### Types of Cloud

There are 3 Types of Cloud

1. Public Cloud ===> Is open for all, anyone can create a account and they can use services provided by these Cloud Providers. Mainly

are AWS, AZURE, GCP, etc...

These cloud providers charges on-demand and on-usage bases, How much we use their services they will charge on that bases.

2. Private Cloud ===> a private cloud is a service that is completely controlled by a single organization and not shared with others

3. Hybrid ===> Hybrid cloud is mixup Public+Private Clouds, for Example :

### Service Models

1. IAAS (Infrastructure As A Service) ===> Like We are taking Infrastructure from the cloud platforms Eg: AWS, AZURE, GCP like that

EC2, VPC, Security group

2. PAAS (Platform As A Service) ===> EBS, SNS, Elastic BeanStalk

3. SAAS (Software As A Service) ===> RDS, S3, Lambda -- using Lambda quickly writing code and execute it any programming language

and it is moving to complete SERVERLESS

### What we need to Deploy an Application

1st we Need an Environment: Elastic BeanStalk

2. Security of the Application

3. Load Balanceing :

4. Auto Scaling : of our Application when the load is increases

5. Health check and Monitoring : it was going to check the Health of Application and Monitoring them, CLOUD WATCH

---- So all these are our Responsibilities as well, only when all these done

### What is EC2

Elastic Compute Cloud EC2 is AWS Service, that gives us Virtual Machines to Run our Applications on them..

AWS called this Instances insted of Virtual Machines

EC2 Uses AMI (Amazon Machine Image)

- AMI is a template that contains a software configuration (for example, an operating system, an application server, and applications)

- These AMI's will have all the Required software’s, like OS and its Dependencies

- For Eg: if we want OS with java, we can find that OS with java already installed AMI

- You can Launch Instance with AMI

### What is SSH

SSH (Secure Shell) is a Protocol that Enables a Secured Communication between Any two Machines in a Network. on Port 22

- If you want to communicate/Access to EC2 instance from your Laptop/Desktop you need to use the SSH Protocol

### What is GIT

- Git is a Distributed Version Control System, for tracking changes in source code

- it is free and open source.

- It is fast very fast and lightweight fast because we have the entire project in our local system

- we don’t need to connect to the remote repository every time

- which collaborates on a project

- And it also maintains the history of the work for the project progresses

#### What is github?

GitHub === github is most widely used remote hosting service for a Git repo and this includes a number of features as well. , which makes it very simple to setup source code repositories and collaborate with developers who use Git..

### What is version control is?

- Version control is a system that records changes to a file..

### Features of Git version control system?

- It is a distributed version control system

- It is Free and Open source

- Branching & Merging (creating branches of your repo, then merging those branches into your main source)

### what is remote repository?

- Remote repository which is online and contains the original source code and version history..

### What is a repository?

- Repository --- is a container of your source code,

### Use case of version control system

- to manage the different versions of the files over a period of time, we need to use version control system

- if you want to go back to the older file, we have both v1 and v2, every version of a particular file is stored

### Branching & Merging in Git?

Collaboration in Git --- Git makes it easy for different teams to collaboratively, work on the same source code

### pull request?

- A request made by a collaborator for their changes to be added to a branch in your repo

- pull request can be approved by the “reviewer” before a merge is performed

### Alternatives to GitHub?

- Atlassian Bitbucket, offers many of the same services and has a few unique ones of its own.

- There is also a GitLab, which includes a whole host of CI and CD features and also monitoring tools..

Note: Each of the cloud platforms also have their own versions of git..

- AWS Code Commit

- GCP GCP Cloud Source repos

git config --list ---- It shows default configurations of git

Mostly use alias command for shortcuts

# git config --global user.name " " = This command is used to configure the user name & email

# git config –global user.email “ ”

# git config user.name ----- It shows user name

# git config user.email ----- It shows user email

# man git ---- It gives detailed info of git commands

# git help config ---- If u want more detail about specific git commands

# git help branch ----

# mkdir company\_web

# cd compant\_web

# git init === This command is used to convert the folder/dir into a empty git repo

# ls -a ---- This command shows all hidden files in Dir

# cd .git ---- get into the .git Dir

# ls ---- this lists all files and Dir, which are used by the Local repo

# cd .. ---

# vi info.txt ----- creating a file

We are creating a webpage for compant…!

:wq

# ls

# we are going to print the text “Company Webpage” into a README.md file..

# echo “Company Webpage” >> README.md

# ls ---- It list two files here

# git status == it shows the current running status of the branch, tracked and untracked files

# git add . === if you want to add files to the staging area we use " git add . "

# git add <filename> == if you want add a particular file to the staging area " git add <filename> "

# git commit -m “ first commit” ---- it save file in our local repository.. with commit id, and we can

Get summary of the commit..

# git commit -a -m “ adding about new pages” --- Here we are using -a Flag means all of the commits in staging area.. to be commit

# git log === this will give us details history of all the commits you make in your local repo

- (HEAD points to latest commit always) it commits with new commit id

- It gives the date and time information of that commit and also the commit message you have used

- It will gives Author information Name & email

# git log --online == it will show you only the latest commit

# git log -2 ---- it will display 2 commits

# git log –stat --- You can see some additional info about, what exactly changed in each of the commits

# git log –patch --- You can see all of the recorded changes, using this command

# .gitignore --- what type of files you want to ignore in the project like (.txt, .py, .scm)

# git show <commitid> --- we can view the single commit,

# git remote -v === It fetch the remote names

# git branch -M main ---- This command is used to Re-name the branch names

# git remote add origin <https://github.com/KHAJASYEED/accent-ks-repo.git> --- The link has been created between local repo and the remote repo

# git remote -v ----- it fetch the remote names

# git branch ---- it shows the current branch, which you are working

# git branch -a ---- it shows all branches in repository

Changing git remote URL….?

# git remote set-url origin <URL Path>

# git branch -c devops ---- This command is used to create a new branch

# git checkout -b new\_branch --- it create new branch, and also checkout to that branch

Delete a Remote branch…..? to delete a remote branch in git

# git push <remote name> -d <branch name>

OR

# git push <remote name> :<branch name>

The pull with –rebase flag command, combines a fetch and a rebase instead of merge

# git pull –rebase <remote name> <branch name>

Now adding ssh-key to the git Hub to perform ssh Authentication

-- go to terminal 

# ssh-keygen ---- this command creates public-key, & private-key

# cd .ssh ---- Go to .ssh Dir

# ls --- this command lists the files & Dir

# cat id\_rsa.pub --- It display the content in the file

-- copy the display content…  go to github  go to settings  click on SSH and GPG keys  click new-ssh-key  give title <name>  and paste the copy content in the Key Box  click Add SSH Key

--- On Terminal side

First We test our connection to github.com

# ssh -T [git@github.com](mailto:git@github.com) ---- This command is used to Authenticate git to github..

Now can connect from our local machine to Github.com

#### Pushing Local repo to Remote repo

-- Go to the git repository

# git push origin main --- This command is used to push the code from Local repo to Remote repo..

--- For confirmation go to github.com and ckeck the files…

# git status

# git restore ---- this command is used to restore the file to its previous version…. When it was in workspace in red colour

Branching in Git?

- When you have multiple branches on your git repo, It can get hard to track all of them

- And also Visualize what state each of the branches are in..

- To avoid confusion, it is best to understand how exactly branching works. And one way to do this is

To visualize branches and commits are recorded on branches

-- There is an online tool which we can use.. <https://git-school.github.io/visualizing-git/>

This is called visualizing kit, and its available at this URL.

Remove files from git local-repo?

# git rm info.txt --- remove files from git local repo

# ls ---- Notes: this has disappeared from our project directory, that is from the working tree.

- Now, if you want a particular file to be deleted from the git repo, but not from the working tree?

- Bcoz we want to eliminate it from the repo temporarily but, we want to continue working on it because we may read it at a later point.

- For that we once again run git rm, but with a cached flag

# git rm –cached README.md ---

And the output confirms that, this has been removed But without a commit, it has not been removed from the repository, but it has been removed from what is considered our index.

- The index in the context of Git is a list of files which have been modified and are being tracked

So by removing the READEME.md file from the index, it means that the removal from the repo will take place at the next commit.

# ls ---- When we do ls, the file is still present in the working dir/workspace

- this is the effect of performing the git rm with a cached flag.

# git status ---- at this point, well at the top, we can see what is present in the git index.

- The two files README.md and info.txt are currently staged for removal.

- This is the effect of us performing git rm for each of these files.

-- However in the case of README.md file, the fact that is still present in the file system has led git to consider it an untracked file which has been added to the repo.

# ls --- confirm that three files are present in the file system, and image Dir

- The README.md is of course, no longer part of index.

# git mv web\_company.html blue\_company.html --- it is used to rename the file/ move the file from dir to an-other dir..

-- this is the point where we create a new branch

# git checkout -b new\_branch --- This command is used to create a new branch, and checkout into the branch and it clone entire project from that branch.

- in fact you see the output , The D next to README.md as well as info.txt coveys that, the files have been removed since the previous commit.

- And importantly, the renaming of a file has been recorded.

# git status --- from the output, we can see that README.md and info.txt will be removed from the repository the next time we perform a commit operation.

- But the renaming of the file does show up as a renamed operation in this output.

- The presence of the README file in our file system causes it to show up as a untracked file, so at this point, we are on the new\_branch which we just created.

So if you want to perform a git commit well, this commit will performed on the new\_branch not on the master

# git commit -m “Rename of company\_web.html and removal of info.txt and README.md”

And when we run this, that fact is conformed in the first line,

# git status --- it is only presence of the README file in our file system, which shows as a untracked file

- and there are no staged changes for us to commit.

# ls --- to confirm that README file is still around.

And now let us switch branches over to the master.

# git checkout main ----- it shows an error, This is because whenever we switch branches, any un-tracked files will be over written if there is one with the same name.

- And if there is no such file in the branch which we are checking out, well that file will be removed from the file system.

- Now this is a situation which you might encounter every now and then,

- Your work is not quite ready to be committed, but you have more urgent task, you need to perform a branch switch.

- So what exactly can you do?

- well, one thing is for you stash your changes. And to do that, you first need to make that all of the changes are at least being tracked.

- So for that you can perform a git add.

# git add . – this command is used to add files to the staging area

- We known that README.md file will now be staged as a new file, added to our repo.

-- And then in order to stash your changes away, so that you can get back to it later, run git stash command.

# git stash --- the changes which we made to our project directory on top of our latest commit have been stashed away..

- Now, let us ga ahead and take a look at the git status

# git status ---and it shows us, there is nothing to commit, working tree clean

- This is because our changes have been stashed away..

# git stash list ---- it displays all multiple stash entries

# git stash show --- it shows the difference between the contents of the stash and the commit which was recorded on the same branch.

- Now we can perform a checkout or a switch to the master branch.

# git checkout main --- it changes from one branch to an-other branch, and this operation is now sucess

# git stash --- This command is used to temporarily stashes changes you have made to your workspace, so you can work on something else, and then come back to it..

-- we then performed a stash of uncommitted changes. And the we switch to the main branch.

# ls -----

- which is why when we do ls, well, the files which we had previously deleted from the new branch, now it show up. This includes the info.txt and README and observer that the html page for the company page

Now appears as company\_web\_page, and not brocadero\_web\_page.

- so we are not truly working on the master branch of our repo.

- And let just say we need to make a change to this branch.

# git merge --- This command is used to merge from one branch to an-other branch.. if you want to merge new\_branch to main branch, then you switch to main branch, and perform “merge command here”.

- When we run this a pop up will appear in order for us to record a message for this merge..

- it always good practice to record something meaningful here..

- and this message is recorded in a MERGE\_MSG file in the .git Dir

# git merge new\_branch ----

# git status –

# git log –oneline ----- it shows the latest commit recorded on the branch

# git push orign main --- this command is used to push the changes/commits to the remote repo

We will also take a look at the pull request feature which is available, when merging branches on GitHub

-- Go to GitHub  click on pull request 

Adding collaborators to repo… adding teammates to the working project

-- go to GitHub  click on web-page-repo  click settings tab  click on collaborate  give password 

Add person  hanumabanavath  add to this repository 

Invitation is sent to the email address. so she will need to accept that

GitHub === github is most widely used remote hosting service for a Git repo and this includes a number of features as well..

Adding Tag to the commits..

# git tag --- By default it will all of the tags currently available in our repo

# git tag stable main --- name of the tag is Stable, this stable tag apply on the latest commit in the master branch (This is a light weight Tag) which gives little info

# git tag --- it shows the available tags in our repo

Annotated tag ---

# git tag -a v0.1 -m “0.1 release” <commitID>

-a === which means the tag is stored there will be other metadata such as the name & email of the person doing the tagging which is include.

V0.1 -- is name of the tag

-m -- message

Push the Tags to the GitHub

# git push –tags --- it only push the Tags to the remote repository.

Deleting the Tags -----

- We have to delete Tags on Both Sides in GitHub as well as in Terminal

-- go to GitHub  go to project Dir  click on tags tab  click on tag name  click on delete

In Terminal

-- go to terminal 

# git tag

# git tag -d “tag-name” --- remove the tag from our repo, and create a commit id

# git tag

Deleting a Tag on Remote repo ----

# git push origin :v0.1 ----

-- give more empty space after origin, that means nill

v0.1 --- is tag name which we want to delete on remote repo

What is a project Management feature, to create and track issues with your project using GitHub.

-- go to GitHub repo – click on issues tab  new issue  write the issues – it is at open state

If u want to assign a to some one - click on Assignees and assign the issues to other person

\*\* Milestones are essentially targets for your project..

- but in the context of GitHub, they can serve as a container of issues.

- we have already to issues in GitHub. Issues tab

- and in order to link them with Milestones, We can head over to the Milestone section/tab.

- create anew milestone  web-page-milestone  create a milestone

What is a milestone?

Milestone is a container of issues, you can only attach a single milestone to an issue, but you can

Have milestones with multiple issues

- This feature can allow to group, similar issues together into a single container

\*\*\* Using project Boards

\*\* change integration with Rebase and Merge ---

- There are 3 classical approaches which can be utilized in order to integrate change.

1. Merge 2. Rebase 3. Cherry-pick

Merge – is preferred whenever you want to maintain history of the changes.

- this command is used to merge one branch to an-other branch

When you call merge, the changes made in feature branch to the master, and it creates a new commit which is referred as merge commit. And it creates a copy and it can be used in order to track the history

Rebase – it is considered as a better approach to streamline a complex history, you will be able to change the commit history by interactive rebase.

- when you call rebase, where rebase is not maintaining any duplicates. It is adding the feature branch in the same master branch

Git cherry-pick -- it is a powerful command that enables arbitrary Git commits to be picked up by reference, and then append to the current working head

# git reset . === if you want to unstage all the files from stageing area to workspace/dir " git reset . "

# git reset <filename> == if you want to unstage a particular file from stageing to workspace/dir " git rest <filename> "

# git status == it shows the current running status of the branch, tracked and untracked files

# git remote add -u origin master/main < URL of Remote repository>

# git push –set-upstream orign master ====== Adding / creating an Upstream trackeing

OR

# git push -u orign master

Remove a Global identity

# git config –global –remove-section “user.name”

# git config –global –remove-section “user.email”

Fetch the Remote names

# git remote -v === It fetch the remote names

$ **git remote** -v

origin https:**//**github.com**/**myusername**/**repo.git **(**fetch**)** origin https:**//**github.com**/**myusername**/**repo.git **(**push**)** upstream ***# łhis ﾕine may or may noł be here***

If upstream is there already (it is on *some* Git versions) you need to set the URL (currently it's empty):

# git remote set-url upstream https://github.com/projectusername/repo.git

Before you generate an SSH key, you can check to see if you have any existing SSH keys.

# ls -al .ssh ===> it list the content of your .ssh directory

# ssh-keygen ==== create public and private key pair

# git diff == it shows the difference between two commit ids " git diff commitid1 commitid2 "

Git Hooks:

- Git hooks are scripts that run automatically every time a particular commit occurs in a Git repository

# git merge == The "merge" command is used to integrate changes from another branch.

# git branch -d <branch name> --- this command is used to delete the branch

### How to Undo or Revert the changes you made

Before stageing:

After stageing:

After commit:

1. Before stageing:

- Delete come lines in <file.txt> ----> vi file.txt ----> delete some lines here ---> :wq ---> git status

# git checkout file.txt

2. After stageing:

go to vi file.txt ---> remove some lines --> : wq -----> git status ---> git add . ( adding to stageing area)

# git reset HEAD file.txt ---> cat file.txt there is no changes in file.txt Next

# git checkout file.txt -----> here we see the # cat file.txt

### Git Branches

# git branch ---- it will shows you the current branch you are on

- by default git creates a branch calle master/main

# git branch -a ---- it shows the all branches in the repository

#### create branch

# git branch -b dev ----- it creates branch, and also checkout to that branch

# git branch dev ---- creates a new branch called dev

# git checkout <branch name> ---> to switch to another branch

### Git Merge : The "merge" command is used to integrate changes from one branch to another branch.

# git merge dev ---- Now we are in master branch execute command here, to merge dev branch into master branch

- git status ---- we are in master branch

- git log ----- it shows HEAD master, dev

#### Resolving merge conflicts:

go to the conflicts file, in that file we use >>>>> and ====== and <<<<< symbols, remove symbols, and here you decide

what you want to keep ----> :wq

after that git add . ----- adding to the stageing area

after that git commit -m " " ----- commit it to the local repo

after that git merge

### Delete branch

# git branch -d <branch name> --- this command is used to delete the branch

#### Adding a new Remote Repository in git

# git remote -v ------ this will shows all available git remote repository in our local machine

# git remote add origin <URL>

# git remote -v ----------------- now it will show the remote repos

#### Push the code from local to Remote

# git push -u origin master/main

#### git pull & git push

# git pull ----- it will get changes from remote repository to local repository and it will auto merge

# git fetch ---- it will get changes from remote repository to local repository without Merge

- git fetch is used to see the what changes are happened in the Remote repository

# git push ---- git push is used, to push the changes from local repository to remote repository

#### Merge conflicts on the same branch

- If the merge conflict can happen on the same branch as well

let say you are editing a file.txt ---> you have delete some lines here ---> :wq

git add . ----> git commit -m " " ----> before you go ahead and push your changes, and other developer might pull the code,

Make the changes and push it onto the central repository.

So go to the remote repo and make some changes in the file.txt -----> and commit it

and at this time you push the code what will happen

# git push ------ get some errors here rejected

what ever it is make # git pull ----> get changes from remote repo ---> and after that open vi file.txt ---> remove >>>> <<<< ====

symbols ----> and decide what you have to keep ----> :wq

# git add . ----> git commit -m " " -----> git push

#### Git fetch Vs Git pull

Git fetch == git fetch is used to see the what changes are happened in the Remote repository

git pull ----- it will get changes from remote repository to local repository and it will do auto merge

#### Git Reset Vs Git Revert

Git reset:

Let's say we have four different commits that we have made on our repository and the HEAD is currently.

But at some point we have discovered that we have issues with Commit 3 and 4 and we want to go back to commit..

to do that.

We lose if we use # git reset command here

- I want the latest two commits to be gone, because I have issues with them..

# git reset --hard < commit id > which commit you want to HEAD/latest ---> after that

(here we use --hard, bcoz the two commits are already pushed to the Remote repository)

# git push -f ( the normal push will not do it, so we use -f (forceable) ) it will do force update to remote repo

note:

- git reset will be dangerous, bcoz the latest commit will be gone permanently... we couldn’t recover

Git Revert:

- it will remove the changes from previous commit, and commit with new commit ID, in this process it doesn’t delete the data, and

it doesn’t delete the history of the commit ID... git revert is doing in public, it will see all in your remote repo

# Git revert < commit id > ----> git push ----> go and check in remote repo

#### Concept of cherry pick command

git cherry pick:

- Git's cherry-pick command is used to copy the changes introduced in a single commit onto a branch as a new commit

- This command is useful in situations where you know you need a specific change applied to some other

branch of your repository, and you also know that the changes made in commits prior to the commit you're cherry-picking are not needed.

#### Git Stash Command

- let open up the file.txt and do some changes here,

- and suddenly you got a urgent work to do on bug fixes, in your local repo / other branch

- At this time git stash command is very useful,

- So if you say git stash whatever work you have done here, will be moved to a temporary location for you

# git stash ----> saved working directory ---> go to another branch and do the work after that ----

# git stash pop ---> the file.txt will come back to you, and continuing working