DevOps is a set of methodology or set of rules

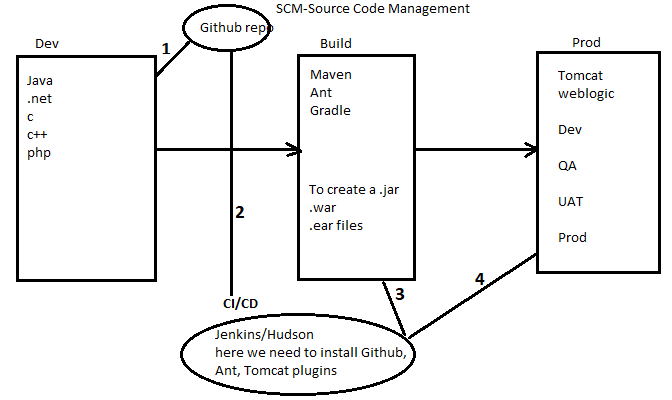
DevOps :- it reduces the gap between the developer’s activities and operational activities.

Devops is the software development method and communication, collaboration and integration between the developer activities and operational activities.

**Why DevOps:**

* It’s a automated process
* Where each and every activity is a predefined with set of rules and regulations
* Like where we have to store the code, when we have to build the code and where we have to deploy the code and how many places we have to deploy the code

**Who involves :** Developers, Build Team, Release Team, System Admins and QA team

.

.java file----->.class file----> .exe file(.war, .ear file)---> deploy this file in linux servers

Step1: code is stored into a github repo

Step2: when the code get updated in the github. Jenkins plugin(respective to tool like tomcat,

weblogic) based tool clone the code from github.

* Configure details like github(url, credentials), maven(path, environment variables),

Tomcat or weblogic or jboss server(url, username, password) in plugin

* Job creation: which git repo, which build tool, which server will configure in job

Step3: now build(test cases, phases) then it creates a executable file(.war, .ear)

Step4: deploy this executable file in the servers(dependency job created process one by one)

Step5: at last it will generate the notification mail

SCM tools: SVN, GIT, Perforce, CVS, TFS

Build tools: Ant, Maven, Gradle, Apache Builder, visual build

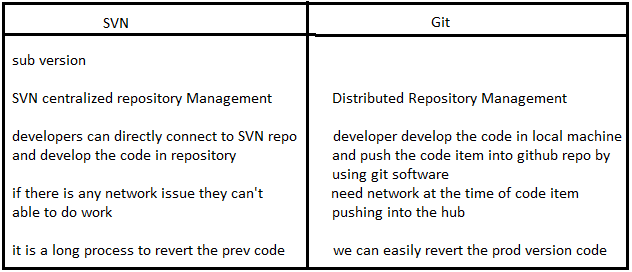
Servers: Tomcat, JBoss, Weblogic, websphere

CI/CD tools: Jenkins, Hudson, Honeypot, Bamboo

Cloud: AWS, IBM cloud, Microsoft Azure, Oracle Cloud

Configuration Management tool: chef, puppet, Ansible

**Tool1:** SCM: Source Code Management tool or Version Control tool(SVN, Git(UI, commands))



**How to install Git?**

* To install Git click on this link https://git-scm.com/downloads Download as per your OS and install.
* After installing Git open Git Bash.
* Create a New folder and enter into that dir.
* First we need to initilize the git repository

-> git init

* After that we need to configure user name and email id

-> git config --global user.name " pavan "

-> git config --global user.email " pkumar.datascience@gmail.com "

-> git config --list ( here we can list all configurations )

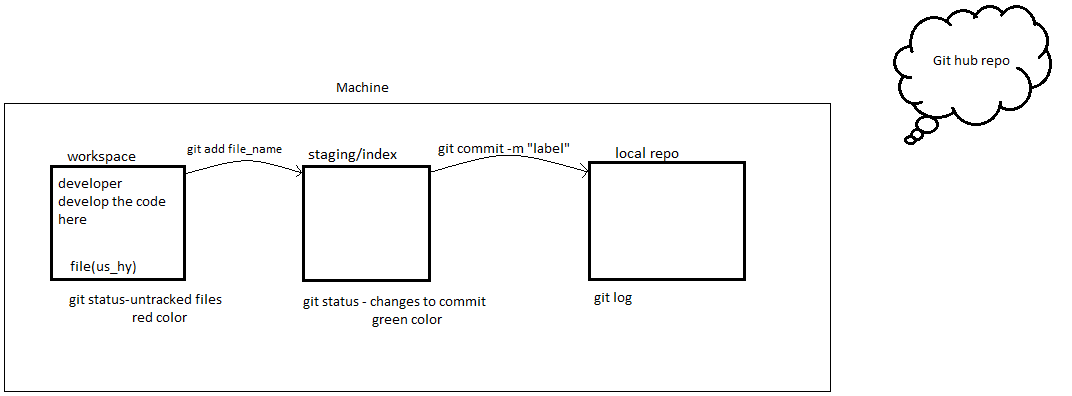
* Create some sample files by using Touch or Vi or Cat commands

Ex: Touch file1

vi file2

cat > file3

**Git having three stages:**

****

when we are creating the files it will under workspace area

**git status** 🡪 Now it will shows all files in workspace or index or local repo

**git init** 🡪 to initialize the git repo(.git will create)

**ls** 🡪tolist the files in directory

**ls –a** 🡪it displays .git directory

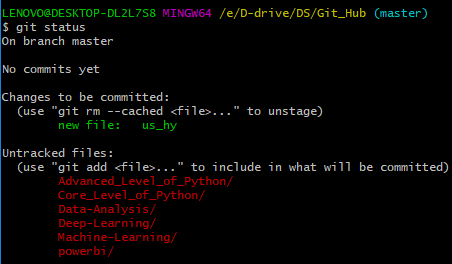
To configure user name and email (By default branch is master)

* git config --global user.name “pavan”
* git config --global user.email “pkumar.datascience@gmail.com”
* git config --list(to check whether the username and email are configure)

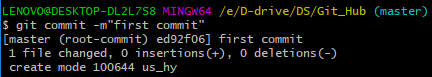
touch data.txt 🡪 to create empty file

git add data.txt 🡪 to promote code from workspace to staging area

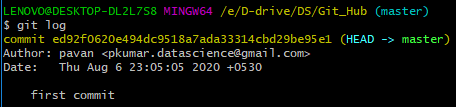
git status 🡪 to check the files



git commit –m”first commit” 🡪 to commit the code from staging into local repo

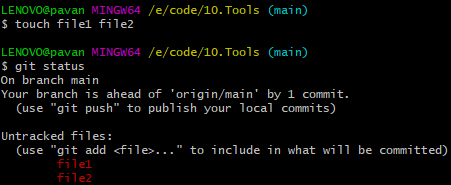


git log 🡪 to check files in local repo



git push 🡪 to push the code from local repo to central repo

**1. Create two files file1, file2**

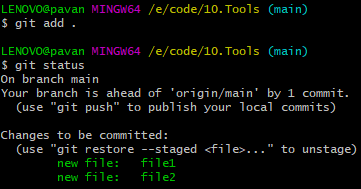


**2. move the files from workspace to staging area**

git add file1 file2 (or) git add . (or)

git add \* (or) git add –A 🡪 to add multiple files into staging area

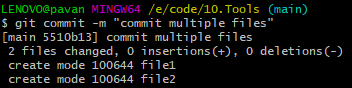
git add file1 🡪 to add single file



**3.move the files from staging to local repo**

git commit –m “commit file2” file2 🡪 to commit single file

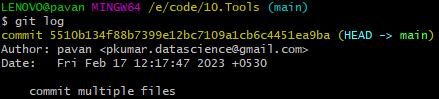
git commit –m “multiple files” 🡪 to commit multiple files



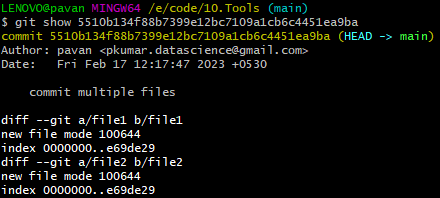
git log

Note: if we commit multiple files in single commit then we will get only single commit id

Ex: if I commit 10 files at a time then only one commit id will generate

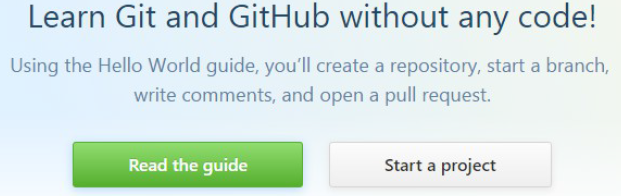


git show <commit\_id > 🡪 to see the how many files in local repo

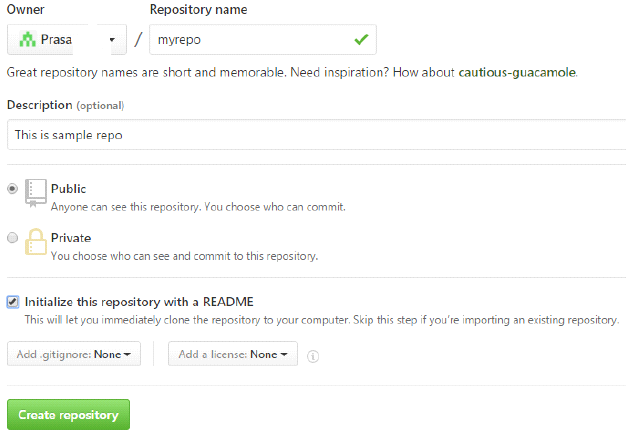


**Topic 2 : How to creatre Github repository**

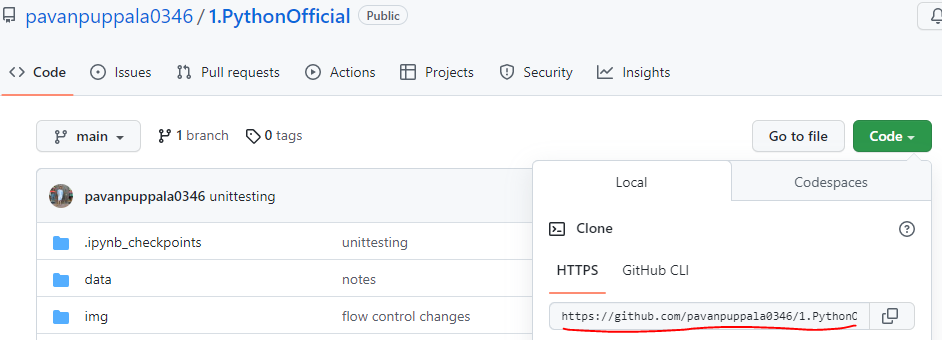
* https://github.com/ create a github account
* Now login to github account
* Click on start project option see the below snapshot.



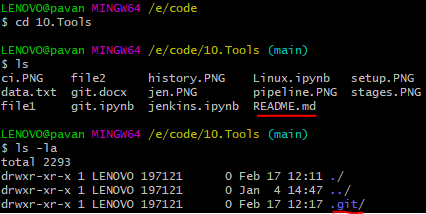
* Give the repository name and click on initialize this repository with a README.
* Click on Create repository.



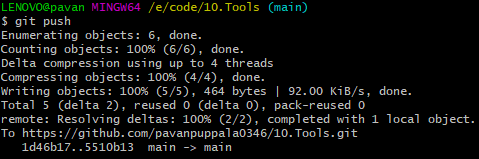
* Click on Clone or download tab and copy the path



* Now open git, create a new folder like stash and enter into stash dir
  + - git clone path/of/github/repository
    - Ex:- git clone <https://github.com/pavanpuppala0346/1.PythonOfficial.git>



* Enter into stash directory. Now here create some sample text files. This all files Add and commit
  + - git push path/of/github/repo Branch name (by default Branch name is master)
    - ex:- git push <https://github.com/pavanpuppala0346/1.PythonOfficial.git> master

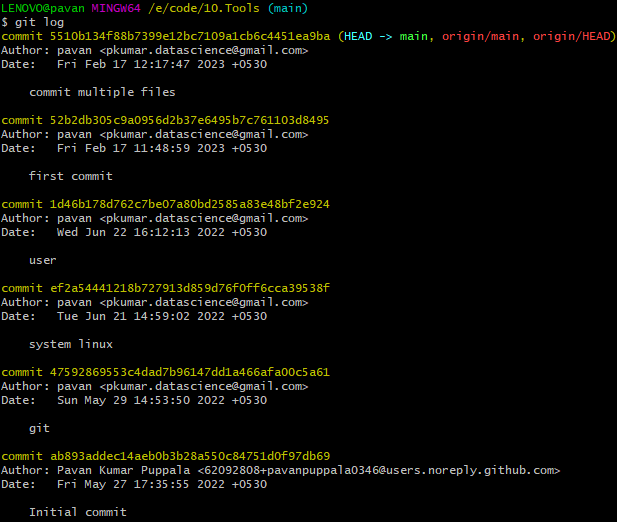


* Now go to github and refresh it, here it will displays all files

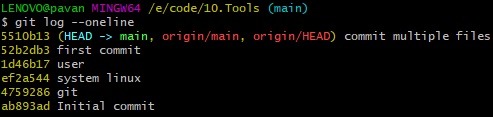
**push code from local repo to centralized repo**

* create machine-learning/ repo
* git clone <https://github.com/pavan-puppala/machine-learning.git>
* ls
* cd machine\_learning/
* ls
* ls –la
* touch f1 f2 f3
* git add .
* git status
* git commit –m”commit message”
* git status
* git log
* git push 🡪 to push from local repo to centralized repo

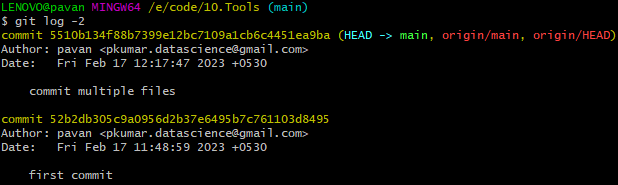
git log



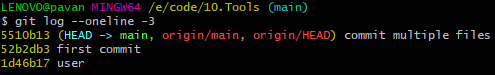
git log --oneline 🡪 each commit\_id with 7 characters and label message



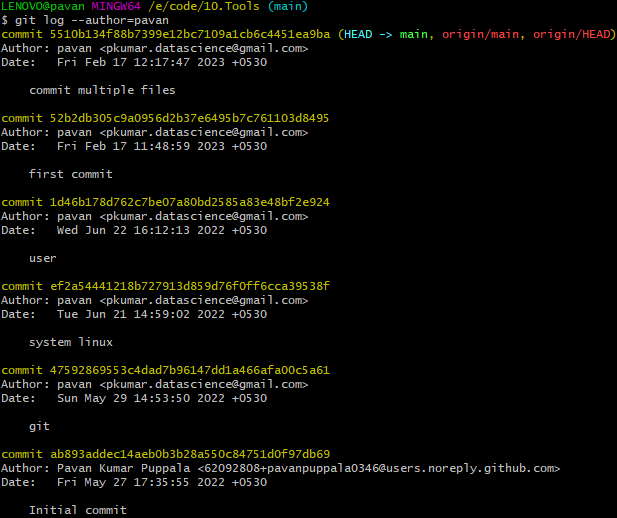
git log <–n> 🡪 two latest logs



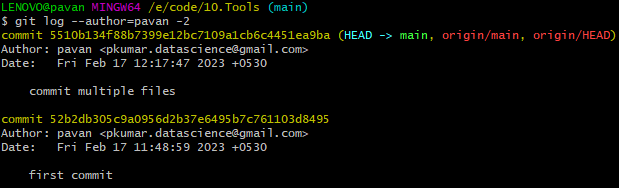
git log --oneline <-n> 🡪 three latest commit\_id’s with 7 characters



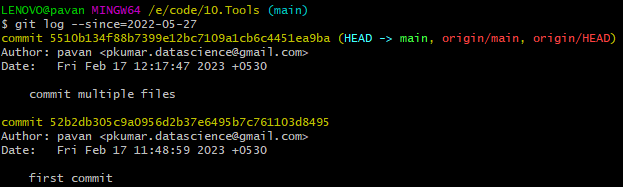
git log --author=pavan 🡪 to check the logs by author name wise



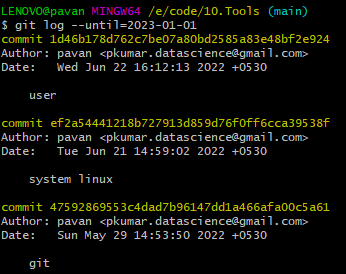
git log --author==pavan -n 🡪 two latest logs by author name wise



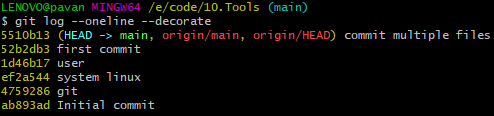
git log --since=yy-mm-dd



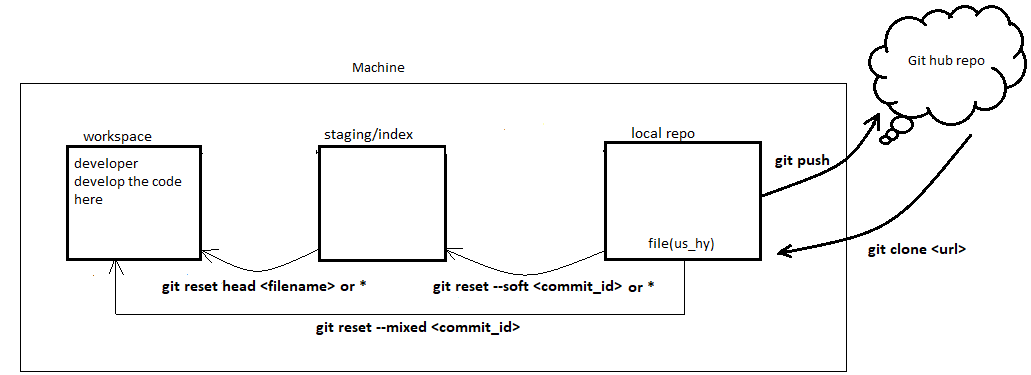
git log --until=yy-mm-dd



git log --oneline –decorate 🡪 it returns logs and with branch name



**Now how to roll back the commits from Local repository to staging to workspace**



Commit\_id1

Commit\_id2

Commit\_id3

Commit\_id4

So if you want to roll back the commit changes of 1 we have to give the commit\_id2 to command

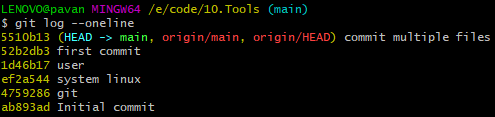
So if you want to roll back the commit changes of 2 we have to give the commit\_id3 to command

So if you want to roll back the commit changes of 3 we have to give the commit\_id4 to command

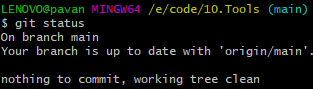
So if you want to roll back the commit changes of 4 we have to give the readme commit\_id2 to command

git log

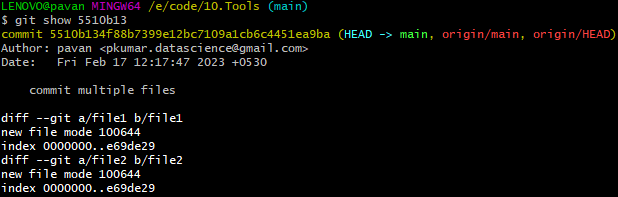
git log - - oneline



git status



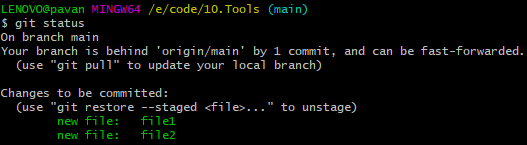
git show <commit\_id>



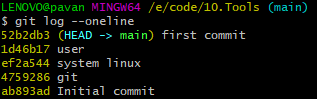
git reset - - soft <commit\_id>



git status



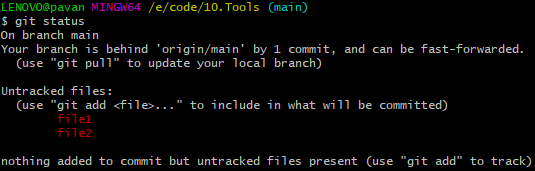
git log - - oneline



git reset head <filename>

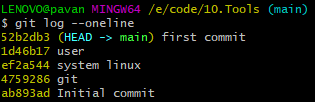


git status



**Now how to roll back the commits directly from Local repository to workspace**

git log - -oneline



git show <commit\_id>

git reset - -mixed <commit\_id>

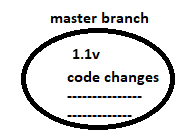


git log - -oneline

git status

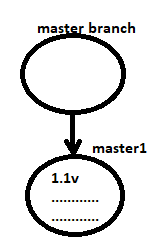
**Topic 3 : How to create Branches**

**Branches:** If you want to revert back the code again you have to clone from the centralized repo



To overcome from above line

We need to create another branch then do code modification in master1



Default branch is master:

ls -la

git branch



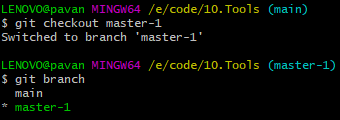
git branch master-1 🡪 to create new branch



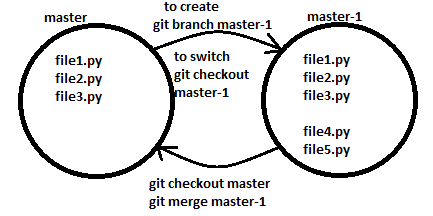
git branch or git branch - -list



git checkout <branch name>🡪 to switch the branch

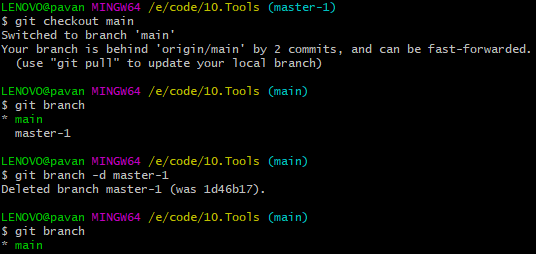


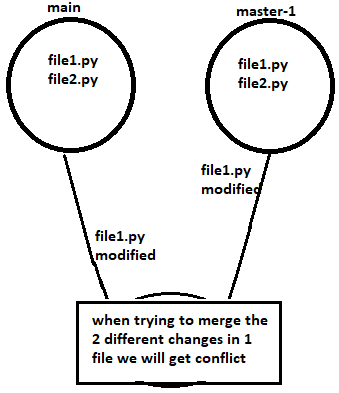
Note: whatever commits present in master branch all commits are copied to master-1

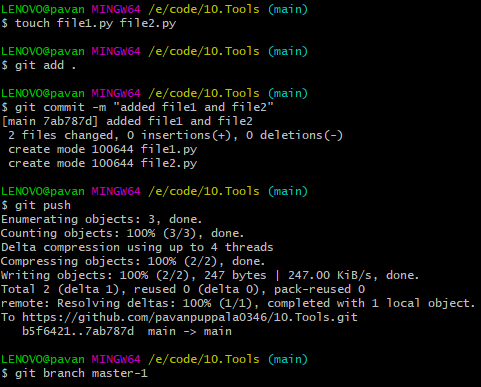


**How to delete a Branch**

* Before deleting branch we need to switch to different branch
* git checkout <main branch>
* git branch
* git branch -d <branch name>
* ex:- git branch -d master-1



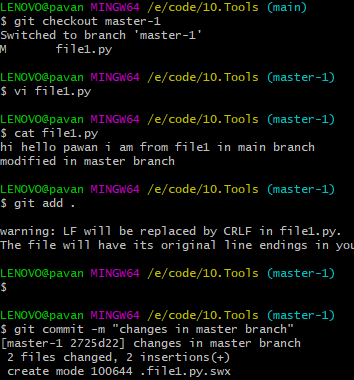
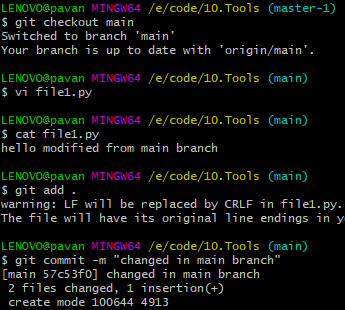
**Topic 4: conflicts**

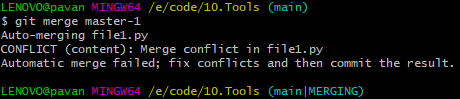


Modified file1.py in main branch



Switch to master-1 branch: git checkout master-1

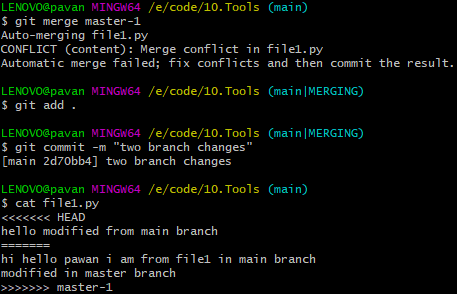




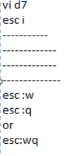
git merge –abort 🡪 To come out from the conflicts



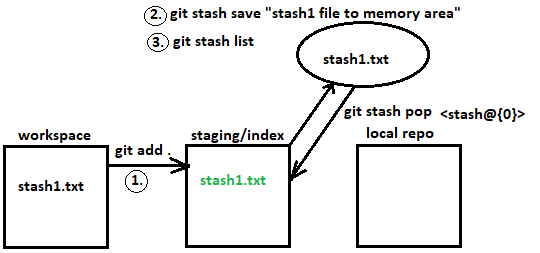
Solution:

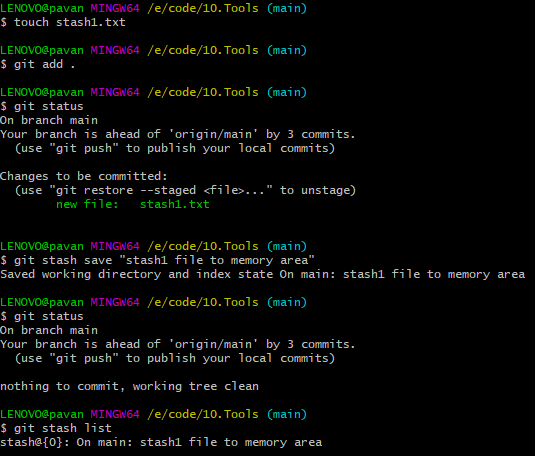


**Note:**

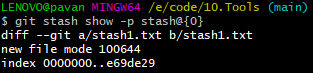
* touch file 🡪 to create a file
* touch file1.py file2.py file3.py 🡪 to create multiple files
* vi file1.py or cat >file1.py 🡪 to edit the file1.py
* i 🡪 to insert the data
* :w! 🡪 to override the file
* :q! 🡪 to quit from file1.py
* ctrl+d 🡪 to save and quit
* cat file1.py 🡪 to read the file1.py

**Stash: it is temporary memory area provided by git**





git stash show –p stash@{0} 🡪 to see the data



**Roll back to staging area from stash area**

git stash pop 🡪 rollback and delete the all files

git stash pop stash@{0} 🡪 rollback and delete the particular file

git stash apply 🡪 rollback all files

git stash apply stash@{0} 🡪 rollback specific file

**delete:**

git stash drop

git stash drop stash@{0}

**Topic 5: alias, git architecture**

For simple commands directly we use commands like git log, git push, git pull

We have some scenario’s where we need to use long commands at that time we use **Alias**

git config - -list 🡪

git status or git s 🡪 both will give the status

**how will create alias ?**

git config - -global alias.l “log” 🡪 we created alias for log

git log and git l both will get same output

git config - -global alias.l1 “log - -oneline” 🡪 created alias for “git log - -online”

git l1

**how will remove the alias ?**

git config - -global - -unset <alias.aliasname>

git config - -global - -unset user.name

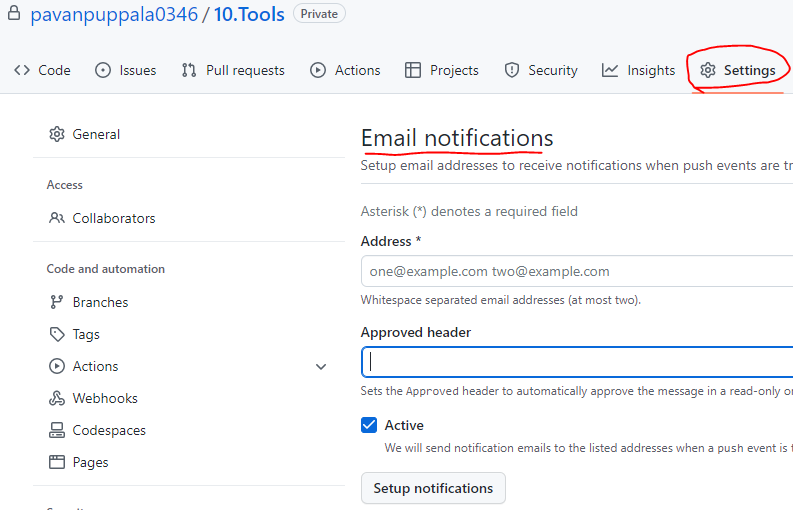
git config - -list

git commit –am “modified the file in staging area” 🡪 directly move from staging to staging

vi .gitignore

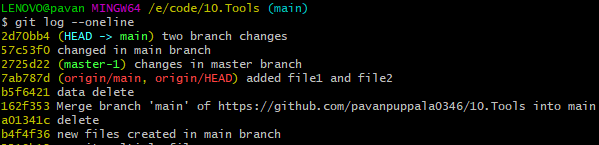
insert the filenames 🡪 to hide the files in workspace

**To get the alert options in mail**

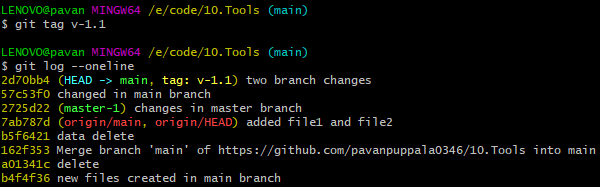


**We can add tags**

git log - -oneline



git tag v-1.1

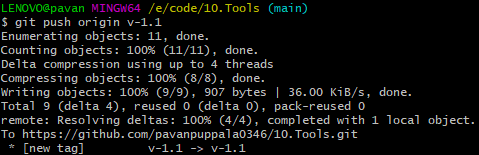


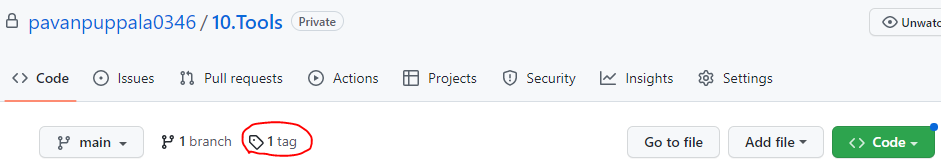
git tag (or) git show v-1.1



**Push the code to central repo**

Git push origin v-1.1

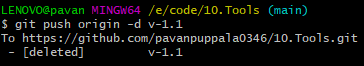


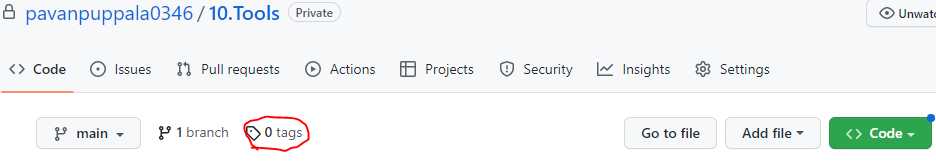


git tag –d v-1.1 🡪 to delete the tag in local repo



git push origin –d v-1.1 🡪 to delete the tag in central repo

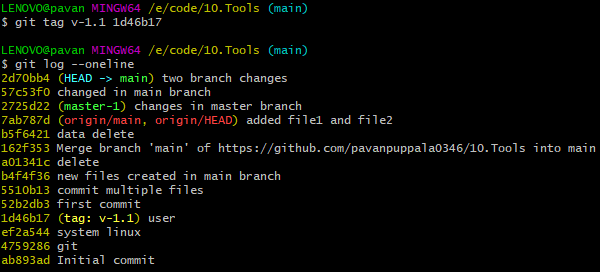




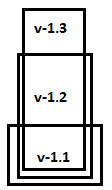
**create a tag for specific commits**

git log - -oneline

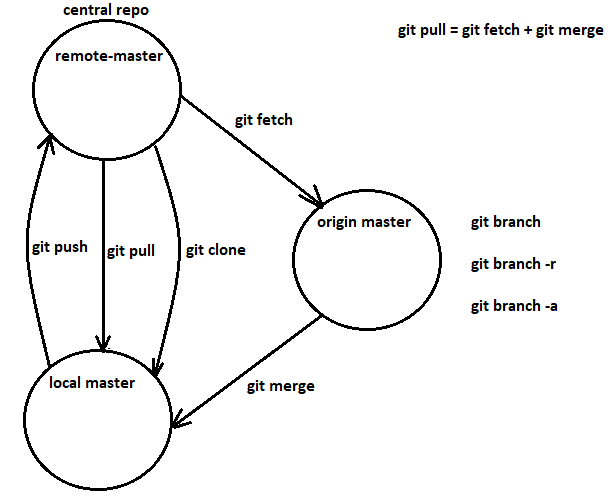
git tag <tag id> <commit id>



git push - -tags 🡪 Push multiple tags



Topic 6:



**How to rename the branch names?**

git branch

git branch –m <old branch name> <new branch name>

**How to rename the file names?**

mv <old file> <new file>