Git [maintaing file with different timestamps]

30/1/23.

→ Version control :- Version control Software (VCS) is also referred as a Source code managment (Scm) or Revision control system.

→ It is the way to keep track of the changes in the code so that if something goes wrong, we can make comparison in different code versions and revert to any prevision version that we want.

Why version control :

→ Tracking different version.

→ Collaboration and sharing files.

→ Historical (history) information.

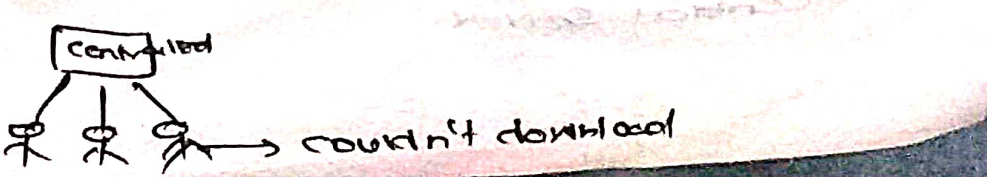
→ Retrieve past version.

→ manage branches.

→ Types of VCS :-

1. Centralized version control system [CVCS]:
   - → CVS
   - → SVN
   - → Perforce.
2. Distrubuted version control system [DVCS]
   - → Git
   - → Mercurial

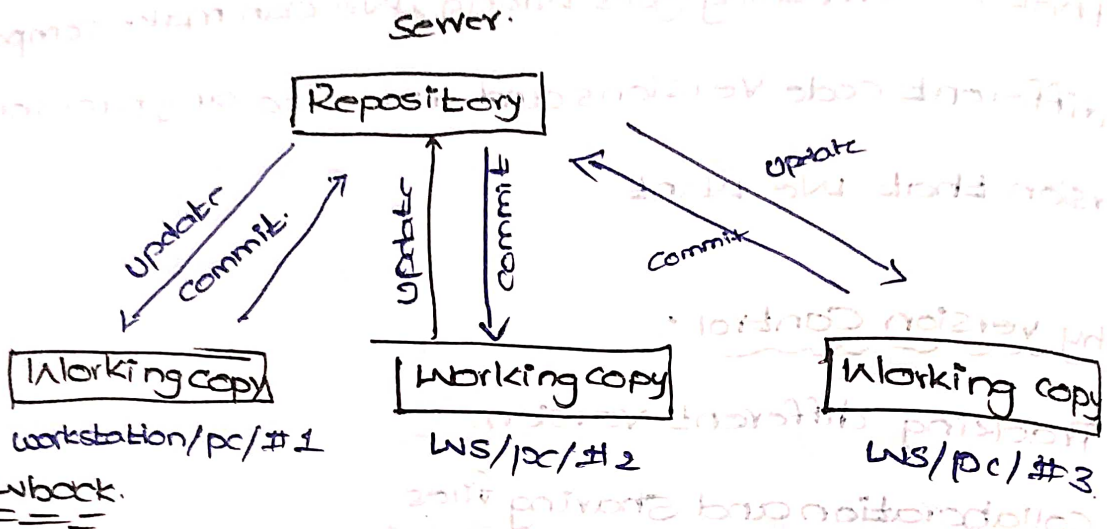
CVCS.



* CVCS *

→ CVCS uses a central server to store all files and enable team collaboration.

→ CVCS works on a single repository to which user can directly access to central server.



Drawback.

→ It is not locally avialbe.

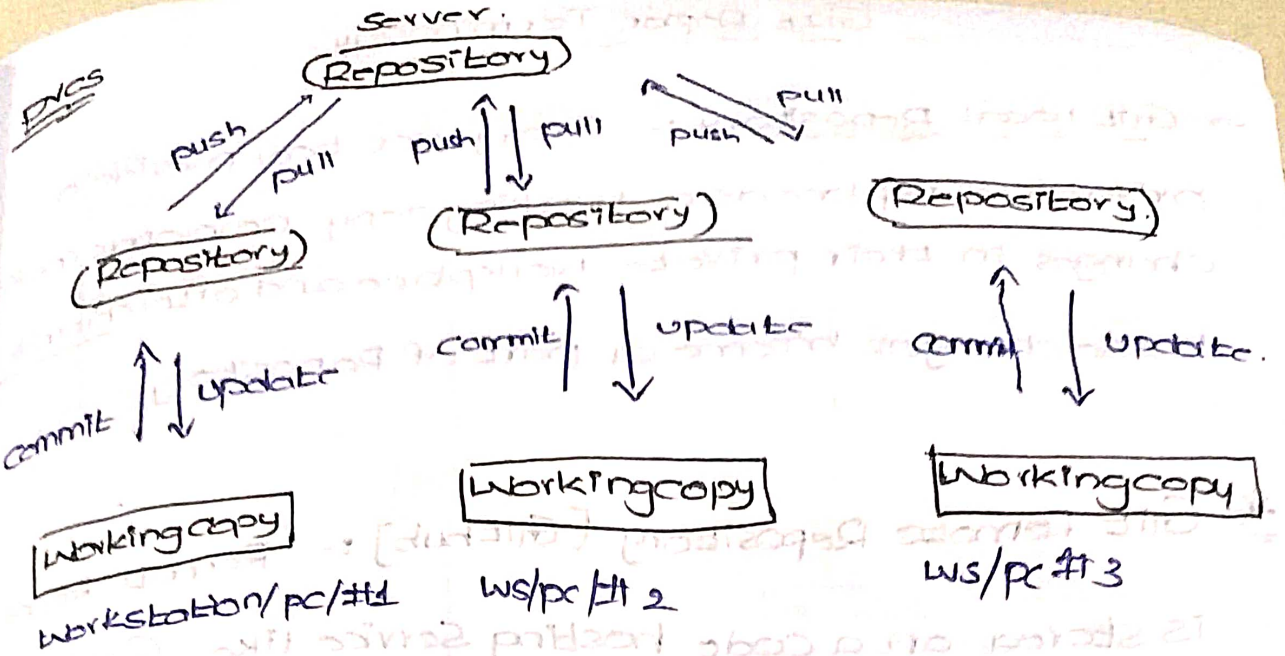
→ crash of CVCS will result in losing the entire data of the project.

* DVCS * [Realtime maintain]

→ In Distrubuted VCS, every contributor has a local copy or clone of the main repository.

→ User can change and commit local Repo without any interference.

→ User can update their local Repo from the Central server.



→ If central server get crashed at any point of time the last data can be easily recovered from any one of the contrubuitor's local repositories.

GIT [version control]

→ Git is a "version control system" [VCS] (or) Source code managment [SCM] Tool.

→ Git is a Distrubuted/Decentralized version control system, meaning your local copy of code is a complete version control repository. Most operation are local

→ Simple way to keep multiple version of a file (or) directory.

→ Benifits of Git:

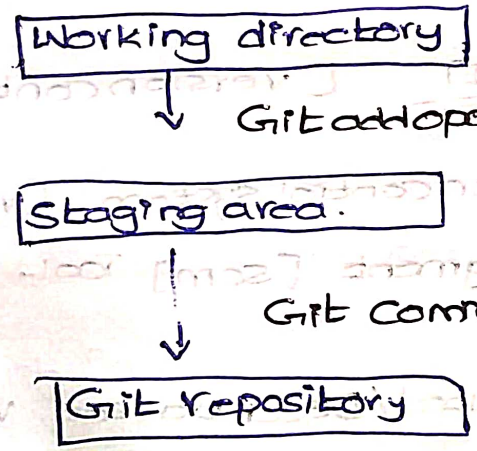
- Free & opensource.
- Simultaneous development.
- Faster-release.
- Built-in- integration
- Strong community support
- Git work with your team
- pull request
- Branch policies.

Git Repository Terminology:

→ Git local Repository :- Every vcs tool provide a private workplace as a working copy. Developers make changes in their private workplace and after commit, these changes become a part of Repository.

* Git Remote Repository [Github] :- Remote Repository is stored on a code hosting service like Github (or) an Internal Server.

Stages of GIT:



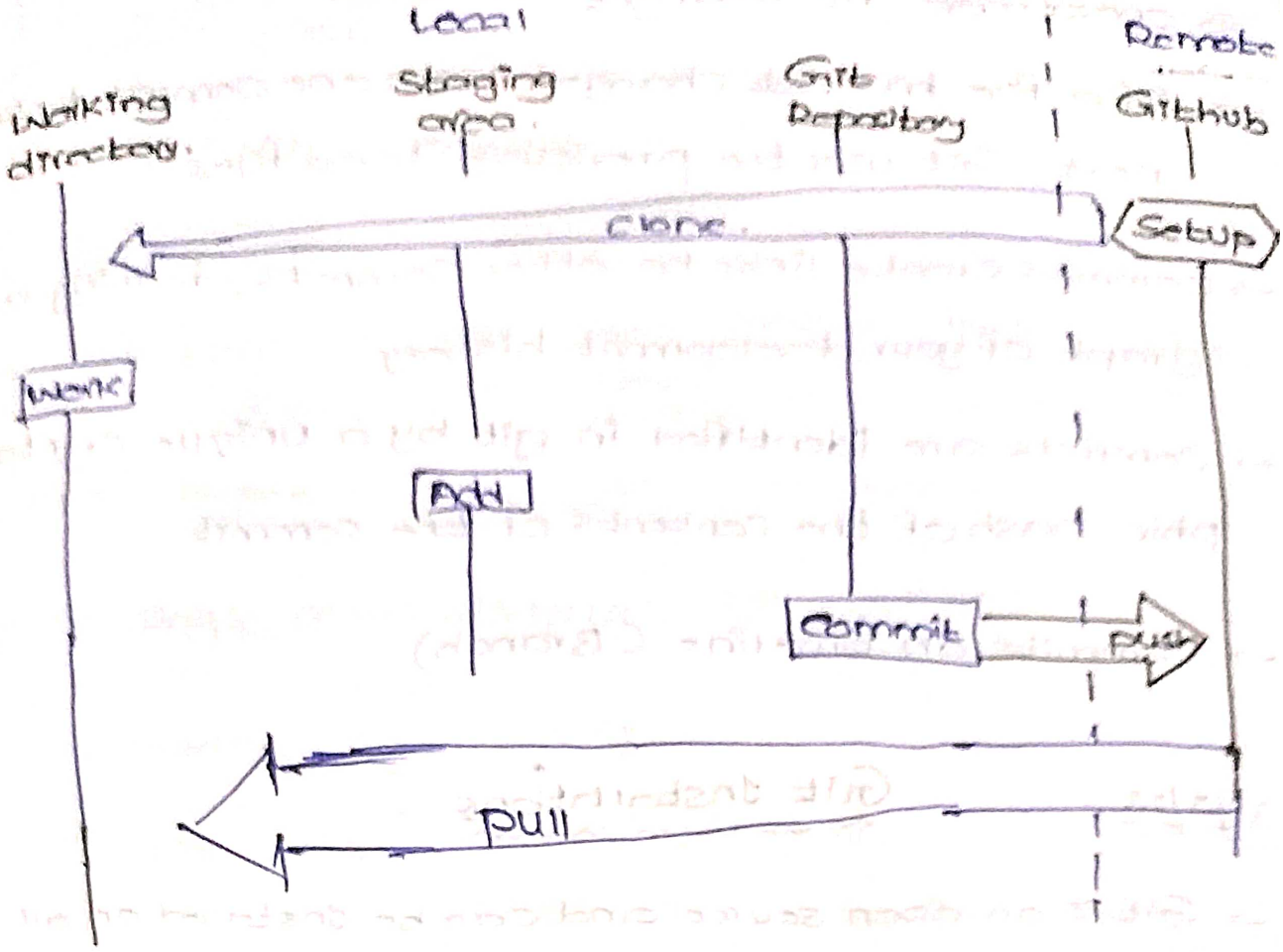
→ Working directory :- Area of live files, also known as untracked area of Git.

→ Staging area :- Staging area is when git starts tracking and saving changes that occur in files.

→ Git Repository : Also called 'local Repo' is your.

• "git" repo • it's area where GIT save everything
↓ folder → automatically create when run init

Git Workflow:



→ Add → commit → push → pull

* Git Basic Terms *

Repository:

→ Collation of files managed by git.

→ Git can be initialized on a project to creat a Git repository.

→ A Git repository is the .git folder that contain all your necessary repository files.

→ It is a working directory/workspace.

Commit:

→ A commit is a snapshot of all your files at a point in time.

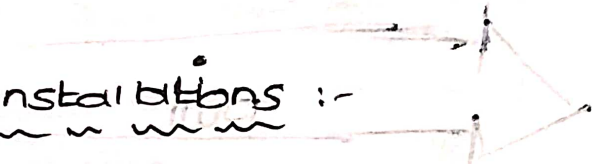
→ one or more file changes.

→ If a file has not changed from one commit to next, Git uses the previously stored files.

→ commits create links to other commits, forming a graph of your development history.

→ commits are identified in git by a unique cryptographic hash of the contents of the commit.

→ commits on timeline (Branch)



31/1/23 Git Installations :-

→ Git is an open source and can be installed on all major os. → Windows, unix/Linux, Mac.

→ Installing git on windows:

→ Google → git install → git scm (Windows) → 64bit Standalone

→ open Gitbash [command prompt]

git version

* Installing Git on Ec2 Linux :.

Ec2 → Instance → Launch Instances - Git, Linux (Amz)

(port 22) SG → SSH → 22, HTTP → 80 } Enable → Launch Instance.

→ yum update -y.

→ yum install git -y.

→ git version.

→ git help → clone add restore bisect grep
init mv rm diff log

show status.

* For single command:- help:- git add help.

* Creating Workdirectory :-

git init [Repository creation].

git init. → Initialized it. Working directory location.

→ ls -git

(.git)

→ make a project:-

mkdir project

touch aws.

git status

git add

* GitHub:- Github is an internet hosting service for Software development and version control using git

→ It provides the distrubuted version control of git access control, bug tracking, Software features request, task managment, continuous Integration.

* SIGnup git Account:-

Github → signup.

1/2/23

* Git Commands:

open gitbash in the window

→ git version

git help.

create a project: mkdir myprojectsgit

cd myprojectsgit.

pwd.

→ git init

ls -a → create a folder, and change the cloud.

mkdir cloud.

cd cloud.

pwd.

→ Now initialize → git init → .git file created

ls -a

ls .git

Setting user profile. [Git configuration]

↓

Which user update the code

→ Syntax:

git config --global setting-value.

git config --global user.name "Raju"

git config --global user.email "rnrajuuu@gmail.com"

To check: git --config --global --list

Files are available in the location :- cat ~/.gitconfig

- Color Settings:

git config --global color.ui true

status color:

git config --global color.status auto → default.

git config --global color.branch blue.

list: git config --global --list

Apply commands

1. pwd.

git status.

cat > script.sh. → #!/bin/bash.
This is my 1st programm
echo "Welcome to Git" → ctrl+d [save] → create a file on local repository.

2. git status.
↓
untracked file.

3. → git add script.sh → track [move to stagging area]

git status.

4. → git commit -m "Three lines are added" → message

ls.

Step1:
→ To modify a file:- vi script.sh.

```
#!bin/bash.
echo "welcome to git"
uname -r [kernel release]   (ctrl+d)
```

→ git status

Step2: git add script.sh

git status.

git commit -m "Fourth line has been added"

* Ex2.

Step1: cat> file1
Hii
cat> file2.
Hello
} ctrl+d

→ git status [verify]

→ read all files at a time.

Step2: git add .

git status.

git commit -m "Two files are created"

Ex3:
* vi index.html

```
<html>
<body bgcolor:red text=blue>.
<marquee><H1> welcome to devops </H1></marquee>
</body>.
</html>
```

git status.

git add index.html

git status.

git commit -m "

To restore a file from staging to Working area.

git restore --staged index.html

git status.

modify sourcecode. vi index.html → add smtg to the file, :wq!

git status.

git add index.html

git status

git commit -m "New Webpage is created"

* git diff → It shows changes between commit, commit and working tree -- and so on..

→ modify the File → vi file1 → Hi
Hello
Gm :wq!

→ git status [modified filename]

→ git diff → Result

a/file b/file1
---
a file1
+++ b/file1 → modified file

→ git add file1

→ git diff.

→ git commit -m " Two lines are appended " → after last line

git diff

git diff file1

* git log :- It shows the commit logs

→ git log → to check commit logs

→ List only Raju committed file :-

git log --author = "Raju"

git log -- author = "rmraju.uu@gmail.com"

* Removing Files :

→ create a file add info :-

cat > debug.log

→ a
b
c : ctrl+d

→ git add debug.log

git commit -m "log file added"

git status → Reporsitry area.

To remove: git rm debug.log } verify

git status.

git commit -m "log file has been removed"

git Status.

* Moving Files :

→ ls.

→ git mv Script.sh devops

↳ file need to moved

↳ Newfolder I want to be created

GitHub

1/2/23

- GitHub is a code hosting/sharing platform for version control and collaboration.
- Anyone can signup and host a public code repository for free which make GitHub especially popular with opensource projects.
- Using GitHub, you will
  - → creat and use a repository.
  - → start and manage a new branch.
  - → making changes to a file and push them to GitHub as commits
  - → open and merge a pull request.

* SignIn GitHub → Create Repository → Repositryname (devops) → Description (my personal project) → pubik (✓) private
- → Add readmefile → Create a Repository.

* Create a file to add to main branch.

addfile → docker. → This is a docker textfile → (frame)

commit new file (docker file is created) → commit

* Clone the repository:

Go to gitbash → cd git-projects/

ls.

undercode in github copyurl [HTTPS]

→ git clone -b main https://github.com/sysgeekuu/devops.git
                ↳ branch name

→ ls [to verify whether it clone or) not]

→ cd /devops/
  ls → output. Readme. docker.

How to commit newfile:

→ cat > ansible.
  Hi
  Helb.

→ git status.

→ git add .

→ git -m "ansible file is created"

→ git status.

To push a commit in to Remote.

git push origin master main → (add credentials)
                                ↳ branch name.

→ To set branch name presisentently :-

git push --set-upstream origin main

Test:
  vi ansible
  Hi
  Hello
  ansiblefile → 3rd line added : !#!2!

  git add .

  git commit -m "Third line added"

  git push

→ from github

* pull the source code:

Add file → create new file → kubernet.yml → (clusterfile)

commit new file [yml file is created] → created

→ Gitbash

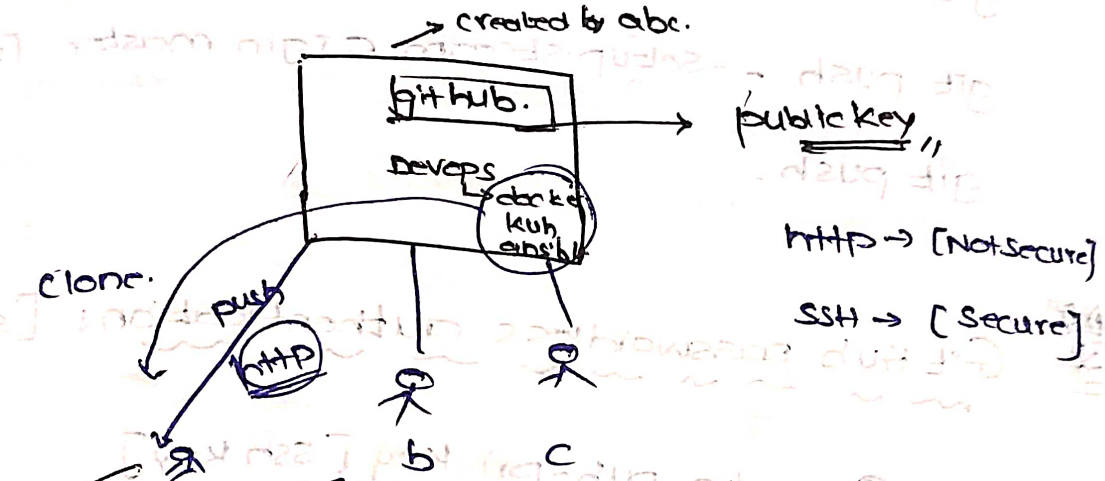
→ To check repository name:-

git remote -v

→ git pull

→ is

→



2nd Scenario:

Adding Remote Repo to an existing local Repo:-

→ cloud.

devops.

→ Create a Repository → cloud → public → create a project

(project's already exist) → We need to add

→ adding repo :

mkdir cloud.

→ cd cloud.

ls. git init

→ git remote add origin https://github.com/syam4u/cloud/.git

"rm -rf cloud" ~/git-projects

create files

touch aws azure gcp

git add .

git commit -m "Three file are created"

git push --set-upstream origin master. [setup]

git push.

* Git Hub passwordless authentication : [SSH]

Step1. Generate pub-pri key [ssh key]

ssh-keygen [~.ssh/ id_rsa] → 3times Enter

↓ names

cd ~/.ssh

ls → id_rsa (private), id_rsa.pub → public

→ cat id_rsa.pub [openfile]

↓

copy Entire code, [=]

→ Then go to git Hub, click on account name → Settings → leftside (SSH and Gpg keys) → click on

New sshkey (paste key) → Title name: mylaptop.
Keytype :- Authentication.
→ Add.

confirmation.
confirm → Succesfulladded,

Testing SSH connection:
→ Testing.
ssh -T git@github.com → yes → Succesful added

cd git-projects/
cd cloud.
touch oracle.
gitadd.
git commit -m "newfile oracle is created"
git push

3/2/23 :- Git Branch

→ A branch is a version of the Repository that diverges from the main project.

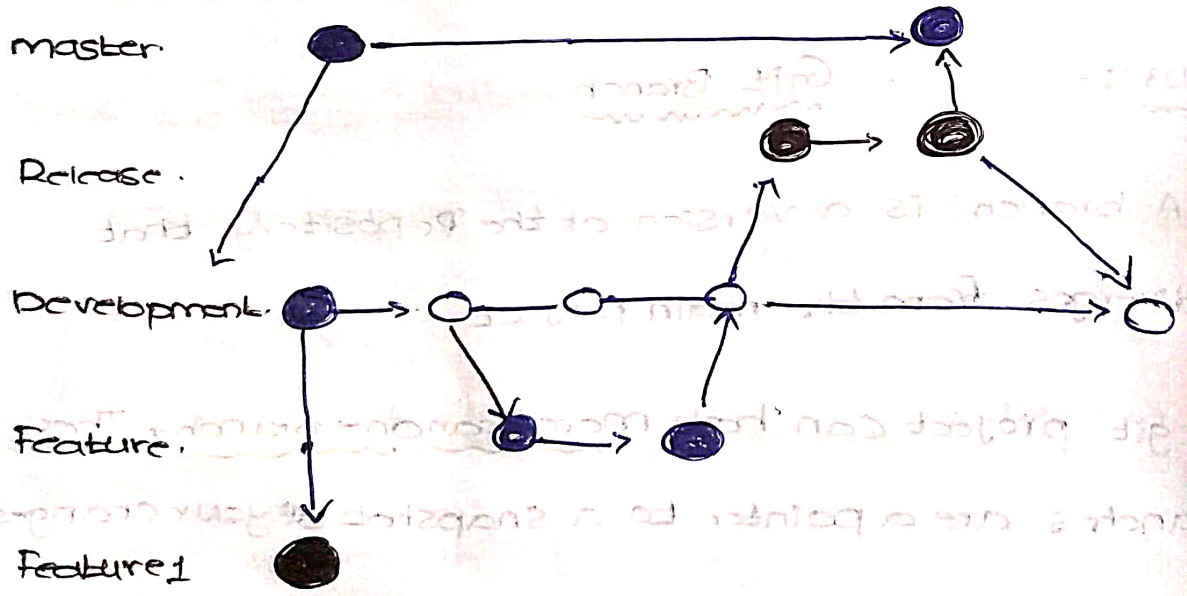
→ A git project can have more than one branch. These branches are a pointer to a snapshot of your changes.

→ A branch in Git is simply a lightweight movable pointer to one of these commits.

→ The default branch name in Git is Master.

Real World Branching Scenario :-

1. Master Branch :- production ready copy.
2. Development copy :- It is a developer branch where Continuous work will be done.
3. RELEASE BRanch :- This branch is created from the development branch to make it ready for the relea... and it is used for bug tracking and documentation purpose.
4. Feature Branch : Whenever the developers working on the new features, they use a feature branches and commit the code to that branch.



- To check current branch :- git branch.
- " " Remote branch :- git branch -r. (Github)
- " " All branch name :- git branch -a

Create a new branch:- git branch development ← Bname

Check:- git branch

*→ Indicate current branch

How to switch current branch to another:-

git checkout/switch branchname.

git checkout development. [o/p :- * development]

* Create a file under development.

vim program.py.

print ("Hello World") :wq!

git add .

git commit -m "python program has been created"

git push --set-upstream origin development.

→ Go to Github

↘ Check the branchname (development)

→ Create a two folder under development.

mkdir aws.

cd aws

cat>awsfile1: Hi

cat>awsfile2: Hello [ctrl+d]

→ git add .

git commit -m "Two files are created"

→ cd..

2nd,

mkdir azure.

cd azure.

cat> azurefile1 : Hi

cat> azurefile2 : Hello Gm.

git add .

git commit -m "Two files are updated by azure"

cd..

ls → aws azure

-> git push origin development. [branches updated by development]

* Create a new branch with name feature.

git branch feature

git branch

git checkout feature [switching branch]

mkdir gcp.

cd gcp.

touch gcpfile

git add .

git commit -m " File is created by featur branch"

git push origin feature.

Git diff: It shows changes between commits, commit and working tree.

git diff development feature.

merge: branch :-

source ---> destination.

To merge (feature) branch to (development).

git checkout development.

git merge feature [select all files, then we need to push to development]

git push origin development

To merge development branch to master.

git checkout master.

git merge development.

git push orgin master. → production copy ready and release.

* Deleting branches:

→ delete

git branch -d feature.

git branch -d development

} locally branch deleted ↓ it placed in. Remote.

** Deleting Remote branch:-

git push origin : refs/heads/feature.

git push origin : refs/heads/development

!

CheckIn Github.

* git clean :- It remove untracked files from the Working tree.

→ git clean -n [or] git clean -f

* Deleting a Repository Github :-

Github → Settings → Delete this repository (and log write repository name [cloud] → passwd → delete

* Interview Q/A:

Difference between git fetch & pull :- → latest changes,

Git-fetch: Download objects and refs from another repository. It downloads only latest changes into the local repository. It downloads fresh changes that other developers have pushed to the remote repo Since the last fetch and allow you to review and merge manually at a later time using git merge. because it doesn't change your working directory or the staging area, it is entirely safe, and you can run it as often you want.

* First create a file in remote repository.

git status.

git fetch.

git status

git log

git log origin/main [logson remote repository

merge
git origin/main.

git status.

is .

Git pull : It download the latest changes in to the local repository and it also automatically merge changes in your working directory. one important thing to keep in mind is that will merge only in to the current working branch.

∴ **gitpull = git fetch + git merge**

* 6/2/23

WEB Servers → To hosting websites,

A Web server is a network service that servers content (webpages) to a client over the internet. Web.

→ Webservers is also known as HTTP (HyperText Transport protocol) Servers.

→ A popular webserves in linux are : 1. Apache HTTP server.
2. NGINX Server.

** Apache HTTP Server:

→ Apache is a open-source webserver developed by the Apache Software Foundation (ASF)

→ This is a solid and stable Web server that has been

→ It is also an option to use the SSL protocol,

making website safe and secure.

Https: 443, port number

http: 80

Interview Question

main configuration files (pre-requisites)

1. package : httpd.
2. Document root location : /var/www/html
3. Configuration files : /etc/httpd/conf/httpd.conf

/etc/httpd/conf.d/ssl.conf

4. Auxiliary directory : /etc/httpd/conf.d/
5. Default web page : /etc/httpd/conf.d/welcome.conf

auxiliary

6. modules location : /usr/lib64/httpd/modules.
7. Log Files location : /var/log/httpd/
8. Log Files : access_log, error_log
9. Service/Daemon : httpd
10. port : httpd - 80

https/ssl - 443

Installing & configuring httpd:

Create EC2 linux → Webserver (Redhat linux) → t2.micro.

ports → SSH [V] → launch. → Connect instance.
HTTPS [V]
HTTP [V]

step1 To change hostname : hostname Webserver.

vi /etc/hostname. Webserver : wq!

→ check ipaddr: ip a.

step2. open etc/host files. → vi /etc/hosts.

172.31.61.43 Webserver : wq!

step3. bash.

yum update -y (yum = dnf)

cat /etc/redhat-release. [Linux version]

uname -r [Linux kernel version]

step4. yum install httpd -y [Webservers] installation

httpd -V

→ Systemctl start httpd.

Systemctl enable httpd.

Systemctl status httpd

→ To check which ports : opening :- netstat -pantl

yum install net-tools -y

Webserver (google) // Result

* By default Apache loads, welcome.conf file.

Cat /etc/httpd/conf.d/ welcome.conf

auxiliary location.

* Go to document root location:- cd /var/www/html

ls.

vi index.html

```
<html>
<body bgcolor=red text=yellow>.
<marquee><H1> Welcome to Webserver...</H1>
</marquee>.
</body>.
</html>.
```

:wq!

→ google ip and Check Result.

↳ custom pages are mainted

→ If document root is empty it will redirected to default file welcome.conf.

* logfiles.

cd /var/log/httpd/

ls.

tail -f access_log. → monitor who log [particulary by typing ip address,

tail -f error_log.

NGINX Webserver.

→ The NGINX server is an Apache alternative server.

→ NGINX is faster than apache webserver

→ NGINX is "High performance" and modular service that you can use for:

1. Webserver.
2. Reverse proxy
3. Load balancer.

Note: By default nginx act as a "Web server".

main configuration files (pre-questions)

package : nginx

document root location : /usr/share/nginx/html

configuration files : /etc/nginx/nginx.conf.

logfiles location : /var/log/nginx

logfile : access_log, error_log

Service/Daemon : nginx

ports : Httpd-80, Ssl-443

* create another Ec2 instance → Launch instances,

Connect NGINX Server,

→ hostname nginx Webserver.

vi /etc/hostname. : Webserver :wq!

ifconfig.

vi /etc/hosts : Ipaddress hostname :wq!

→ bash.

yum update -y

Installing nginx Webserver :-

→ yum install nginx -y

→ Systemctl start nginx.
Systemctl enable nginx.
Systemctl Status nginx
} starting the service nginx

. yum install net-tools -y

To check port :- netstat -pantl

) To check default Webpage :-

cd /usr/share/nginx/html

ls → o/p :- index.html

vi index.html → modify and check :wq!

Refresh & Check
Webpages

Creating Custom page :-

vi /etc/nginx/nginx.conf

↓

Go to Server Section.

server {
listen: 80

server_name example.com ← domain created

root /var/www/example.com;

access_log /var/log/nginx/example.com/access.log;

error_log /var/log/nginx/example.com/error.log;

:wq!

→ create a directory under :-

mkdir -p /var/www/example.com.

cd /var/www/example.com/

vi index.html

<html>
<body bgcolor = yellow text = blue>.

<marquee> <H1> Welcome to nginx </marquee></H1>

</body>

</html>. :wq!

and location.

→ mkdir -p /var/~~www~~log/nginx/example.com.

→ System restart nginx
ctl

→ Trouble-shoot :-