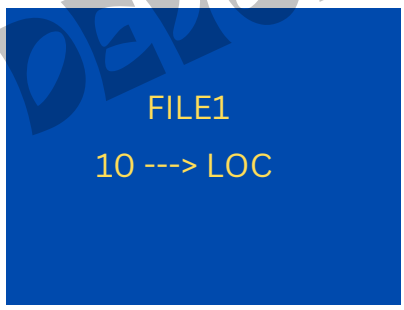
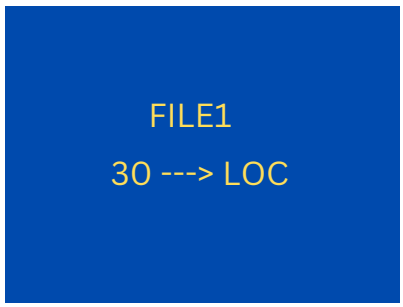
GIT: GLOBAL INFORMATION TRACKER

WHY GIT?

- IT IS USED FOR SOURCE CODE MANAGEMENT.
- GIT IS USED TO MAINTAIN MULTIPLE VERSIONS OF THE SAME FILE.



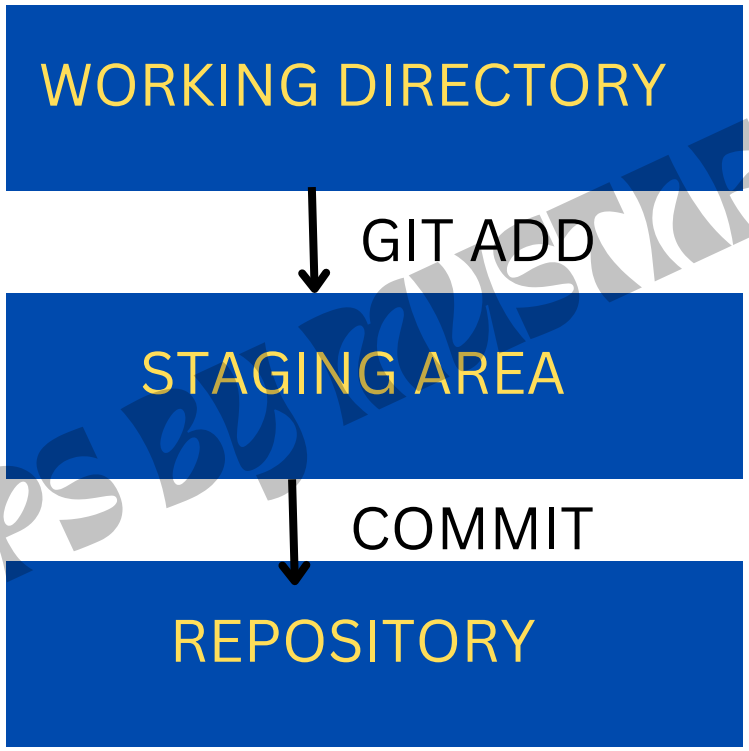


- GIT IS FREE AND OPEN SOURCE.
- GIT IS PLATFORM INDEPENDENT.

IT SUPPORTS LINUX, MAC OS, AND WINDOWS.

- GIT IS USED TO TRACK THE FILE.
- GIT WAS INTRODUCED IN THE YEAR OF 2005.

GIT STAGES:



WORKING DIRECTORY:

- In this stage git is only aware of having files in the project.
- It will not track these files until we commit those files.

STAGING AREA:

- The staging area is like a rough draft space, it's where you can git add the version of a file or multiple files that you want to save in your next commit.
- In other words, in the next version of your project.

REPOSITORY:

- Repository in Git is considered as your project folder.
- A repository has all the project-related data.
- It contains the collection of the files and also history of changes made to those files.

We have 3 repositories in GIT

- LOCAL REPO
- REMOTE REPO
- CENTRAL REPO

DEVOPS BY MUSTAFA

LOCAL REPO:

The Local Repository is everything in your .git directory. Mainly what you will see in your Local Repository are all of your checkpoints or commits. It is the area that saves everything (so don't delete it).

REMOTE REPO:

The remote repository is a Git repository that is stored on some remote computer.

CENTRAL REPO:

This will be present in our GITHUB

INSTALL GIT:

yum install git -y

yum: yellowdog updater modifier

To check the git version:

git --version

git init . : to get empty repo

To track the file:

git add file_name : single file

git add aws azure gcp: multiple files

git add * : all regular files

git add . : including hidden files

GIT INSTALL: yum install git -y
TO INITILIZE AN EMPTY REPO: git init .

here .(dot) represents the current directory (~). so it will creates .git folder (hidden)

to check the version: git --version
it returns: git version 2.38.1

GIT is used to track the file
to track a file

1. touch file_name : create a new file
2. git add file_name : add git to file
3. to check : git status

green color file represents tracking files
red color files represents untracking file

tracking means: we have a record that what we made the changes on a file.
it will track the each and every change in a file.

to commit a file: git commit -m "message" file_name

once we commit the file, it will goes to repository.
we can see the history of a file by using git log

DEVOPS BY MUSTAFA

to track multiple file: git add f1 f2 f3
to track all files: git add *
to track all files including hidden files: git add .

to commit multiple files: git commit -m "msg" f1 f2
to commit all tracking files: git commit -m "msg" .

.(dot) represents the all tracking files
notes: we can't commit the untracking files.

to untack a file: git rm --cached file_name
once we untrack a file, that files goes to working directory.

DEVOPS BY MUSTAFA

to untrack the files:

git rm --cached file_name

to commit file:

git commit -m "message" file_name

git commit -m "message" aws azure gcp

git commit -m "message" .

to check the history

git log

git log:

git log -2: latest 2 commit

git log --oneline: used to see only commit id's and commit messages

git config: it is used to configure the user with their mail id in git

command: **git config user.name "user_name"**

git config user.email "user@email.com"

git restore: used to get back the deleted files

command: **git restore file-name**

git show: it is used to show the file names in log

command: git show commit_id --name-only

GIT BRANCH:

- A branch represents an independent line of development.
- A branch is a way to isolate development work on a particular aspect of a project.
- The git branch command lets you create, list, rename, and delete branches.
- The default branch name in Git is master.

to see the current branch: git branch
to create a branch: git branch branch_name
to go to a branch: git checkout branch
to delete a branch: git branch -d branch_name
to rename a branch: git branch -m old new
to create and switch at a time: git checkout -b branch_name

GIT MERGE:

It allows us to get the code from one branch to another. This is useful when developers work on the same code and want to integrate their changes before pushing them up in a branch.

command: git merge branch_name

GIT MERGE CONFLICTS

Merge conflicts happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge.

GIT makes merging super easy!
CONFLICTS generally araise when two people two people have changed the same lines in a file (or) if one developer deleted a file while another developer is working on the same file!
In this situation git cannot determine what is correct!

Lets understand in a simple way!

cat>file1 : hai all
add & commit
git checkout -b branch1
cat>file1 : 1234
add & commit
git checkout master
cat>>file1 : abcd
add & commit

git merge branch1 : remove it

DEVOPS BY MUSTAFA

identify merge conflicts:

see the file in master branch then you will see both the data in a single file including branch names

resolve:

open file in VIM EDITOR and delete all the conflict dividers and save it! add git to that file and commit it with the command (git commit -m "merged and resolved the conflict issue in abc.txt")

merge:

if you have 5 commits in master branch and only 1 commit in devops branch, to get all the commits from master branch to devops branch we can use merge in git. (command: git merge branch_name)

cherry-pick:

if you have 5 commits in master branch and only 1 commit in devops branch, to get specific commit from master branch to devops branch we can use cherry pick in git. (git cherry-pick commit_id).

DEVOPS BY MUSTAFA

MERGE VS REBASE:

When there are changes on the master branch that you want to incorporate into your java, you can either merge the changes in or rebase your branch from a different point.

merge takes the changes from one branch and merges them into another branch in one merge commit.
rebase adjusts the point at which a branch actually branched off (i.e. moves the branch to a new starting point from the base branch)

to merge: git merge branch name
to rebase: git rebase branch_name

GIT STASH:

Using the git stash command, developers can temporarily save changes made in the working directory. It allows them to quickly switch contexts when they are not quite ready to commit changes. And it allows them to more easily switch between branches.
Generally, the stash's meaning is "store something safely in a hidden place."

- git stash
- git stash apply
- git stash list
- git stash pop
- git stash clear

SOME EXTRA COMMITS:

git show <commit> --stat : you'll see the commit summary along with the files that changed and details on how they changed.
git commit --amend -m "New commit message" : to edit the commit message
git commit --amend --no-edit : used to add some files in previous commit. (--no-edit means that the commit message does not change.)
git update-ref -d HEAD : used to delete 1st commit in the git
git reset commit: used to delete all the commits (upto the commit id)
git commit --amend --author "Author Name <Author Email>" : used to change the author of latest commit

DEVOPS BY MUSTAFA