DevOps is a set of methodology or set of rules

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

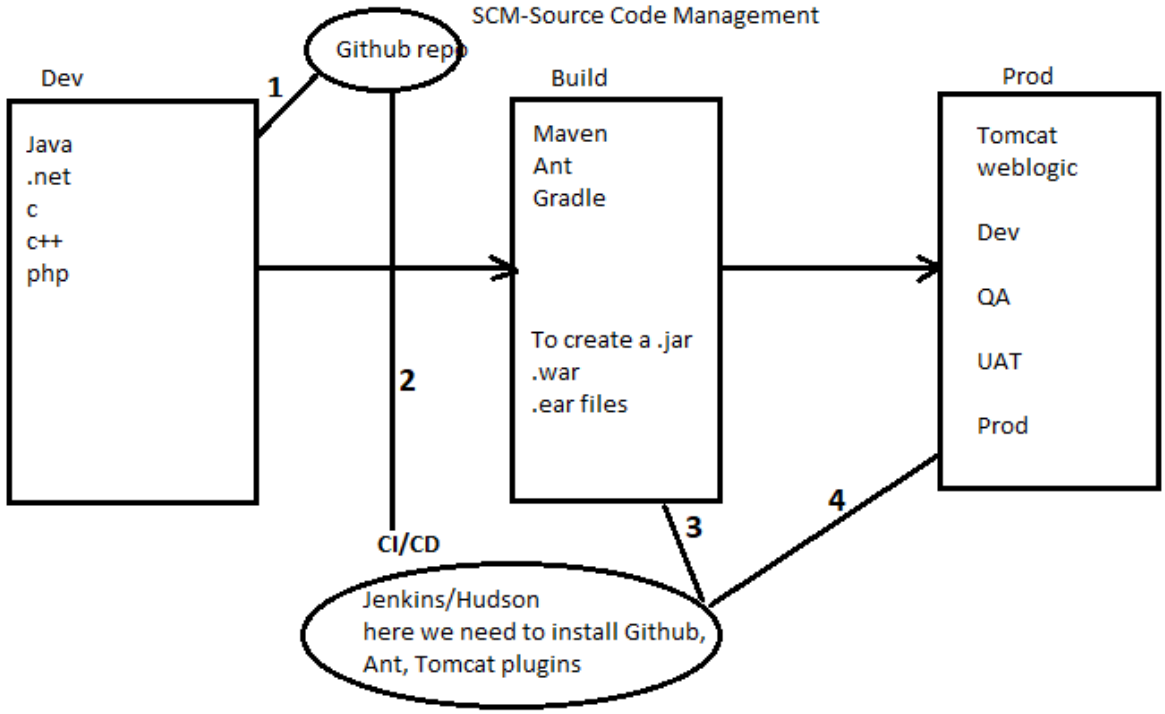
Devops is the software development method and communication, collaboration and integration between the developer activities and operational activities.

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

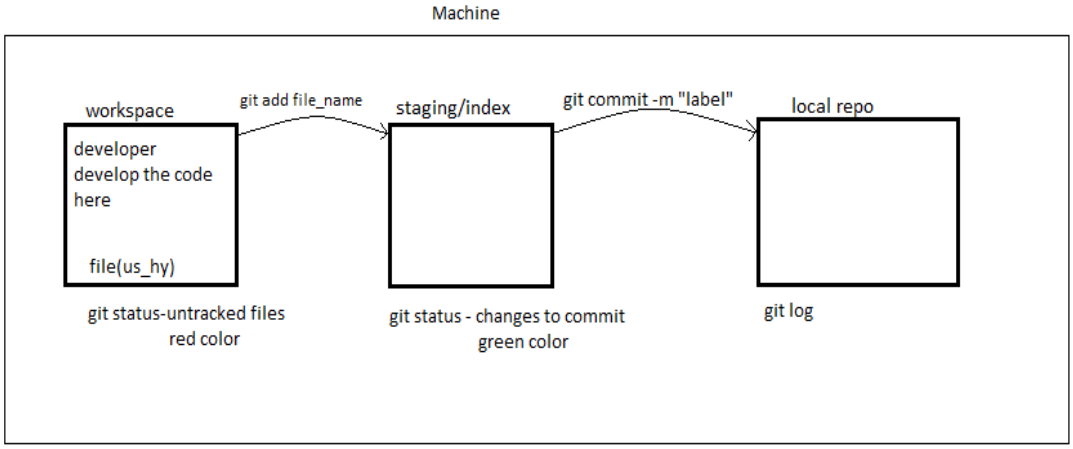
Tool1: SCM: Source Code Management tool or Version Control tool(SVN, Git(UI, commands))

SVN	Git
sub version	
SVN centralized repository Management	Distributed Repository Management
developers can directly connect to SVN repo and develop the code in repository	developer develop the code in local machine and push the code item into github repo by using git software
if there is any network issue they can't able to do work	need network at the time of code item pushing into the hub
it is a long process to revert the prev code	we can easily revert the prod version code

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 → to list 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

```
LENOVO@DESKTOP-DL2L7S8 MINGW64 /e/D-drive/DS/Git_Hub (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   us_hy

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Advanced_Level_of_Python/
        Core_Level_of_Python/
        Data-Analysis/
        Deep-Learning/
        Machine-Learning/
        powerbi/
```

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

```
LENOVO@DESKTOP-DL2L7S8 MINGW64 /e/D-drive/DS/Git_Hub (master)
$ git commit -m"first commit"
[master (root-commit) ed92f06] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 us_hy
```

git log → to check files in local repo

```
LENOVO@DESKTOP-DL2L7S8 MINGW64 /e/D-drive/DS/Git_Hub (master)
$ git log
commit ed92f0620e494dc9518a7ada33314cbd29be95e1 (HEAD -> master)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Thu Aug 6 23:05:05 2020 +0530

    first commit
```

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

1. Create two files file1, file2

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ touch file1 file2

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git add .

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file1
        new file:   file2
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git commit -m "commit multiple files"
[main 5510b13] commit multiple files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1
 create mode 100644 file2
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files
```

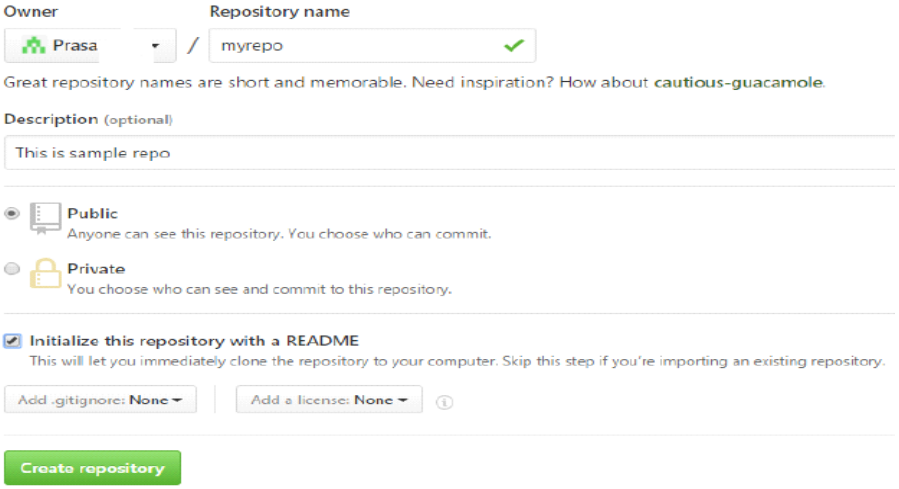
git show <commit_id > → to see the how many files in local repo

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git show 5510b134f88b7399e12bc7109a1cb6c4451ea9ba
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

diff --git a/file1 b/file1
new file mode 100644
index 0000000..e69de29
diff --git a/file2 b/file2
new file mode 100644
index 0000000..e69de29
```

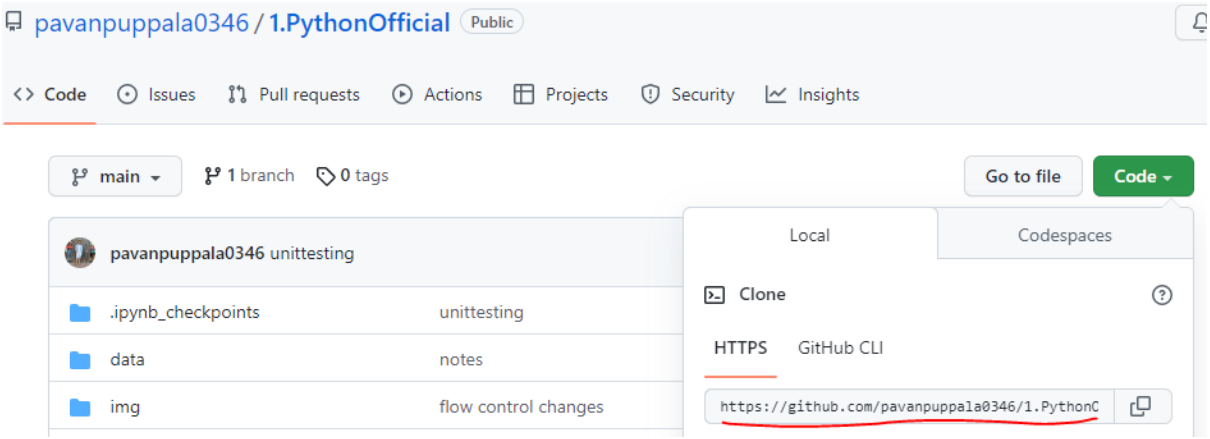
Topic 2 : How to creatre Github repository

- 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

```
LENOVO@pavan MINGW64 /e/code
$ cd 10.Tools

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ ls
ci.PNG     file2      history.PNG    Linux.ipynb    setup.PNG
data.txt   git.docx   jen.PNG        pipeline.PNG   stages.PNG
file1      git.ipynb  jenkins.ipynb  README.md

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ ls -la
total 2293
drwxr-xr-x 1 LENOVO 197121      0 Feb 17 12:11 ./
drwxr-xr-x 1 LENOVO 197121      0 Jan  4 14:47 ../
drwxr-xr-x 1 LENOVO 197121      0 Feb 17 12:17 .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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 464 bytes | 92.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/pavanpuppala0346/10.Tools.git
   1d46b17..5510b13  main -> main
```

- 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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

commit 52b2db305c9a0956d2b37e6495b7c761103d8495
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 11:48:59 2023 +0530

    first commit

commit 1d46b178d762c7be07a80bd2585a83e48bf2e924
Author: pavan <pkumar.datascience@gmail.com>
Date:   Wed Jun 22 16:12:13 2022 +0530

    user

commit ef2a54441218b727913d859d76f0ff6cca39538f
Author: pavan <pkumar.datascience@gmail.com>
Date:   Tue Jun 21 14:59:02 2022 +0530

    system linux

commit 47592869553c4dad7b96147dd1a466afa00c5a61
Author: pavan <pkumar.datascience@gmail.com>
Date:   Sun May 29 14:53:50 2022 +0530

    git

commit ab893addec14aeb0b3b28a550c84751d0f97db69
Author: Pavan Kumar Puppala <62092808+pavanpuppala0346@users.noreply.github.com>
Date:   Fri May 27 17:35:55 2022 +0530

    Initial commit
```

git log --oneline → each commit_id with 7 characters and label message

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
5510b13 (HEAD -> main, origin/main, origin/HEAD) commit multiple files
52b2db3 first commit
1d46b17 user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

git log <–n> → two latest logs

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log -2
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

commit 52b2db305c9a0956d2b37e6495b7c761103d8495
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 11:48:59 2023 +0530

    first commit
```

git log --oneline <-n> → three latest commit_id's with 7 characters

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline -3
5510b13 (HEAD -> main, origin/main, origin/HEAD) commit multiple files
52b2db3 first commit
1d46b17 user
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --author=pavan
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

commit 52b2db305c9a0956d2b37e6495b7c761103d8495
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 11:48:59 2023 +0530

    first commit

commit 1d46b178d762c7be07a80bd2585a83e48bf2e924
Author: pavan <pkumar.datascience@gmail.com>
Date:   Wed Jun 22 16:12:13 2022 +0530

    user

commit ef2a54441218b727913d859d76f0ff6cca39538f
Author: pavan <pkumar.datascience@gmail.com>
Date:   Tue Jun 21 14:59:02 2022 +0530

    system linux

commit 47592869553c4dad7b96147dd1a466afa00c5a61
Author: pavan <pkumar.datascience@gmail.com>
Date:   Sun May 29 14:53:50 2022 +0530

    git

commit ab893addec14aeb0b3b28a550c84751d0f97db69
Author: Pavan Kumar Puppala <62092808+pavanpuppala0346@users.noreply.github.com>
Date:   Fri May 27 17:35:55 2022 +0530

    Initial commit
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --author=pavan -2
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

commit 52b2db305c9a0956d2b37e6495b7c761103d8495
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 11:48:59 2023 +0530

    first commit
```

git log --since=yy-mm-dd

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --since=2022-05-27
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

commit 52b2db305c9a0956d2b37e6495b7c761103d8495
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 11:48:59 2023 +0530

    first commit
```

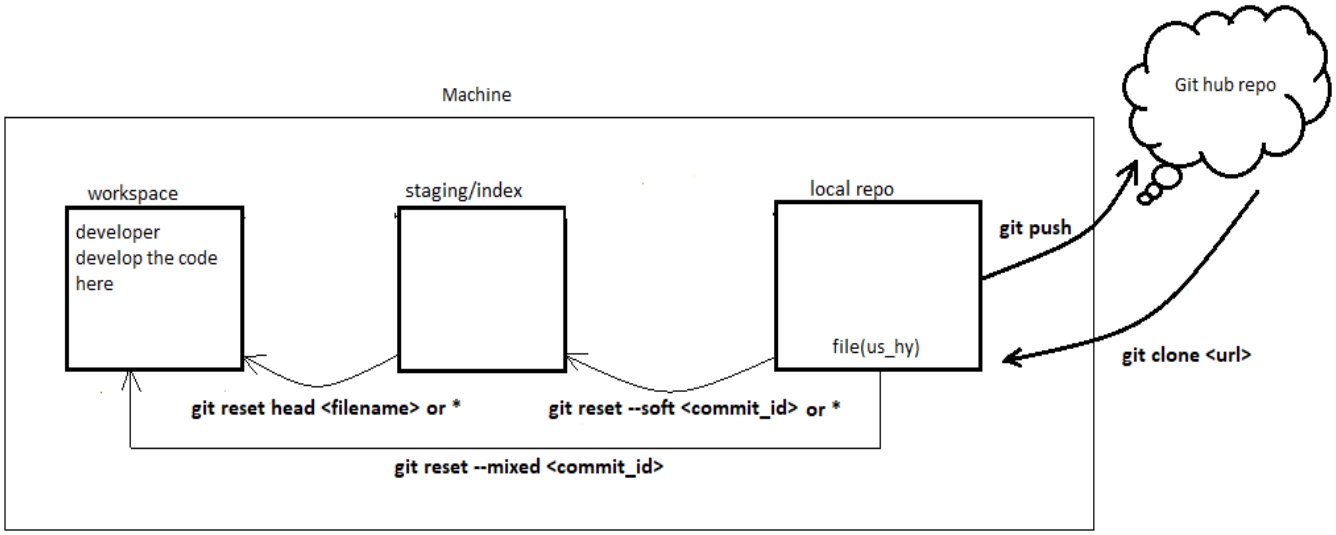
git log --until=yy-mm-dd

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --until=2023-01-01
commit 1d46b178d762c7be07a80bd2585a83e48bf2e924
Author: pavan <pkumar.datascience@gmail.com>
Date:   Wed Jun 22 16:12:13 2022 +0530

    user

commit ef2a54441218b727913d859d76f0ff6cca39538f
Author: pavan <pkumar.datascience@gmail.com>
Date:   Tue Jun 21 14:59:02 2022 +0530

    system linux

commit 47592869553c4dad7b96147dd1a466afa00c5a61
Author: pavan <pkumar.datascience@gmail.com>
Date:   Sun May 29 14:53:50 2022 +0530

    git
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline --decorate
5510b13 (HEAD -> main, origin/main, origin/HEAD) commit multiple files
52b2db3 first commit
1d46b17 user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
5510b13 (HEAD -> main, origin/main, origin/HEAD) commit multiple files
52b2db3 first commit
1d46b17 user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

git status

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

git show <commit_id>

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git show 5510b13
commit 5510b134f88b7399e12bc7109a1cb6c4451ea9ba (HEAD -> main, origin/main, origin/HEAD)
Author: pavan <pkumar.datascience@gmail.com>
Date:   Fri Feb 17 12:17:47 2023 +0530

    commit multiple files

diff --git a/file1 b/file1
new file mode 100644
index 0000000..e69de29
diff --git a/file2 b/file2
new file mode 100644
index 0000000..e69de29
```

git reset - - soft <commit_id>

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git reset --soft 52b2db3
```

git status

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file1
        new file:   file2
```

git log - - oneline

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
52b2db3 (HEAD -> main) first commit
1d46b17 user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

git reset head <filename>

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git reset head file1 file2
```

git status

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
        file2

nothing added to commit but untracked files present (use "git add" to track)
```

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

git log - -oneline

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
52b2db3 (HEAD -> main) first commit
1d46b17 user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

git show <commit_id>

git reset - -mixed <commit_id>

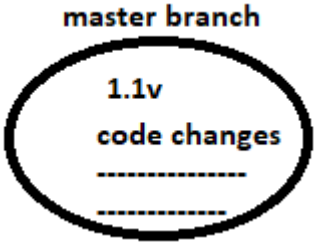
```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git reset --mixed 1d46b17
```

git log - -oneline

git status

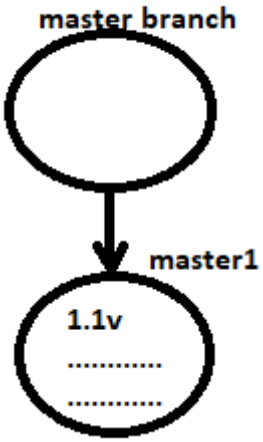
Topic 3 : How to create Branches

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch
* main
```

git branch master-1 → to create new branch

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch master-1
```

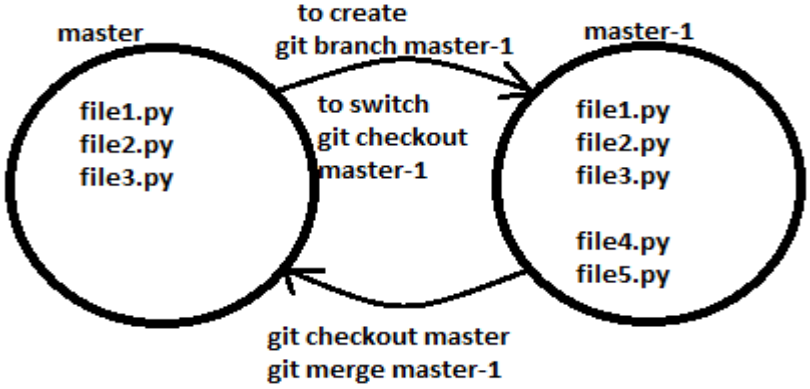
git branch or git branch - -list

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch
* main
  master-1
```

git checkout <branch name>→ to switch the branch

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git checkout master-1
Switched to branch 'master-1'

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ git branch
  main
* master-1
```

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



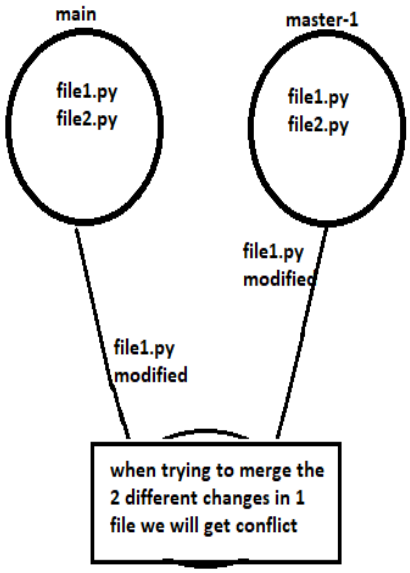
How to delete a Branch

- 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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch
* main
  master-1

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch -d master-1
Deleted branch master-1 (was 1d46b17).

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch
* main
```

Topic 4: conflicts

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ touch file1.py file2.py

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git add .

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git commit -m "added file1 and file2"
[main 7ab787d] added file1 and file2
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.py
 create mode 100644 file2.py

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 247 bytes | 247.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/pavanpuppala0346/10.Tools.git
   b5f6421..7ab787d  main -> main

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git branch master-1
```



Modified file1.py in main branch

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ cat file1.py
hi hello pawan i am from file1 in main branch
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git checkout master-1
Switched to branch 'master-1'
M       file1.py

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ vi file1.py

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ cat file1.py
hi hello pawan i am from file1 in main branch
modified in master branch

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ git add .

warning: LF will be replaced by CRLF in file1.py.
The file will have its original line endings in you

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$

LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ git commit -m "changes in master branch"
[master-1 2725d22] changes in master branch
 2 files changed, 2 insertions(+)
 create mode 100644 .file1.py.swx
```

```
LENOVO@pavan MINGW64 /e/code/10.Tools (master-1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ vi file1.py

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ cat file1.py
hello modified from main branch

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git add .
warning: LF will be replaced by CRLF in file1.py.
The file will have its original line endings in y

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git commit -m "changed in main branch"
[main 57c53f0] changed in main branch
 2 files changed, 1 insertion(+)
 create mode 100644 4913
```

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git merge master-1
Auto-merging file1.py
CONFLICT (content): Merge conflict in file1.py
Automatic merge failed; fix conflicts and then commit the result.

LENOVO@pavan MINGW64 /e/code/10.Tools (main|MERGING)
```

git merge –abort → To come out from the conflicts

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main|MERGING)
$ git merge --abort

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
```

Solution:

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git merge master-1
Auto-merging file1.py
CONFLICT (content): Merge conflict in file1.py
Automatic merge failed; fix conflicts and then commit the result.

LENOVO@pavan MINGW64 /e/code/10.Tools (main|MERGING)
$ git add .

LENOVO@pavan MINGW64 /e/code/10.Tools (main|MERGING)
$ git commit -m "two branch changes"
[main 2d70bb4] two branch changes

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ cat file1.py
<<<<<<< HEAD
hello modified from main branch
=======
hi hello pawan i am from file1 in main branch
modified in master branch
>>>>>>> master-1
```

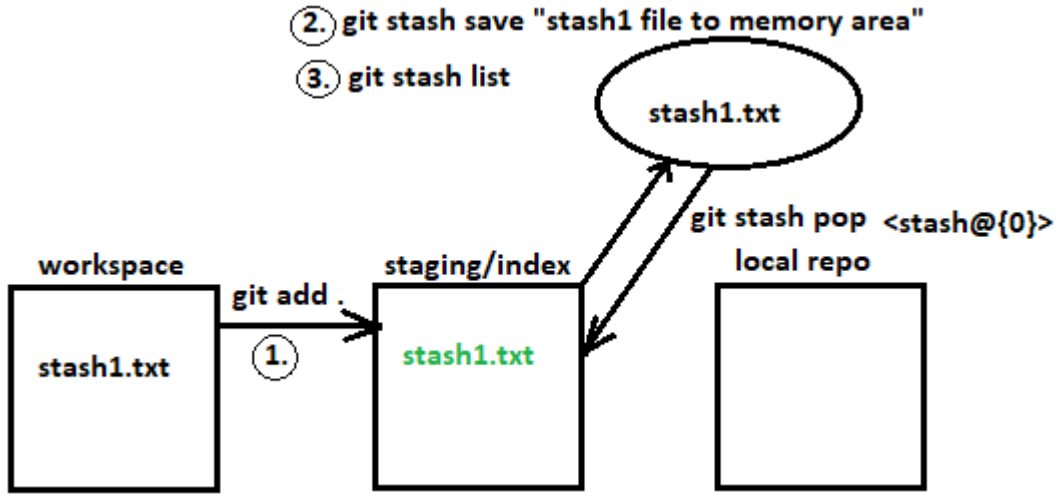
Note:

- 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

vi d7
esc i

---
esc :w
esc :q
or
esc:wq

Stash: it is temporary memory area provided by git




```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ touch stash1.txt

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git add .

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   stash1.txt


LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git stash save "stash1 file to memory area"
Saved working directory and index state On main: stash1 file to memory area

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git stash list
stash@{0}: On main: stash1 file to memory area
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git stash show -p stash@{0}
diff --git a/stash1.txt b/stash1.txt
new file mode 100644
index 0000000..e69de29
```

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

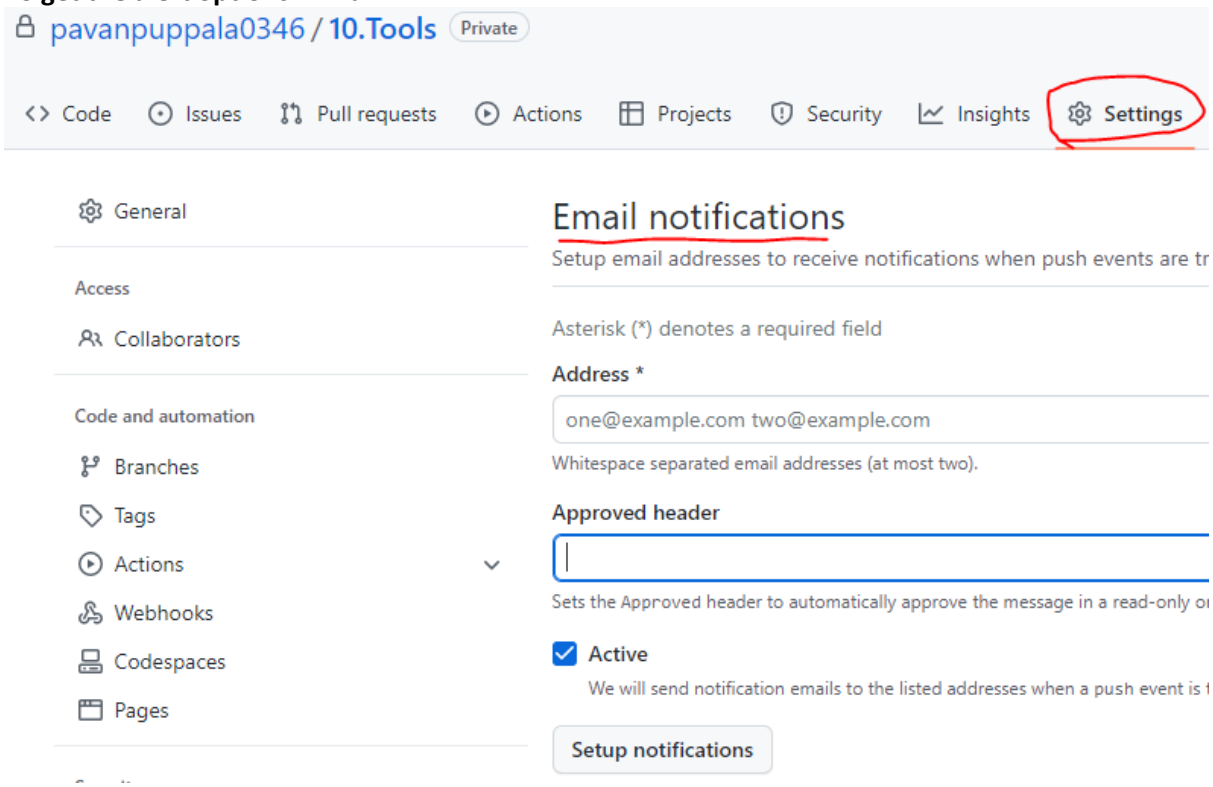
git config - -global alias.l1 "log - -oneline" → created alias for "git log - -online"
git l1

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
2d70bb4 (HEAD -> main) two branch changes
57c53f0 changed in main branch
2725d22 (master-1) changes in master branch
7ab787d (origin/main, origin/HEAD) added file1 and file2
b5f6421 data delete
162f353 Merge branch 'main' of https://github.com/pavanpuppala0346/10.Tools into main
a01341c delete
b4f4f36 new files created in main branch
```

git tag v-1.1

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git tag v-1.1

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
2d70bb4 (HEAD -> main, tag: v-1.1) two branch changes
57c53f0 changed in main branch
2725d22 (master-1) changes in master branch
7ab787d (origin/main, origin/HEAD) added file1 and file2
b5f6421 data delete
162f353 Merge branch 'main' of https://github.com/pavanpuppala0346/10.Tools into main
a01341c delete
b4f4f36 new files created in main branch
```

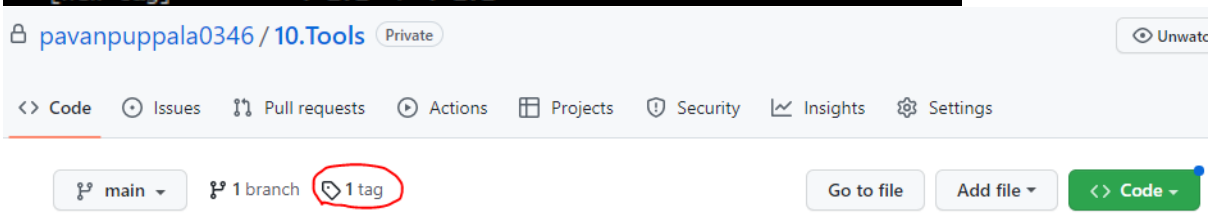
git tag (or) git show v-1.1

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git tag
v-1.1
```

Push the code to central repo

Git push origin v-1.1

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git push origin v-1.1
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 907 bytes | 36.00 KiB/s, done.
Total 9 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/pavanpuppala0346/10.Tools.git
 * [new tag]         v-1.1 -> v-1.1
```

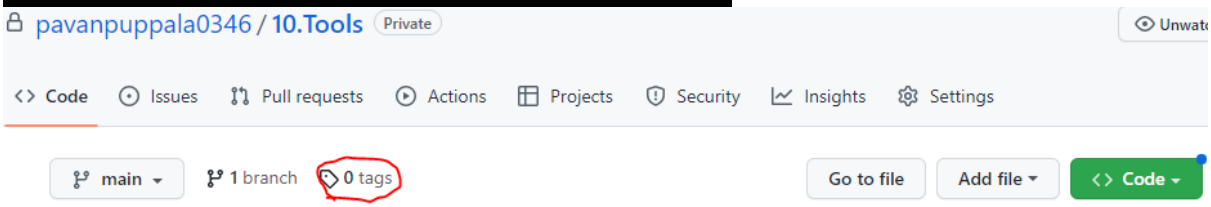


git tag –d v-1.1 → to delete the tag in local repo

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git tag -d v-1.1
Deleted tag 'v-1.1' (was 2d70bb4)
```

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

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git push origin -d v-1.1
To https://github.com/pavanpuppala0346/10.Tools.git
 - [deleted]         v-1.1
```



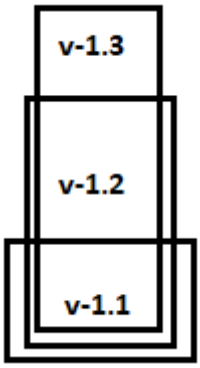
create a tag for specific commits

git log - -oneline

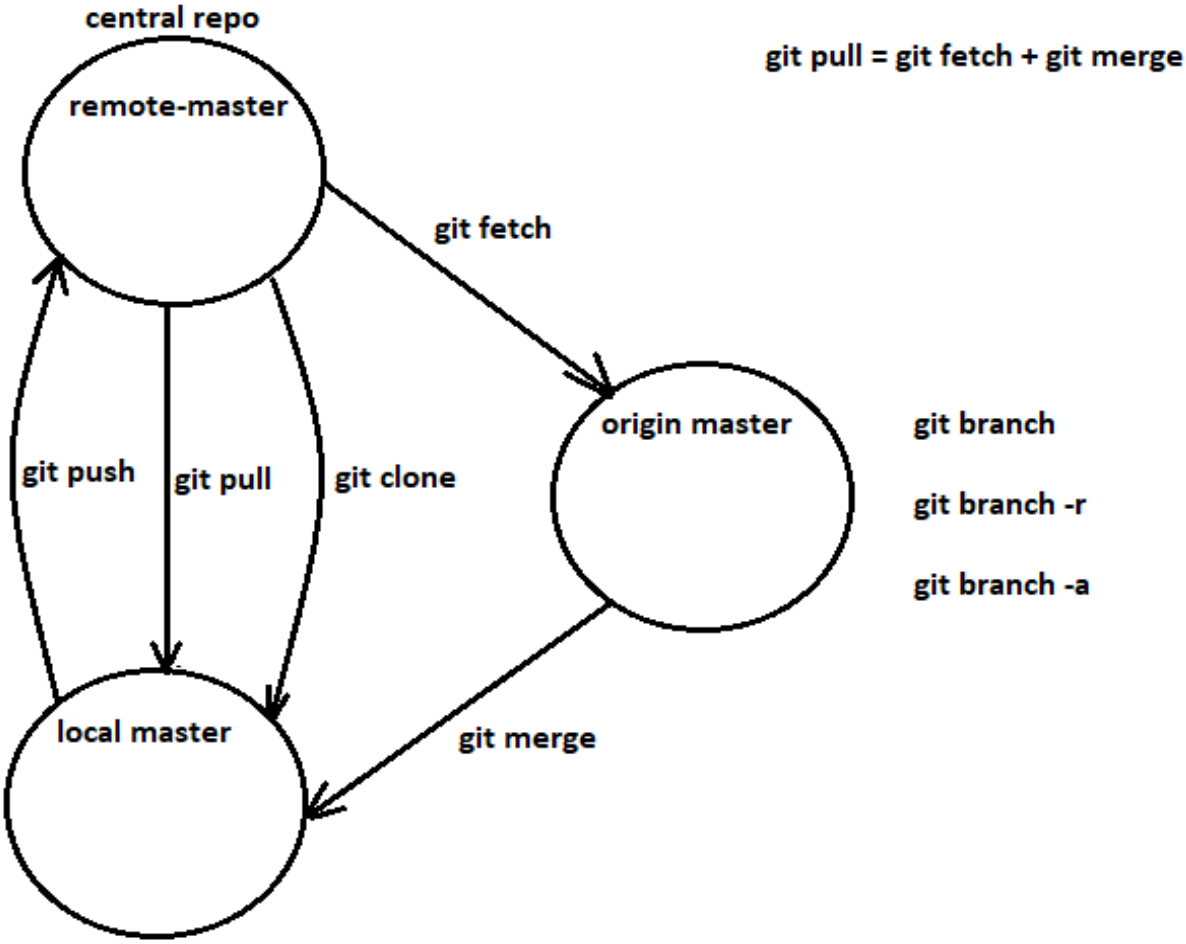
git tag <tag id> <commit id>

```
LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git tag v-1.1 1d46b17

LENOVO@pavan MINGW64 /e/code/10.Tools (main)
$ git log --oneline
2d70bb4 (HEAD -> main) two branch changes
57c53f0 changed in main branch
2725d22 (master-1) changes in master branch
7ab787d (origin/main, origin/HEAD) added file1 and file2
b5f6421 data delete
162f353 Merge branch 'main' of https://github.com/pavanpuppala0346/10.Tools into main
a01341c delete
b4f4f36 new files created in main branch
5510b13 commit multiple files
52b2db3 first commit
1d46b17 (tag: v-1.1) user
ef2a544 system linux
4759286 git
ab893ad Initial commit
```

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>