can remove/delete files from rep. by (git rm) filename.py.

Then ls -l
Then git status
git commit -m" ——— ".

git mv → to rename file
git mv old name new name

UNDOING CHANGES B/F COMMITING

git checkout → to undo previos changes
→ -p → this flag doesnt show entire file but the previous change that was made.
↑
undoing unstaged changes

git reset → to undo staged changes
git reset HEAD " " ——→ file name

Amending Commits

update commit to make change
--amend
git commend --amend → take files from the staging area and overwrite the previous commit. (removes previous commit & overwrites.
Setting up a local rep is ok but avoid doing it in a public repository.

Rollbacks

revert → inverse the changes made in a bad commit
git revert HEAD. removes lines that we added in previous commit
Best option for undoing commits on public branch
commit ID - 40 char log.
It is essentially hash on SHA-1.0 which is used to guarantee the consistency of our repository.

Date :

if
detect there's any change we dont wanna commit

Deleting / Renaming files

git remote add origin
↳ To set that directory as origin

Branching & Merging

Branch → pointer to particular commit. Indep. line of development

Def Branch → Master

git branch - shows all branches in repo.

Create new - git branch <newb>
current branch is indicated with asterick

git checkout to checkout latest snapshot for files & for branches.

to switch - git checkout newb.
single step —

git checkout -b nameofnewbranch (NONB)

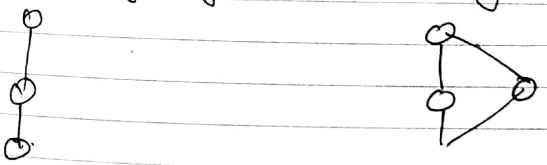
When we checkout a new branch and commit on it, those changes will be added to the history of that branch

delete branch we dont need.
git branch -d newbranch

Merging - combining branch data and history together:
From master
git merge newbranch

2 ways merging

fastforward 3 way merg



Merge conflicts Normally git automatic merges but sometimes we have

Date :

master
Same file 2ch
commit

branch
same file 2ch
Commit

merge conflict

a merge conflict

git log --graph --oneline
↳ shows graph.

To fix error -
open file from terminal only

git branch -D - forcibly delete branch

git merge --abort - abort merge in case of conflict

Working with Remotes

Intro to Github.

① First step is to create local copy of repo use git clone ~~Change~~ It has created the ~~dir~~ a local rep. as a directory

Date :

② cd TOP
ls -l → it is empty only has readme.md - markdown

subl.
git commit -a -m

git config --global credential.helper cache → so that we dont have to put our pass again & again

Scm Providers — Github
~~Bitbucket~~
Gitlab

increased privacy, control
Customization
http → gen used for readonly access clone & pull repo
https & SSH - method of authentication you decide who can push

git remote -v

git remote show origin → to see all info about repo.

git branch -r → to see branches that our repo is handling
Readonly.

Date :

If we want to make a change to the remote branch, then
- pull remote branch
merge with local branch
push back to origin

Fetching New Changes

git remote show origin.

git doesnt keep remote & local branches in sync automatically
~~It waits to move~~
git fetch → fetched content mirrored to local repo.

git status (in this case branch is behind)
Merge → merges local branch & master online repo.

Diff b/w fetch & pull.

git fetch fetches ~~repo~~ remote updates but doesnt merge, git pull fetches remote update & merges.

Date :

git fetch - to see changes
then git merge - to see if we want that in our repo

To get content of branch without automatically merging
- git remote update

>>> } conflict markers.
<<< } - should be removed b/f merge

git check -b branchname - create + checkout branch
to move a branch to git —

git push -u origin branchname
this also sets up check/update the branch on our local repository.

Rebase - Move current branch on top of this branch
Prevents 3 way merges.

Date :

Collaborating

→ Click on pencil.
We create a fork
It will create a new branch so we would have to send a pull request.

→ Pull - A pull req is a commit or series of commits that you send to owner of repo so that they incorporate to their tree.

→ Make a change proposal.

Create pull request.

fork → Clone → Local Copy
Someone's proj.
↓ Make Changes.

Date :

Code Reviews

going through someone else's code, documentation or configuration & checking that it all makes sense & follows expected patterns.

Coder V tools - help us comment on someone else's code.

→ Better var names
→ Comments + Documentation
→ Large code → small fnc.

issue tracker, IRC channel, Slack.
Bug Tracker Issue Tracker
Bugzilla.

Go to issues and there you will find.
assign issues on yourself

Travis - use for continuous integration.

Artifacts, pipelines Travis-ci.com.