git

git --version

git config --global user.name <username>

git config --global user.email <emailId>

To check

git config user.name

git config user.email

cd ..

cd Shaun

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ls or dir

mkdir test

touch index.html

(To create a file )

rm index.html (To remove a file)

rmdir test (To remove a directory)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

git init

It is kind of initializing the git. The .git folder can some files. So all the files under this .git file will be reviewed by

this .git file like any changes or deletion likewise.

Create repository and inside this there is a .git f

Lets see staging files in git

To check the status of file which we have changed and which files are in staging area

git status

index.html(RED COLOR)

Now to move these to staging area

git add <filename>

git add index.html

Again lets check the status by git status after adding index.html to staging area

git status shows now index.html(GREEN COLOR)

If by mistake I have staged any file and want to UNSTAGE that file again

git rm --cached <filename>

For example git rm --cached index.html

One thing if we want to add all changed file to staging at once.

we can use

git add .

Now we can commit our changes.

Now the question arise why we are using staging area why we cannot commit directly.

The answer is that we can review our changes before doing a final commit. It is more safe way to commit changed.

Also, if there are 2 different files which are changed. Now we can move 1 file to the staging area and

commit that file only . And later move the another file to staging area and do a commit later. So the staging helps

to do a separate commit of the files which we have changed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now let’s do a git commit to the files by using below command

git commit -m "Your message related to the commit"

For example, git commit -m "fixed bug in the header"

m is for message it gives us a brief idea on how the git works. Like something descriptive

So, for example it says 2 files changed and 13 insertion.

13 lines of added are added.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I want to see all the history of git commits which are done before.

git log

But git log shows all the description of all the commit which is difficult to see

git log --oneline

it shows only one line for each commit

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

UNDOING THINGS which went wrong

1) Checkout commit(Safe)

git checkout b73erb5r

b73erb5r is the unique id of that commit

(THIS IS READ ONLY)

git checkout master

means it detached from the master branch

It opens in read only

2) Revert commit(Little risky)

git revert <id>

It will go to a screen where u can add a new commit.

If we do not want to change anything in this commit then do below to exit without changing anything

Shift : wq

Its like removing the commit like it never happened before

3) Reset commit(Very Dangerous)

Reset to some old commit but its like permanently delete all the commit after that old commit we want to go back

git reset <id>

this will delete all the commit after this commit but still in the id old changes are there in case we want to take backup

git reset --hard

It will delete and also remove changes from the editor also

GIT Branches

We use git branches so that we can create a new branch out of master branch and try to add new feature in the branch. If it works then we can merge this branch to the master branch otherwise we can delete this new branch. So, that is the benefit of the branches.

How to create a branch?

git branch <branch\_name>

For example

git branch feature-1

How to check all the branches ?

git branch -a

\*(Astrick) shows the current branch we are in

Below diagram show we are currently in the master branch

Text

Description automatically generated

How to switch from one branch to another branch?

git checkout <branch\_name>

For example, git branch feature-1. It will switch from masters to feature-1 branch.

How to delete the one branch?

git branch -d <branch\_name>

d for delete

But -d (Small d) ONLY WORKS if that new branch is merged with the master branch.

So in this case we will use -D(Capital D) here.

First go the master branch and then delete this new branch.

git branch -D feature-1

Text

Description automatically generated

If we want to checkout and create a new branch simultaneously then we can use below command

git checkout -b feature-1

Now how to merge a branch to the master branch.

First we have to be in the branch we want to merge into.

git merge feature-a

In case of conflict resolve it.

GITHUB

We can push to repository

git push <repository\_path> <repository\_branch>

But it is difficult to create to type repository path all the time

We can set up an alias

git remote add origin <repository\_name>

origin is alias name for repository like for example <https://github.com/abkuma/obecity.git>

How to clone?

git clone <url\_repository>

git remote -v

1)Go to master branch and pull the latest changes from the master

2)create a new branch and push the latest code of master to the this new branch

3)make your changes and push it

4) Your teammates will do the pull request and check the changes

5)After everything is fine you can merge with master branch

6) After than if you wish you can delete the new branch you have created