Git & Git-Hub

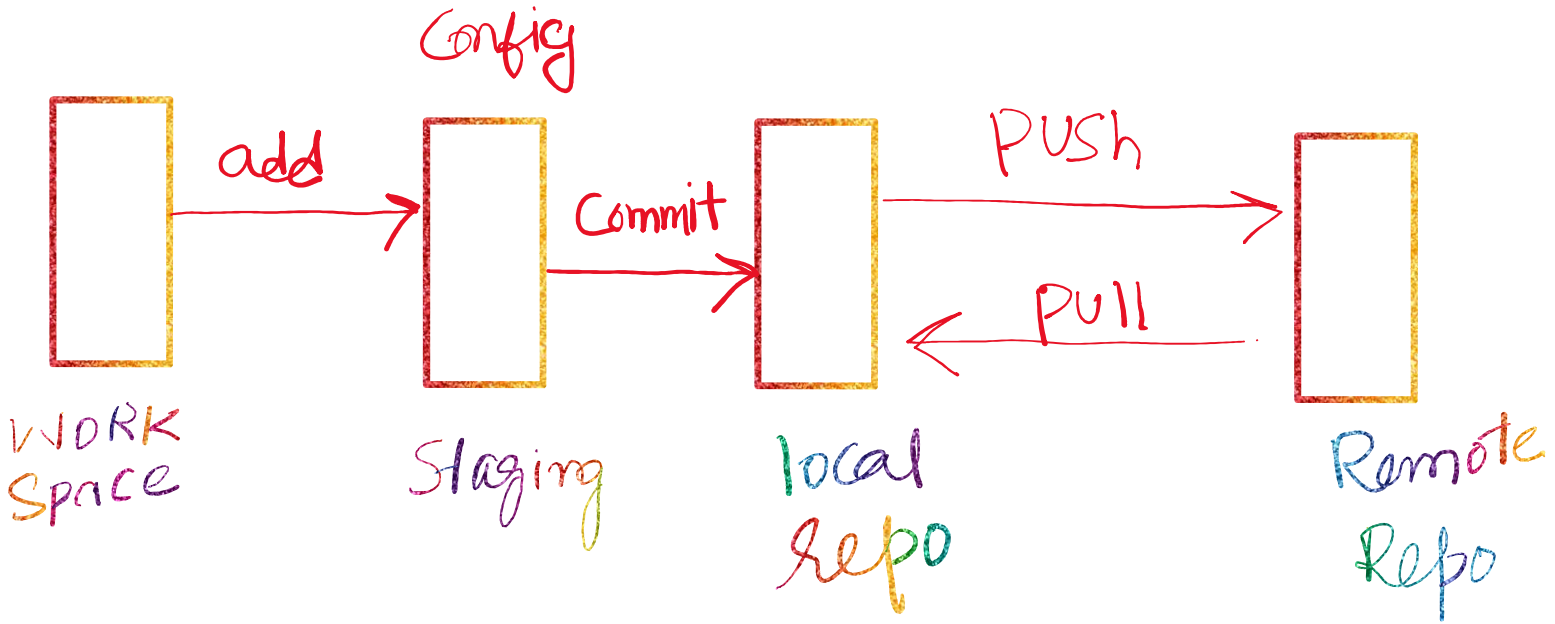
Monday, March 27, 2023 9:17 AM

Git bash:

- Git --version
- Mkdir foldername
- Touch filename
- Echo text inside file >>filename
- Cat filename ---> prints what is inside a file
- Ls --->lists what is contained in a folder
- Ls -a --> lists all files
- Clear/ ctrl+l ---> clear bash screen

>> --> append

>--> right over



Git Commands :-

---

1. Git init ---> initiating Git in a folder
2. Git status
3. Git add ---> adding from work space to staging area
   a. Git add . ----> add all
   b. Git add .* ---> add files having same extension
   c. Git add -a ---> add all
   d. Git add file_name1 [file_name2] [file_name3]

Before making commit we need to config once:

Git config [--global/system] user.name "Username"
Git config [--global/system] user.email "email"
Git config [--global/system] color.ui auto

4. Git commit ---> staging to local repo
   a. Git commit -m "Type a message"
5. Git log ----> details about commits tasks performed--> author date and commit message.

Modifying File:-

If file is already tracked(git add .) and now we have modified the file we don't need to stage it once again.
Use combined code: git commit -a -m "type message"

Git diff:

Comparing changes between different stages of version control

Git diff file_name --> compares files in work space and staging

Git diff file_name --> compares files in work space and staging

$ git diff earn.txt

diff --git a/earn.txt b/earn.txt
index eb71c68..ed1c103 100644
--- a/earn.txt
+++ b/earn.txt
@@ -1,2 +1,3 @@
 hello world {space indicates no changes}
 This line is appended and is to be staged
+This line is to be appended and is not to be staged

a/file_name ----> staging --- indicates something is yet to be staged to this file
b/file_name ------> workspace +++ indicates something has been added to this file

Eb71c68 -----> represents hash of staging
Ed1c103 -------> represents hash of work space

100644 ------> git file mode
100 ----> represents file type
644 ----> represents file permissions --> rw-r-r

4 ---> represents read permission - r
2 ---> write - w
1 ---> execute - e

compares files in work space and Last Commit(local repo)

Git diff head file_name

compares files in staged and Last Commit(local repo)

Git diff --staged head file_name

compares files in work space and Specific Commit(local repo)

Git log --oneline ---->…gives commit_ids of commits performed

Git diff commit_id file-name

compares files in staged and Specific Commit(local repo)

Git diff --staged commit_id

compares files in two Specific Commits(local repo)

Git diff comit_id   commit_id file_name

compares files in two branches

Git diff master_branch_name test_branch_name

compares files in local_repo and remote_repo

Git diff master_branch_name origin/branch_name

Removing Files:

Git ls-files -----> accessing files inside staging area

Use of git rm

1. Removing files from staging area and work space:
   Git rm file_name -----> removing specific file

   Git rm -r . --------> removing all files

2. From staging area:
   Git rm --cached file_name

   Git rm --cached -r .

3. From work space:
   Rm file_name

   Rm -r .

Git checkout:
The checkout command is used to discard Unstaged changes in the tracked files of the working directory

Git checkout -- file_name

Git Reset commonds:

1. Undo changes done to staging area:
    a. Git reset file_name
2. Undo commit at local repo.:
    a. Git reset <mode> <commit_id>
    b. Mode:-
        i. Will decide changes applicable to staging / work space or both or none
            1) --mixed: changes also applied to staging area
            2) --soft: changes are not applied to any
            3) --hard: changes are applied to both

GIT - BRANCHING:-

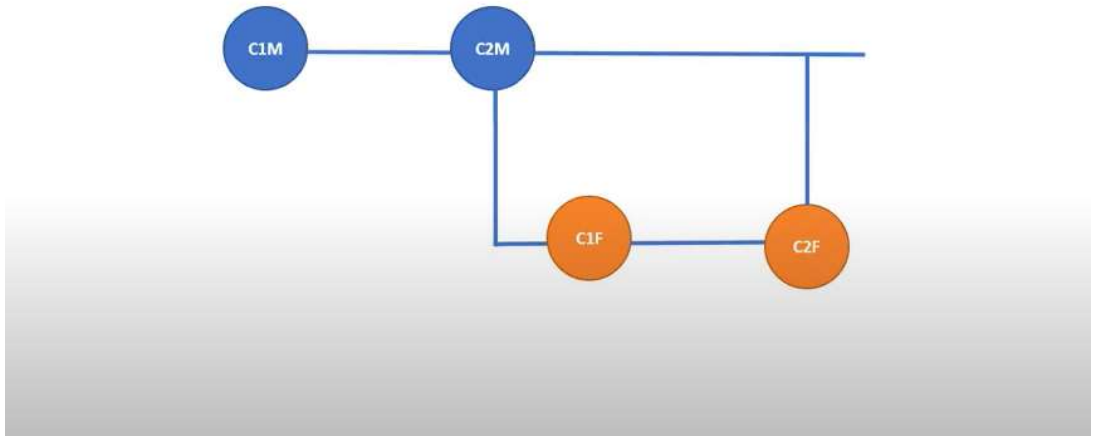
Git branch ----> view all branches available

Git branch branch_name -------> creating a new branch

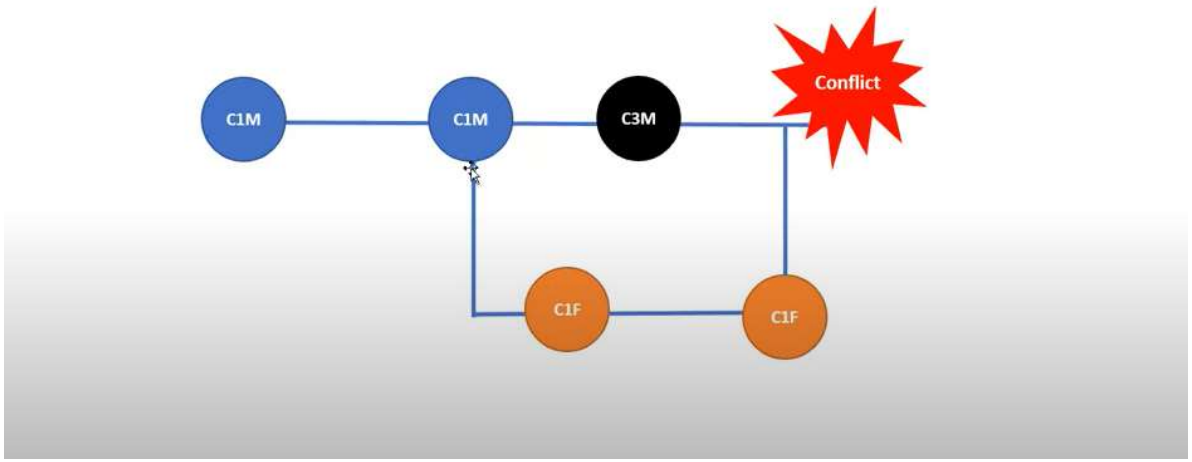
Git checkout branch_name -----> switching between branches

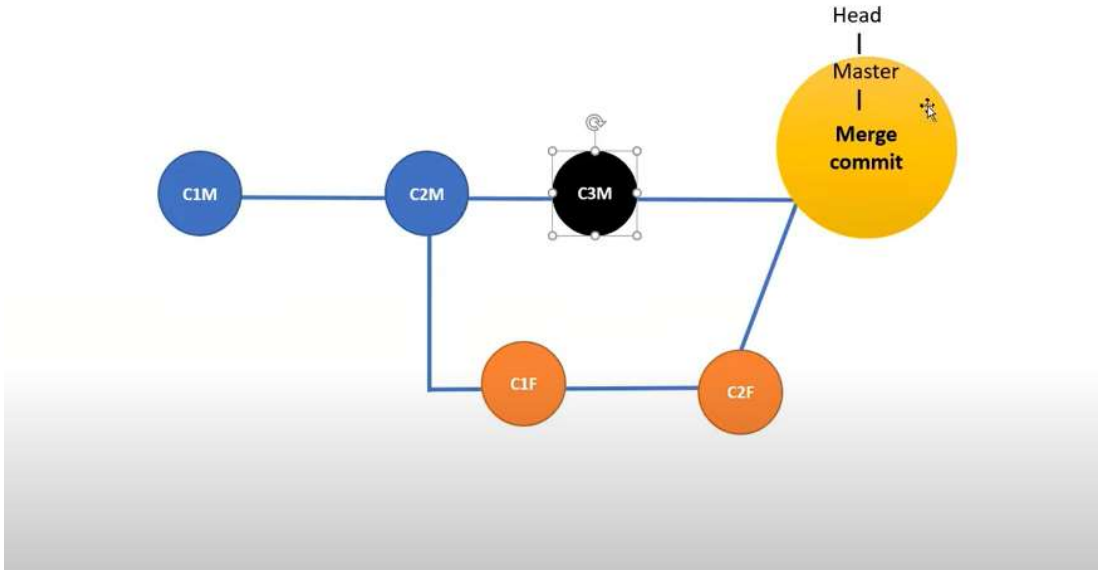
Git checkout -b branch_name -----> create and switch in single command

Fast-forward Merge



Three-way Merge



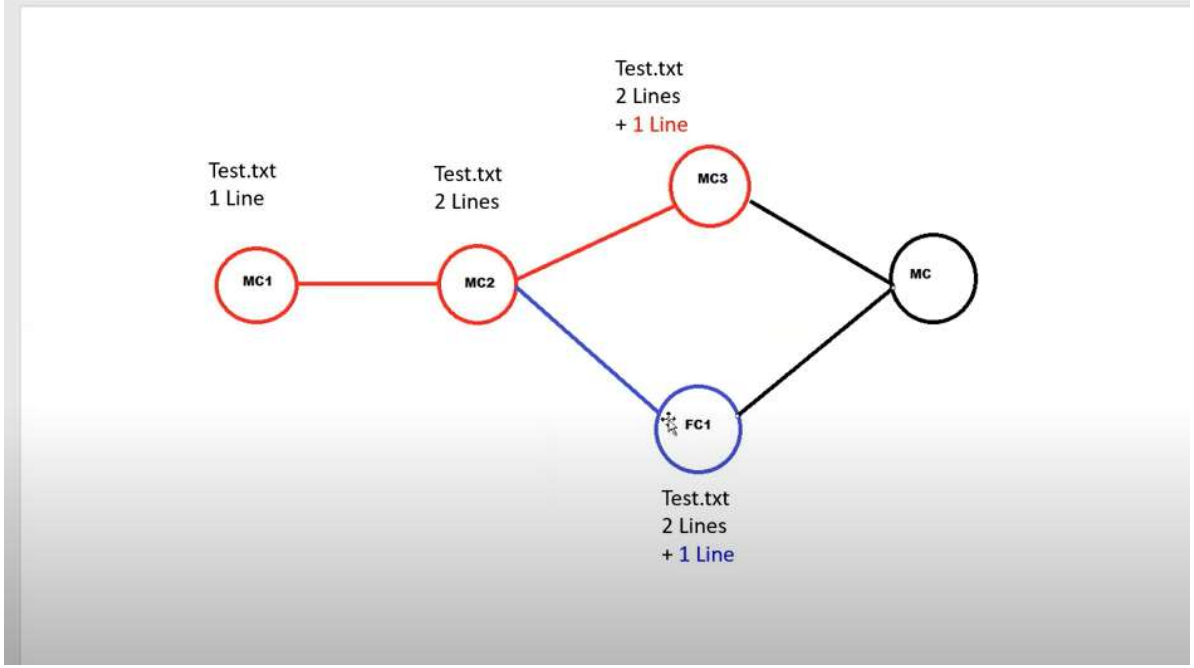


Merging:-

1. Switch to master branch
    a. Git checkout master
2. Git merge branch_name ----> merging specific branch to the master branch

*** same for three way(recursive strategy) or fast forward merge

In case of three way merge (recursive strategy) after merge a message pops up------> we use :wq! To save and exit the merge



Conflict arises when we change something in the main branch that is also the part of the branches.

Git merge branch_name -----> we will get a message about conflict

Cat file_name ----> view to see what is the conflict

Vi file_name or vim file_name ---------> open file

Make changes in the content (dd ----> for deleting lines which are not required)

:wq! -------> save and exit

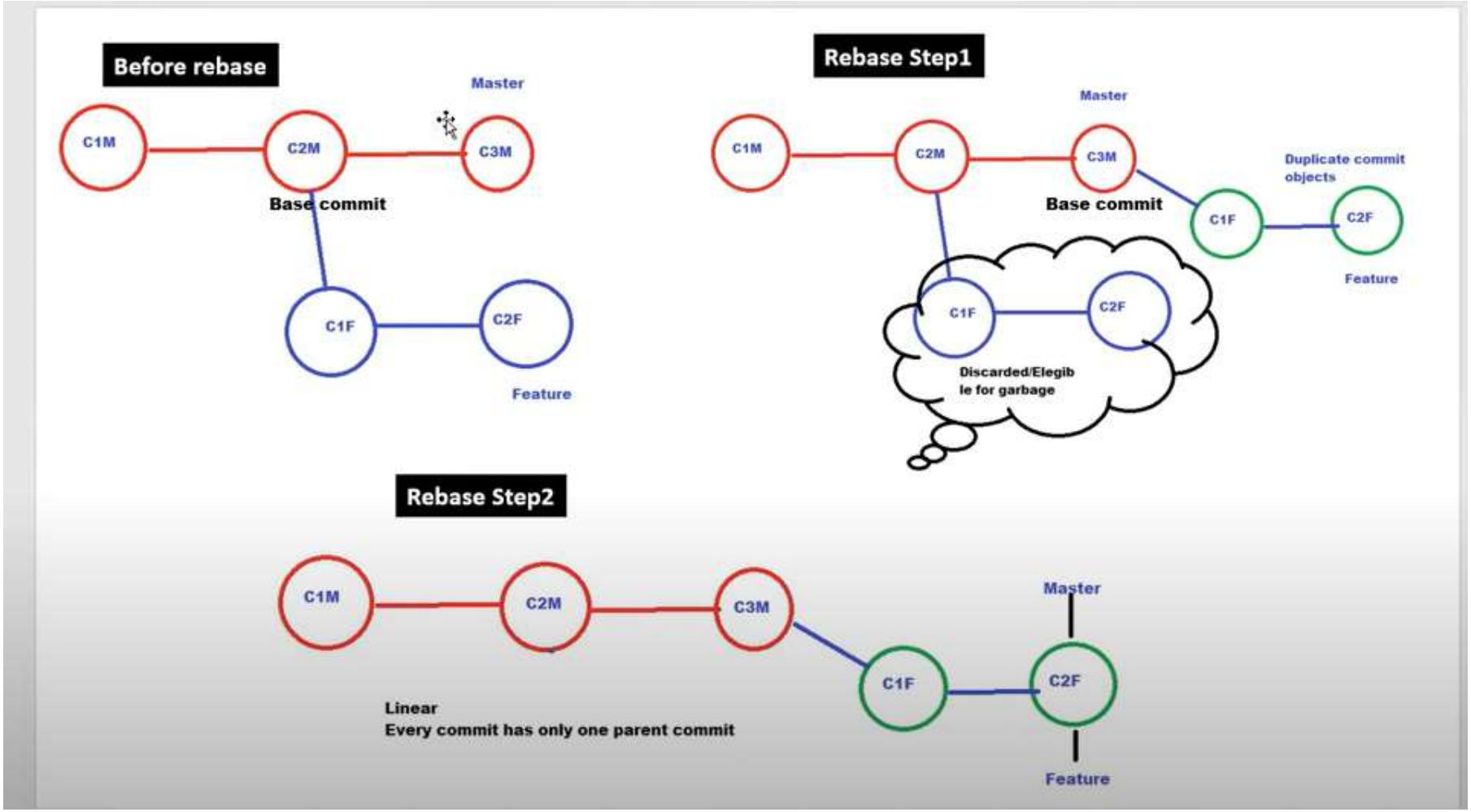
After resolving conflict:

Create a commit -------> git commit -a -m "type message"

Merge branch now ------> git merge branch_name

Graphical view of log ------> git log --oneline --graph

Deleting branch which has been merged -------> git branch -d branch_name



Screen clipping taken: 3/28/2023 10:38 AM

Rebase not recommended in remote repo
Rebase erases history

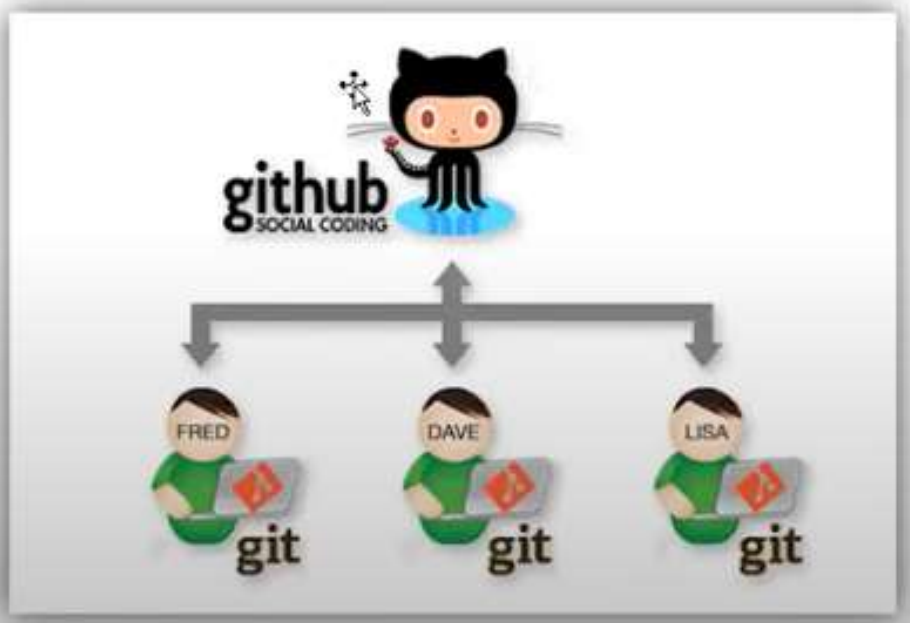
1. Switch to feature branch
2. Execute rebase in feature branch over master ------> git rebase master

a. This will create new commit objects with different commit ids

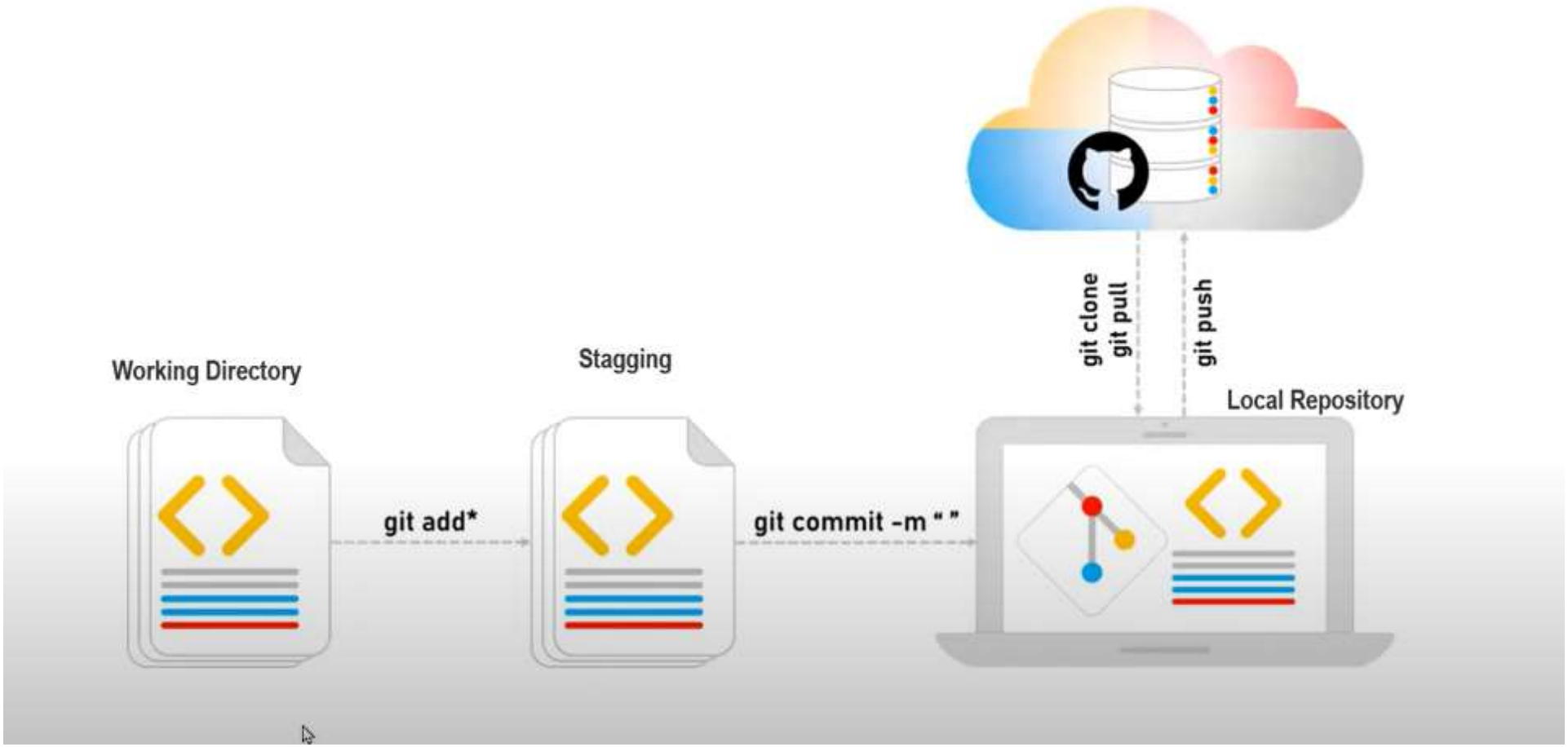
**** git log --oneline --branch_name ------> gives log of particular branch

Github

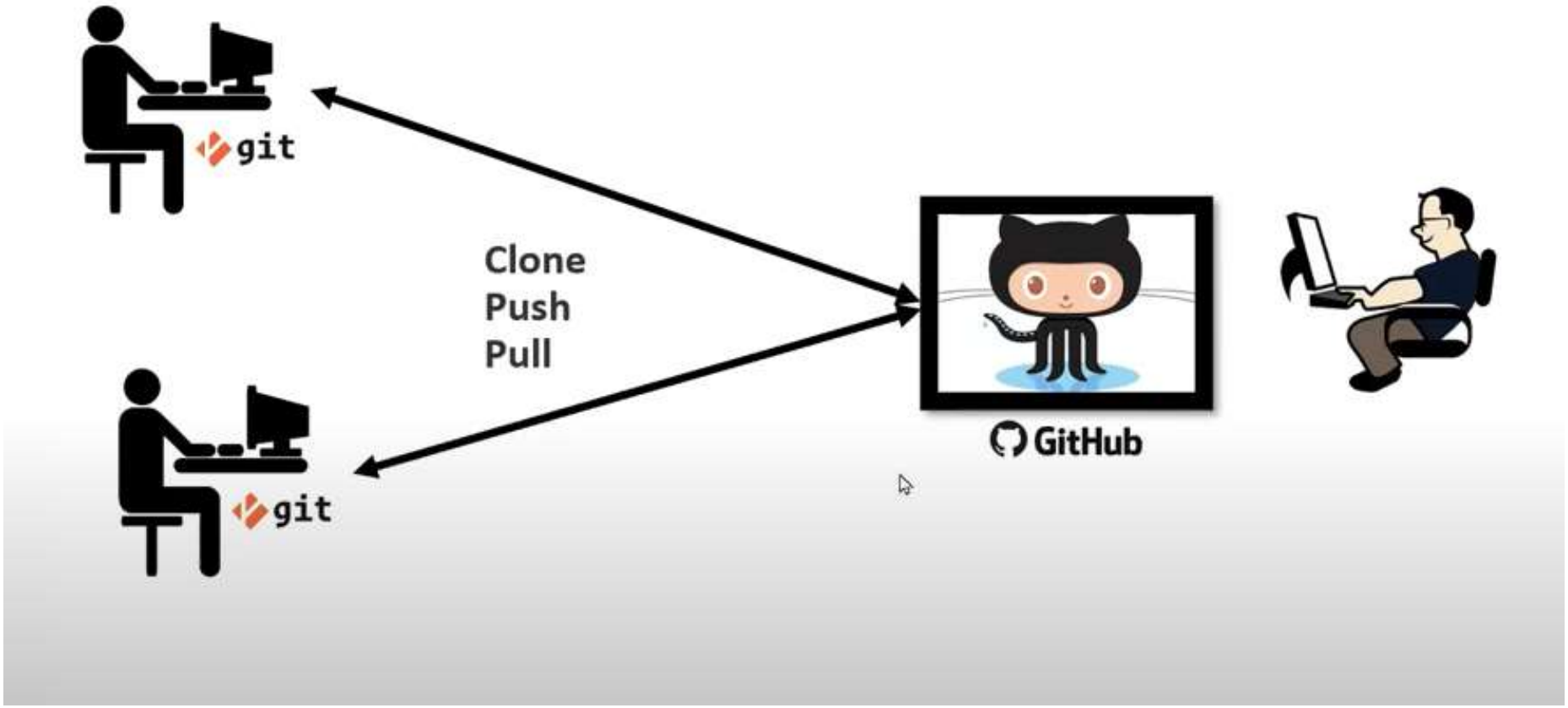
- GitHub is a hosting service for git repositories.
- Git is the tool, while GitHub is the service to use git.



Screen clipping taken: 3/28/2023 10:48 AM



Screen clipping taken: 3/28/2023 10:49 AM



Screen clipping taken: 3/28/2023 10:56 AM

Cloning to local repo -----> git clone paste_the_url
Go to code of github repo clone and copy

Pushing from local repo to remote repo:

1. Already created repo and that repo is in our local repo too:
    a. Git push origin branch_name_in_that_remote_repo

1. We only have local_repo:
    a. Changing name of branch to main where we want to push the local repo ------> git branch -M main
    b. create a remote origin where we need to push the local repo to -----> git remote add origin url_of_newly_created_repo_on_remote_repo
    c. Push to this remote repo ----> git push -u origin main