**Technologies**

## *GIT:*

Git is a free and open-source distributed version control system.

It is developed to manage projects with high speed and efficiency. The version control system allows us to monitor and work together with our team members at the same workspace.

A version control application allows us to keep track of all the changes that we make in the files of our project

in GIT, The HEAD points out the last commit in the current checkout branch. It is like a pointer to any reference. The HEAD can be understood as the "current branch".

git workflow

-------------

working directory --add--> staging area ---commit--> local rep(.git) --push--> remote repo(git hub, bitbucket etc)

Commands:

git --version

clear

git config --list //shows all details of user

git config --global

git config --global -e //to edit

git config --global user.name “fayaz.shaik”

git config --global user.email fayaz.shaik@google.com

git config --global https.proxy 192.168.111.52:8050

git config --global http.proxy 192.168.111.52:8050

git config --global --unset https.proxy

git config --global --unset http.proxy

git config --global alias.<alias name> "command name to be aliased" //aliasing any command temporary

---> Ex: git config --global alias.allcommits "log --oneline --graph --all"

alias np=notepad //aliasing editor temporary aliasses

git config --global core.editor "location url of notepad++ in C" //to set thirdparty app as default editor

git config core.editor //to see which editor is there by default

git help <keyword (or) git <keyword> --help

<name of that app> // to use any third party app, if not configured add path in system variables

git remote //to know the name of remote repo

git remote add origin <destination git url> //to link to remote server ie, github

git remote set-url origin <destination git url> //if already added then use set-url to add new git URL

git clone <repo-url> <preferred name if u want> //to copy original project from github

git clone -b <branch name> <branch url> //it clones specific branch in given branch

git fetch origin //to fetches the changes in remote to local origin/master then we merge it with master branch to see new files added in remote repo

git pull origin master // it fetches and merges in master branch automatically

git init //initialize the git folder(.git is ceated)

git status //to view all the files(changes) which are not staged.

git add <updated file name> //to stage the changes, if u want all files then add .(period) after the add

git commit -m "message to commit" //to save in local repo

git push -u origin <branch name> //to push into remote repo or git Hub

---> create .gitignore file and mention the file names or patters(ex: \*.txt) in it those u dont want to push into repo

git diff //same as status but shows actual difference in content ,it has to be in stagged stage

git log // gives history of commits in project

git log --pretty=oneline or full or fuller //based on our req

git log --oneline --graph --all // graph used to show branches as tree &all is used to see logs of all branches

git branch < new branch name> //creating a branch

git branch -a //to see all branches

git checkout <branch name> //to shift to mentioned branch

git checkout -b <new branch name> //creates a new branch and automatically switches to that branch

git branch -m <old name> <new name> //to rename old branch with new name

git branch -d <branch name> //to delete a branch

git merge <banch name> //merges given branch name with present branch(the branch we present), gives graphical view in log

git rebase <branch name> //merges but didnt give graphical view in log

########### p4merge tool is used to see diff and merge procees in easy & simple way ###################

git stash //deletes content that's in staged stage & sent to dustbin(backup).we can take it back if we needed.

git stash list //to see stashed list

git stash apply //it takes back the latest stash in the stack flow

git stash apply stash@{id of stash} //to get a particular stash ex.stash@{2}

git stash pop //takes back latest stash from stack & drops that from stack automatically

git stash drop // deletes latest stash in stack

git cherry-pick <commit-id of another branch> // for applying same commit from one branch into another branch(give another branch commit ID in required branch)

git clean -f -d -x //permanently deletes untracked files only( -f means forcely, -d means directory, -x means .gitignore)

git tag <tag name> //to mark or tag a commit

git tag -a <tag name> //-a for animated tag

git tag //lists all tags

git tag -d <tagname> // to delete a tag

git restore <filename> //to undo changes made in working directory stage (Before add)

----or----

git checkout -- <filename> // same as restore, but for undoing multiple file use dot(. // space should be there btw -- and filename

git reset <file name> //to undo after add

git reset <commit id> //undos commit (use commit id that u want to go back to)

Generating a new SSH key

-------------------------

ls -al ~/.ssh //to see all the avialable shh keys in our PC

ssh-keygen -t rsa -b 4096 -C "your\_email@example.com" //to generate ssh keys

eval "$(ssh-agent -s)" //to runssh-agent

ssh-add -K ~/.ssh/id\_rsa //to add ur private key to ssh-agent

clip < ~/.ssh/id\_rsa.pub //or add shh public key in guthub>settings>ssh & gpg keys>save by copying from .ssh folder

## *DOCKERS:*

Docker is an open source containerization platform for building, deploying, and managing containerized applications.

What exactly is a Docker image?

A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform.

Docker uses container on the host's operating system to run applications. It allows applications to use the same Linux kernel as a system on the host computer, rather than creating a whole virtual operating system

It provides a convenient way to package up applications and preconfigured server environments, which you can use for your own private use or share publicly with other Docker users.

Container----> A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably anywhere.

The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage

Commands:

docker images

docker --help

docker search <img name> //to search images in registery

docker ps -a //to see all containers

docker ps // to see only the running containers

docker ps -l //to see last exited container

docker kill <container id> //to stop container

docker rm <container id> //to remove/delete a container

docker rmi <image id> //to remove an image

docker pull <image name> //to only pull the image to ur local repo

docker run <image name or id> //it pull img from dockerhub if it's not exist in local repo & runs automatically

docker run --rm -it <img name> //it starts the container and deletes it when the process got completed.

docker exec -it --name <container name> <image name/container id> bash //or use run instead of exec, to enter into image or creating the container from root ( /bin/sh means in shell mode & -it means interactive terminal)

docker run -d -p <ports> <our img name> //to run our custom img on our ports( -d : It is used to create a daemon process or -d means detached mode i.e, it runs in the background of terminal so, we can do another work.)

docker run -it --v <location> --name <container name> <image name>

docker attach <container name/Id> //to jump back into running container i.e running in detached mode

docker commit <container Id> //to convert containers to images(after commit, it contains all the modified files)

docker start <container id>

docker stop <container id>

docker restart <container id>

docker build -t <img name> . //-t means tag i.e tag the given image name to this img & . means the docker file is in current path only, otherwise add path

docker network create <network name>

docker network ls

docker run -it --net <network name> --name <container name> <image name> //to start in particular network

docker system prune //deletes all stopped containers, dangling imgs, not used networks

persistent volumes---permannent

ephemeral volumes---temporary

docker volume create <vol name>

docker volume rm <vol name> //to remove vols

docker volume ls //to list them

--> Docker Compose is designed to run multiple containers on a single host system. it uses yaml to manage diff containers as a single service.

docker-compose up -d //to execute compose.yml file

docker-compose down //to stop it

Dockerfile (no extension)// A text file with instructions to build image. Automation of Docker Image Creation

------------------------------

FROM <image name : version>

MAINTAINER <name> <email>

RUN apt-get update && \ // here, \ means there is another line below which belongs to this command

apt-get install -y <any libs>

WORKDIR

ADD or COPY

EXPOSE <port no>

CMD [ <commands> ]

ENTRYPOINT <commands>

VOLUME <path>

Docker-compose.yml

-------------------

version: '3'

services:

web: //any name

image:

build: <directory to build img>//if we didnt want to use dockerhub img

ports:

- <ports>

environment:

- <bla bla ...>

- <bla bla ...>

volumes:

- <path>

database:

image:

ports:

- 8080:80

depends\_on:

- web //any name that presents there above

examples:

----------

From ubuntu

MAINTAINER fayaz

RUN apt-get update

CMD ["echo","hello dockers"]

FROM openjdk

MAINTAINER fayaz

WORKDIR /code

ADD Demo.java /code

//we should write environment variables here if we dont have them in our system

RUN javac Demo.java

CMD ["java","Demo"]

FROM platform/images/alpine-extended:1.0.0

ENV \

CUST\_PATH=./customization \

HOME=.

USER root

COPY run.sh $HOME

COPY dop-tasks-list.json $CUST\_PATH/

COPY dop-projects-list.json $CUST\_PATH/

RUN chmod 755 $HOME/run.sh

WORKDIR $HOME

ENTRYPOINT ["./run.sh"]

Docker Kubernetes

------ ------------

Docker is developed by Docker Inc. Kubernetes is developed by Google.

It was first released in 2013. It was first released in 2014.

It is a container based technology used to create isolated environment for applications. It is an infrastructure for managing multiple containers.

It allows us to use third-party tools like ELK for logging and monitoring. It allows us to use in-built tools for logging and monitoring.

Its public cloud service provider is only Azure. Its public cloud service providers are Google, Azure, and AWS.

It is less customizable. It is highly customizable.

Its container limit is 95000. Its container limit is 300000.

It is easy to install. It is complex to install.

It cannot do auto-scaling. It can do auto-scaling.

It does not provide any dashboard. It provides a Web UI dashboard.

Imp Question: https://cloudchamp.notion.site/Docker-Interview-Questions-9317cee2392147798571c09cf27bad85

## *KUBERNETES:*

Kubectl get nodes -o wide

k -n dop get all

kubectl -n <namespace> get pod | grep <podName>

K get secrets

K get config

k get pod -n dop |grep -v Completed |grep 0/1

K exec -it <pod\_name> -- bash or -- bin/sh

K -n dop get all | grep debug (Searches for all files & filter with name debug)

K -n dop describe pod <pod name>

K -n dop logs <pod name> -f --since 1m or 5m

k -n dop logs onsi-servo-68566f65bd-ldz6x --since 5m > servoLogsNew.json

k -n dop get logs -f app=<ms name>

kubectl -n <namespace> logs -f deployment/<app-name> --all-containers=true --since=10m

kubectl -n dop expose deployment.apps/onsi-servo --type=NodePort --name=servo-exposed

kubectl -n dop expose service servo --name=servo-debug --type=NodePort --port=9527

## *Helm:*

Helm ls -A

Helm ls -n dop

Helm repo list

Helm repo add <repoName> <repoUrl>

Helm repo update

Helm uninstall <chart> -n <namespace>

Helm fetch <repoName>/vfgs-dop <tlc version>

Helm get values onsi -n dop | grep oni

helm history dop -n dop

helm rollback dop <revision-number> -n dop

## *Unix:*

touch,find,ping,curl -0,ssh, ps ax, top,kil <pid>, |(append or pipe)

man <command> ---> gives manual page... information about commads

info <command> ---> gives detailed info abt a command

whatis <command> --->gives purpose of cmd

who ---> lists the name of all users curently logged in.

who am i --->displays user name

hostname --->displays host name(hostname -i --->displays ip address of hostname)

pwd ---> shows present working directory

date ---> current date

cal ---> print current month cal (cal <year> - give full year calender)

clear ----> clears sceen

mkdir [dirName]----> Ceate a new directory

mkdir -p welcome/new/images ----> creates intermediate directories i.e; directories within dir's

mkdir 'my computer' ---> to create a dir that contains ---spaces---.

rmdir [dirName]---> to remove empty dir

cd [dir name] ---> to move inside that directory

cd .. ---> to go back to previous directory(parent dir)

cd ../../.. ---->to tell how many times to go back(Here; 3 times)

cd ---> go's to home directory($)

echo <just prints the matter given here>

echo > <filename> //to create a file > means create a file with the name given next to it

dir //lists the folders in it. same as ls (for windows)

type <file name> //shows the content inside of file like cat command in bash

cat filename.txt -----> to see data inside file

cat > filename.txt ----> to create file and can directly enter data in a file(to exit press---> CRTL+C)

touch <filename> ---> to create a file (with this we can create empty files & hidden files only) //touch ex.txt & touch .dir1

less filename.txt ----> to see data in file only(to quit press--> q)

vi filename.txt ----> to see and modify data in it(to modify data press INSERT...to quit press Esc-->type :wq! to save)

head <filename>---> optputs the first 10 line of a file

rm [filename]---> to remove/del a file

rm -rf <folder to delete>

rm -r [dir name]---> danger cmd i.e; it deletes all files & subDir within a dir and dir itself.

cp [soucefile] [newfile/dir] ---> copies data from source to newfile/ dir

mv [old file] [new file/directory]---> to rename files /move files to another file or directory

grep ----> to search for patterns in a existing file(Global search Regular Expression and Printed)

grep 'fayaz' filename ---> search fayaz keyword in given file name

grep -c ---> counts the matched keyword in file

grep -i ---> ignores caseSensitive of upper & lower cases while searching.

grep -v ----> to print lines that do not contain or match the expression.(ignores given pattern)

history -c ---> to clear entire history

chmod---> to change access/permissions of one or more files(only super user should change)

chmod <permissions> filename //chmod 641 home

logout or exit ---> to quit from unix

df ---> tells how much disk is used and how much is free

ulimit 10000---> tells that the max file size(10000 bytes) that a user can create.

bc ---> u are in calculator mode(to exit type --> quit)

factor ---> u r in prime factor mode.. It displays prime factors of a num(to quit press---> CTRL+C)

nslookup oss-ext1-solstice.vfde.corp.amdocs.aws //for ip of dns

scp <file/folder name> root@<master-IP of target env>:<location in target env> // to copy files/certs from one machine to another

Ex: scp VFGS-ASOM.cer root@10.246.34.12:root/keys/kubernetes

Curls:

curl --location 'https://{{host}}/dop/security/rest/loginservice/login'

--header 'Content-Type: application/json'

--data '{

"userID": "OHProxyUser",

"password": "OHProxyUser123"}'

curl -X POST 'https://api-xtest-1.x-test03.aws.solstice.vodafone.com/accountManagement/v1/billingAccount'

-H 'Authorization:Bearer xxx'

-H 'Accept:application/json'

-H 'Content-Type:application/json'

-d '{

"extensions": {

"migrationInformation":

{

"migrationProcess": "MoD"

}

}}'

FILE SYSTEMS:

ROOT--->file systm is organisedd as a tree with single root node ROOT(.root dir is denoted as /)

DEV --->contains files that control various ip/out devices like terminals, printers,disks etc

BIN --->it contains executable files for most of the unix commands

TEM --->contains temporary files created by unix or user

USR --->contains several dir's each associated with particular user(there is anothe BIN in USR that contains additional cmds)

ETC --->contains binary executable files req for system administration

LIB --->it contains all the library functions provided by unix users

UNIX--->has some predefined directories

VIM: (Advanced VI editor):

3 modes:

**command mode** - initally we are in this mode, but to move explicitly press ESC

**Insert mode** - when we press i/insert to edit text

**Extended command mode** - when we press : then follwed by q(quit), w(save), wq(save & quit), q!(quit forcefully)

All below cmds has to be done in command mode(ESC)

/ -> to search (n -> to move to the next selection)

gg -> to go Beginning of page (first line)

G (shift+g) -> to go to End of page (last line)

yy -> to copy a line

nyy -> to copy n number of lines from cursor, n=number ex: 2yy, 5yy

dd -> to cut the line

p -> to paste the line below cursor point

P (shift+p) -> to paste the line above cursor point

u -> to undo last change (word)

x -> to delete letter by letter

dw -> to delete word by word in a line

dd -> to delete entire line

ndd -> to delete n number of lines from cursor

sudo -i -> to move to the root user

ls -l -> to list with more info

mv file1.txt dir1 -> to move a file to given dir

mv -r dir1 dir2 -> to move one dir to another

touch file{1..5}.txt -> to create multiple files ex: given cmd creates 5 files with names file1.txt, file2.txt...file5.txt