GIT

INTRO

Creating git repositories

git init => initialize empty repository

Master is the primary branch

git status => display state of working directory

configure git

git config –global user.name “FIRST\_NAME LAST\_NAME”

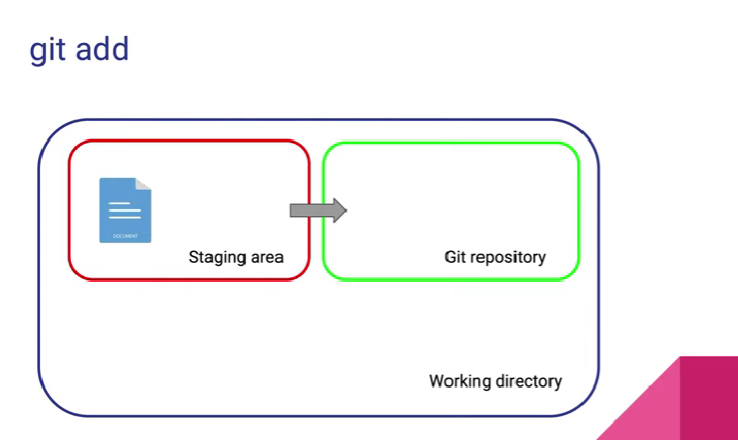
git config --global user.email "MY\_NAME@example.com"

committing project

a commit is a snapshot of the repository at a point in time

steps to create a commit:

* create a new file or modify an existing one
* stage the file by running git add <FILE NAME>
* commit the file by running git commit -m "Message"



Git staging

the staging process allows us to select which files we want to commit

to add all changes to a commit you can use git add . or git add –all

unstaging changes

if you wish to unstage a file, you can use the command:

git reset HEAD <FILE>

viewing changes with git log

use the git log command to view a list of changes

use git log --patch for a more detailed history

commit folders

you cannot commit an empty folder with Git

to have a folder tracked by Git, add an emtpy file called .gitkeep

.gitkeep is just a convention, not a rule.

Deleting files

rm filename => file

rm -rf -- foldername => folder

after deleting we need to add and commit the changes

the .gitignore file

not all files need to be stored in Git but need to be part of the folder used with Git

eg : node\_modules and .env files

creating a branch

branching helps you divert from the main development work

branches are often prefixed with feature, bugfix, hotfix etc.

create a new branch with: git checkout -b <BRANCH NAME>

to switch back to master/main run: git checkout <BRANCH NAME>

to see list of branch : git branch -a

Deleting a branch

Git branch -D <branch\_name> => force delete

Git branch -d <branch\_name> => normal delete

Merging branch

you have to be inside the branch where you want to have the result of the merge

the command you use is git merge <BRANCH YOU WISH TO MERGE>

verify the command output says "fast-forward merge"

use git log to view the commits added

A diagram of a network

Description automatically generated

A diagram of a diagram

Description automatically generated

Advanced merge

Recursing merge :

if you continue making changes in the master branch after the point in time the branch is created, a fast-forward merge is no longer possible.

git merge will try to merge the branches even if they have different parents

To keep history clean use rebasing commit :

we often try to avoid merge commits to keep the history clean

you run rebase on the branch you want to sync with the master/main branch

git rebase master

Merge conflits

merge conflicts are one of the most annoying things in Git

they are a normal occurrence but they still cause a lot of frustration.

you can abort a merge: git merge –abort

you can abort a rebase: git rebase –abort

Git with Gitlab

Working with remote repositories

Git is a distributed versioning system

We can have local repositories, but we can also have remote repositories

create new repo

origin <- alias for the remote repository

commits are not automatically sent to the remote repository

to push changes use git push

example for pushing the master branch: git push origin master

git pull

git pull origin master

rebasing while pulling changes

git reset –hard origin/master => for pulling the code from remote without the local conflit data

git pull origin master –rebase => for auto merging and clear history

cloning remote repository

git clone [git@gitlab.com:abinesh-rv/working-with-git.git](mailto:git@gitlab.com:abinesh-rv/working-with-git.git)

course notes

https://gist.github.com/vdespa/975302dec670736df2299792cebbe8cb