**Logo

Description automatically generated**

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Below you can see the Lifecycle of git and how it works.

Timeline

Description automatically generated

Now we will move on to practical

**1. Install Git:**

first, we download and install the git link is below

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

**2. Set username and email**

After successfully installing the git we open the git bash terminal and set the username and email

$ git config –global user.name Amir

$ git config –global user.email [amirali.cheema@outlook.com](mailto:amirali.cheema@outlook.com)

**3. Git Cloning**

Through cloning, we get all code from GitHub to here in local

For example, I clone some code below is an example of how to get

$ git clone address here folder name

**4. Git Initialize**

If are doing from start in local computer, then first will initialize the folder by using the command

$ git init

$ ls -lart # show all the hidden files

Text

Description automatically generated

after creating some files and changes in code (In my case it’s index.html but didn’t add it)

**5. Git Status**

Status basically the current status after the operation for example I create a file index.html in my folder and check the status and he gave the info that file is untracked

git status # tell the file is not committed yet

Text

Description automatically generated

**6. Git Add**

After that, I add the index.html file by using the command

$ git add index.html # single file add

If I want to add multiple files at the same time, then I will use the following command

$ git add -A # multiple files add

Text

Description automatically generated

**7. Git Commit**

Through commit we can push file from stagged to unmodified

$ git commit

After that it will open vim editor to type press <i> then type initial commit this is the message that I gave to after the change in the file) then from the exit from vim editor by pressing ESC:wq

Text

Description automatically generated

Text

Description automatically generated

Or I can the same thing with the different command which is easy

git commit -m “initial commit”

Now I create some file

$ git status

Text

Description automatically generated

Now I will add all files (that I create) that i using the command

git add -A

Text

Description automatically generated

Now the file are ready to commit. Now I have some changes in statics.html And see the status

Text

Description automatically generated

Now I add the statics file again

$ git add statics.html

Text

Description automatically generated

Now I commit all the files

$ git commit -m ”Added more HTML” # shortcut way to commit instead of using vim editor

Text

Description automatically generated

Now check the file status

Text

Description automatically generated

**8. Git Checkout**

For example, someone change code in file which was wrong. So, to resolve this problem we use

Like in my case file is a change like statistics.html which was the wrong code.

Text

Description automatically generated

And I want the original code which I commit last time.

$ git checkout statics.html # single file checkout

A screenshot of a computer

Description automatically generated with medium confidence

If all file changes with the wrong code, then return them back in the original form

$ git checkout -f # multiple file checkouts

Now I want all the detail of what commit who commit etc

**9. Git log**

I want log is basically gave the summary of your commit that you did

$ git log

Text

Description automatically generated

you can also filter you commit for example I want to see only 1 commit log summary then I will use the following command

$ git log -p -1

Text

Description automatically generated

**10. Git diff**

Git diff basically tell the difference between your last and current commit. In my case, I have some changes in the file of the index.html

$ git diff # it compares the new changes with the last one

Text

Description automatically generated

$ git add - A

After adding, I again change

$ git diff

Text

Description automatically generated

After adding changes files I check again

Text

Description automatically generated

Text

Description automatically generated

Now I want same code back (which was in stagged area) then I will use following command

$ git diff – staged

Text

Description automatically generated

Now back to same code

$ git checkout -f

$ git status

Text

Description automatically generated

After that, I change some in an index.html file

A screenshot of a computer

Description automatically generated with medium confidence

Now I use a shortcut way to add and commit in a single commit

$ git commit -a -m ”Skipped staged area and fixed <”

Text

Description automatically generated

**11. Remove File**

For example, I add and commit wrong then I remove in my case my file is waste.html

Text

Description automatically generated

Here two ways to remove 1st way is to remove only from a stagged area, not a hard drive 2nd remove completely

1st way

$ git rm –cached waste.html # remove from stagged only

Text

Description automatically generated

Text

Description automatically generated

2nd way

git rm waste.html # remove from both stagged area and local

Text

Description automatically generated

Another way to check the status

git status -s # provide a short summary of opr

For example, I change some code in 2 file

Text

Description automatically generated

**12. Ignore File**

First, we make create a file with the name .gitignore And inside whatever I gave the name of a file with the extension it will ignore

For example, I want to ignore folder amir then inside I write amir/ and he ignores all files inside a file Or I want to ignore a specific extension file like \*.cpp or \*.py he ignores these files as well just write inside the .gitignore file

Graphical user interface, text, application

Description automatically generated

**12. Branches**

git branches is a killer tool in git. Our main branch is master we make another branch and work on it and will not affect the master branch as well and if we merge the command then we merge our new branch with master

Graphical user interface, text

Description automatically generated with medium confidence

Currently, we are in branch master so will create another branch with name amirwork

$ git branch amirwork # create an amirwork branch

$ git branch # check the branches

Text

Description automatically generated

git checkout amirwork # it will move from master to amirwork branch

Text

Description automatically generated

Here I work some on waste files through the amirwork branch then go back on a master

Text

Description automatically generated

After finishing work on waste.html I checkout to master

A screenshot of a computer

Description automatically generated with medium confidence

For example, my waste.html work is confirmed then I merge amirwork with the master branch

Text

Description automatically generated

So, in this way branches work. It’s a convenient method to do that if work is okay then you can merge in master.

We can also create multiple branches as well

$ git checkout -b ansarwork # here I did two things first creates a branch and 2nd switch this branch as well

Text

Description automatically generated

And that’s all for git.

**13. Github**

Github is basically a web hosting service we push our code from local to the server.

In git we did everything so, it’s time to push our code. First, I create a repository on GitHub then using the below command I push my commands.

$ git remote add origin https://github.com/AmirAli5/Git\_Complete.git

If you want to change the address type $ git remote set-url then address below is example

$ git remote set-url origin [git@github.com:AmirAli5/Git\_Complete.git](mailto:git@github.com:AmirAli5/Git_Complete.git)

Now I will push using this command

$ git push -u origin master

Here I push using the origin master branch which is the default. Now I change the branch and push the code same repo

Text

Description automatically generated

$ git push -u origin amirwork

Here I push code from the second branch. We can push multiple branches. We can see the code of each branch code. And commit as well. That’s all.

Happy Learning😊