|  |
| --- |
| Installed ubuntu VM using Virtual box on windows OS |
| Learned basic linux commands like pwd,ls,cat,tail,head,touch,mkdir,cp,mv,wc,ln,locate,echo....etc and executed them in the terminal |
| Learned to use vim and nano text editors in linux. Learned to add and modify text using these editors in terminal |
| Familiarized network commands netstat, ss, ifconfig,traceroute and its uses . Also executed and tested it in terminal Familiarized monitoring commands ps,top,htop,iotop.Tested its real time customization on terminal |
| Familiarized text manipulation commands awk: printed fields and columns of file Filtered text and strings using patterns Familiarised comparison operation based expressions Used compound expressions and user defined variables...etc sed: Displayes specific range of lines of file Replaced words and characters of a file Inserting blank lines .....etc Also familiarized grep,tail,head and executed them in terminal |
| Familiarized linux file permissions , how to modify user,group,others permissions and giving special permissions using chmod command. Learned changing file ownership using chown command. Familiarized securing linux server using Uncomplicated firewall(UFW), Familiarized firewall-cmd command by adding services, ports ...etc to firewall |
| Automated tasks like checking status of services,changing permissions of files having full permissions,checking system informations,Updating database ....etc |
| Executed shell scripts using conditional statements and arithmetic,logic and relational operators. |
| Write shellscript to backup the files given as argument with file name current date and time.Executed in terminal and tested the results |
| Write shellscripts using FOR,WHILE,UNTIL and SELECT loops. Executed them in terminal and tested the results. 11. Installed packages using apt command. Familiarized enabling/disabing universe,multiverse and restricted repository on ubuntu. |

Linux structure :



**[ec2-user@ip-10-0-10-195 /]$ ll (root directory files)**

total 16

lrwxrwxrwx  1 root root    7 Dec 15 21:53 bin -> usr/bin **(binary / user command files**

dr-xr-xr-x  4 root root 4096 Dec 15 21:55 boot **(system Kernel / modules / boot loader)**

drwxr-xr-x 14 root root 2860 Dec 27 10:52 dev **(Device files like kbd, mouse, webcam)**

drwxr-xr-x 81 root root 8192 Dec 27 10:52 etc **(System Configuration / script files)**

drwxr-xr-x  3 root root   22 Dec 27 10:52 home **(users home directory/files like ec2-user)**

lrwxrwxrwx  1 root root    7 Dec 15 21:53 lib -> usr/lib **(system/application libraries)**

lrwxrwxrwx  1 root root    9 Dec 15 21:53 lib64 -> usr/lib64

drwxr-xr-x  2 root root    6 Dec 15 21:53 local

drwxr-xr-x  2 root root    6 Apr  9  2019 media **(external media devices like dvd/pendrive)**

drwxr-xr-x  2 root root    6 Apr  9  2019 mnt **(external - like network devices)**

drwxr-xr-x  4 root root   27 Dec 15 21:55 opt **(used by Major Software suite like oracle)**

dr-xr-xr-x 98 root root    0 Dec 27 10:52 proc **(Processor file system - CPU owned)**

dr-xr-x---  3 root root  103 Dec 27 10:52 root **(home directory for root user /super user)**

drwxr-xr-x 29 root root 1000 Dec 27 10:52 run

lrwxrwxrwx  1 root root    8 Dec 15 21:53 sbin -> usr/sbin **(super user binary/commands)**

**d**rwxr-xr-x  2 root root    6 Apr  9  2019 srv

dr-xr-xr-x 13 root root    0 Dec 27 10:52 sys

drwxrwxrwt  8 root root  172 Dec 27 11:04 tmp **(temporary directory for every user)**

drwxr-xr-x 13 root root  155 Dec 15 21:53 usr **(small application files )**

drwxr-xr-x 19 root root  269 Dec 27 10:52 var **(Various -Logs, FTP Home dir, crash dump)**

**3-Tier Model**

1. User / browser (Chrome, Firefox, IE)

2. Webserver (Frontend) (Eg: apache, IIS,ngnix, )

3. Backend (Database - eg:MySQL, oracle,postgresql, MSSQL, Mariadb, Aurora DB )

CMD:

    1  exit

    2  hostname

    3  uname

    4  whoami

    5  date

    6  cal

    7  cal 2023

    8  cal 8 1947

    9  clear

   10  bc

   11  echo " My Name is Anand "

   12  history

Uptime

Os-release

W

Help

Man

Pwd

Mkdir

Rmdir

Mv

Cp

Touch <filename>

Ls

Cd

Ps

Top

Kill

Bg

Fg

Scp

Rsync

Ping

Netstat

Wget

Curl

Nohup

**stat cmd in linux = time stamp of file**

Tar / zip

Tar - tape archive ( adding all files into single file)

Zip - compression (size reduction)

Tar

c - create the archive

t - list the archive

x - extract the files from archive

v - verbose / verify

f - file display

User

pwd,ls,cat,tail,head,touch,mkdir,cp,mv,wc,ln,locate,echo

give example

ChatGPT

Certainly! Here are examples for each of the mentioned commands:

pwd (Print Working Directory):

bash

Copy code

$ pwd

/home/user/documents

ls (List Directory Contents):

shell

Copy code

$ ls

file1.txt file2.txt folder1 folder2

cat (Concatenate and Display):

shell

Copy code

$ cat file1.txt

This is the content of file1.txt.

tail (Display the End of a File):

shell

Copy code

$ tail -n 5 file2.txt

This is the last line of file2.txt.

head (Display the Beginning of a File):

arduino

Copy code

$ head -n 3 file1.txt

This is the first line.

This is the second line.

This is the third line.

touch (Create Empty File or Update File Timestamp):

shell

Copy code

$ touch newfile.txt

mkdir (Make Directory):

shell

Copy code

$ mkdir new\_folder

cp (Copy Files or Directories):

shell

Copy code

$ cp file1.txt new\_folder/

mv (Move or Rename Files or Directories):

shell

Copy code

$ mv file2.txt folder1/

wc (Word Count):

shell

Copy code

$ wc file1.txt

3 12 67 file1.txt

ln (Create Symbolic or Hard Links):

shell

Copy code

$ ln -s file1.txt link\_to\_file1

locate (Find the Location of Files):

arduino

Copy code

$ locate file1.txt

/home/user/documents/file1.txt

echo (Print Text to the Terminal):

shell

Copy code

$ echo "Hello, World!"

Hello, World!

~ - Home Directory - /home/ec2-user

**Absolute Path:** /home/ec2-user/a\_reports/1/graph\_dec\_a1

~/a\_reports/1/graph\_dec\_a1

**Relative Path :**

From Home Directory : a\_reports/1/graph\_dec\_a1

~/a\_reports Dire 1/graph\_dec\_a1

~/a\_reports/1 graph\_dec\_a1

~ /

[ec2-user@ip-10-0-10-130 1]$ mv ~/\*a3\* ~/a\_reports/3 - Absolute Path

[ec2-user@ip-10-0-10-130 1]$ pwd

/home/ec2-user/a\_reports/1

[ec2-user@ip-10-0-10-130 1]$ mv ../../\*a3\* ../3 - Relative Path

**Pipes & Filters**

1. More -
2. Less
3. Head
4. Tail
5. Sort
6. **Grep**
7. Cut
8. Tr
9. Awk =

awk -F: '{print $1}' /etc/passwd

1. Sed
2. wc

**File or dir Permissions – Accessing**

* **Basic Permissions - 3 (read, write & execute) (r,w,x)**

**File types - 7 types**

* **Regular file**

**d=directory, l=link file, c=character device file, b=block device file, s=socket file, p=pipes**

* **File access permissions can be given to owner or user (who created the file/folder), group, others (rest all) - u,g,o**
* **How to assign permissions?**
* **Owner group other**
* **rwx rwx rwx**
* **421 421 421 7 7 7 777**
* **rw- r-x r–**
* **420 401 400 6 5 4 654**
* **rwx rw- r– 7 6 4 764**
* **Rw- rw- r– 664**
* **755 rwx r-x r-x**
* **644 rw- r–  r–**

**chmod (Only owner or root/su can change permissions)**

**$ chmod <permission> <filename>**

[ec2-user@ip-10-0-10-51 ~]$ **ll**

total 4

-rw-rw-r-- 1 ec2-user ec2-user 6 Dec 28 09:24 test

[ec2-user@ip-10-0-10-51 ~]$ **chmod 755 test**

**Umask** = total permissions - default permission

= 777-775 = 002

022 = 777-**755**

700 = umask = 077

[ec2-user@ip-10-0-10-51 ~]$ **umask 022**

[ec2-user@ip-10-0-10-51 ~]$ **umask**

0022

[ec2-user@ip-10-0-10-51 ~]$ **mkdir anand2**

[ec2-user@ip-10-0-10-51 ~]$ **ll**

total 4

drwxrw---- 2 ec2-user ec2-user 6 Dec 28 09:31 anand

drwxrwxr-x 2 ec2-user ec2-user 6 Dec 28 09:36 anand1

drwxr-xr-x 2 ec2-user ec2-user 6 Dec 28 09:41 anand2 (directory perm set to 755

-r-------- 1 ec2-user ec2-user 6 Dec 28 09:24 test

**Vi / ViM Editor - Vi - Visual Interactive / Vi Improved**

Basic Editor

There are 3 Modes of Operation

**Default 1. Command Mode**

2. Editor Mode

3. Global Command Mode

Editor mode to command mode - ^  (escape)

Editor mode to Global command mode ^: (escape +colon)

**1. Command Mode (commands)**

Append - a

Insert - i

Append end of line - A

Delete character - x

Undo - u

Delete line - dd  or ndd

Copy line - yy or nyy ( n for no of lines)

Place / paste - p

Insert blank line O

change word - cw

Bottom of document / ground - G

Top of document - 1G

**3. Global Command Mode ^:(Commands)**

Save/write - w

Save exit - wq or x

Exit without save - q!

Find - /<string> n- next

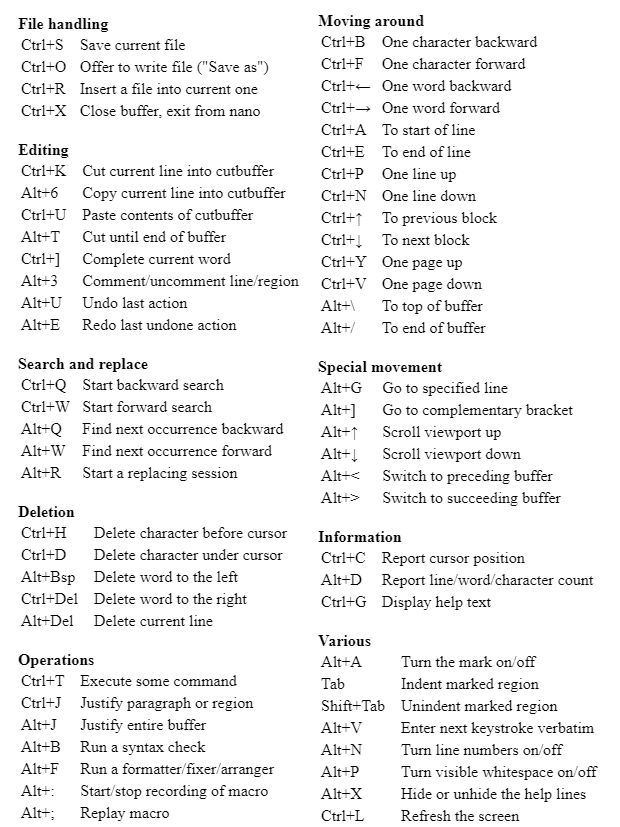
Find / replace - %s/<find>/<replace>/g

See number bar - se nu

See no number bar se nonu



**NANO**

****

**Crontab - Task Scheduler**

**Crontab - 3 Options**

**-l = list the contents of Cron**

**-e = Create Crontab Entry**

**-r = remove crontab entry**

**How to create crontab entry**

**Every line is a scheduled task**

**Every line has 6 column**

**Column Specification**

**1 2 3 4 5 6**

**MM HH DOM MOY DOW**

**Minutes Hours Day of MonthMonth of Year Day of Week task / command**

**(0-59) (00-23) (1-31) (1-12) (0-6)(0-sun)**

**00 09 \* \* \* mkdir abcd**

**30 08 1 \* \* sh xyz**

**30 21 \* \* 5 sh backup**

System cmd :

**df - disk free $ df -h**

**$ du - Disk Usage $ du -s**

Dev/null

Find

**find / -size +50M -exec ls -l {} \; 2>/dev/null**

**find / -name passwd -exec ls -l {}  \; 2>/dev/null**

**vi sudo**

**chmod**

**usermod**

**bash\_profile**

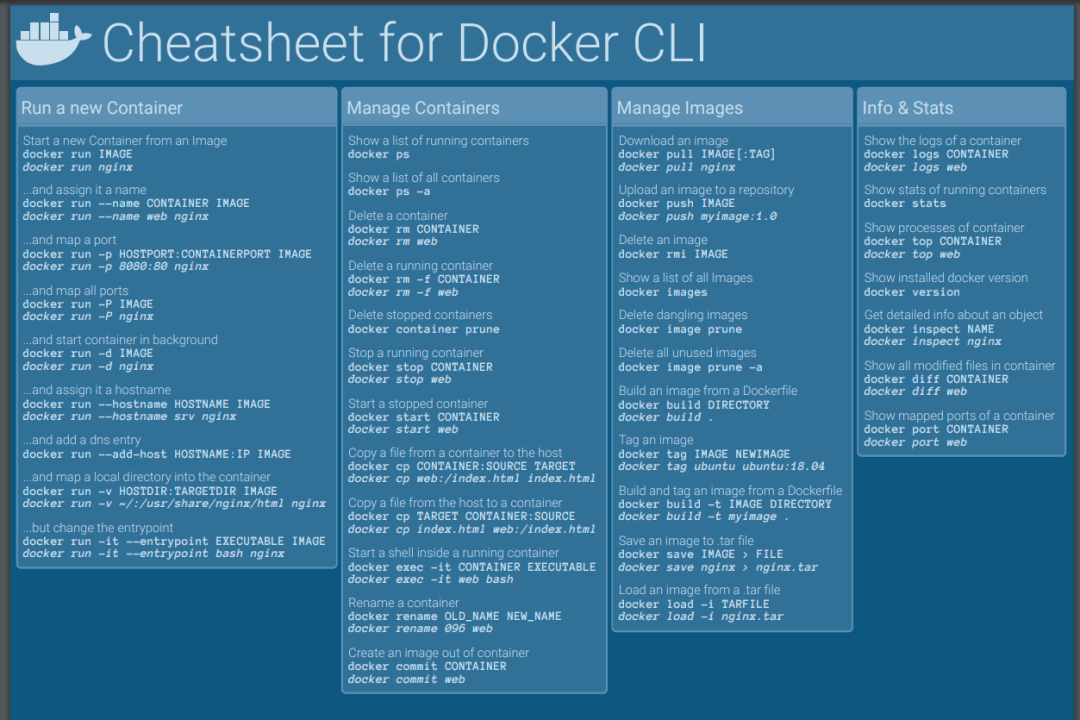
**bashrc = every terminal opened**

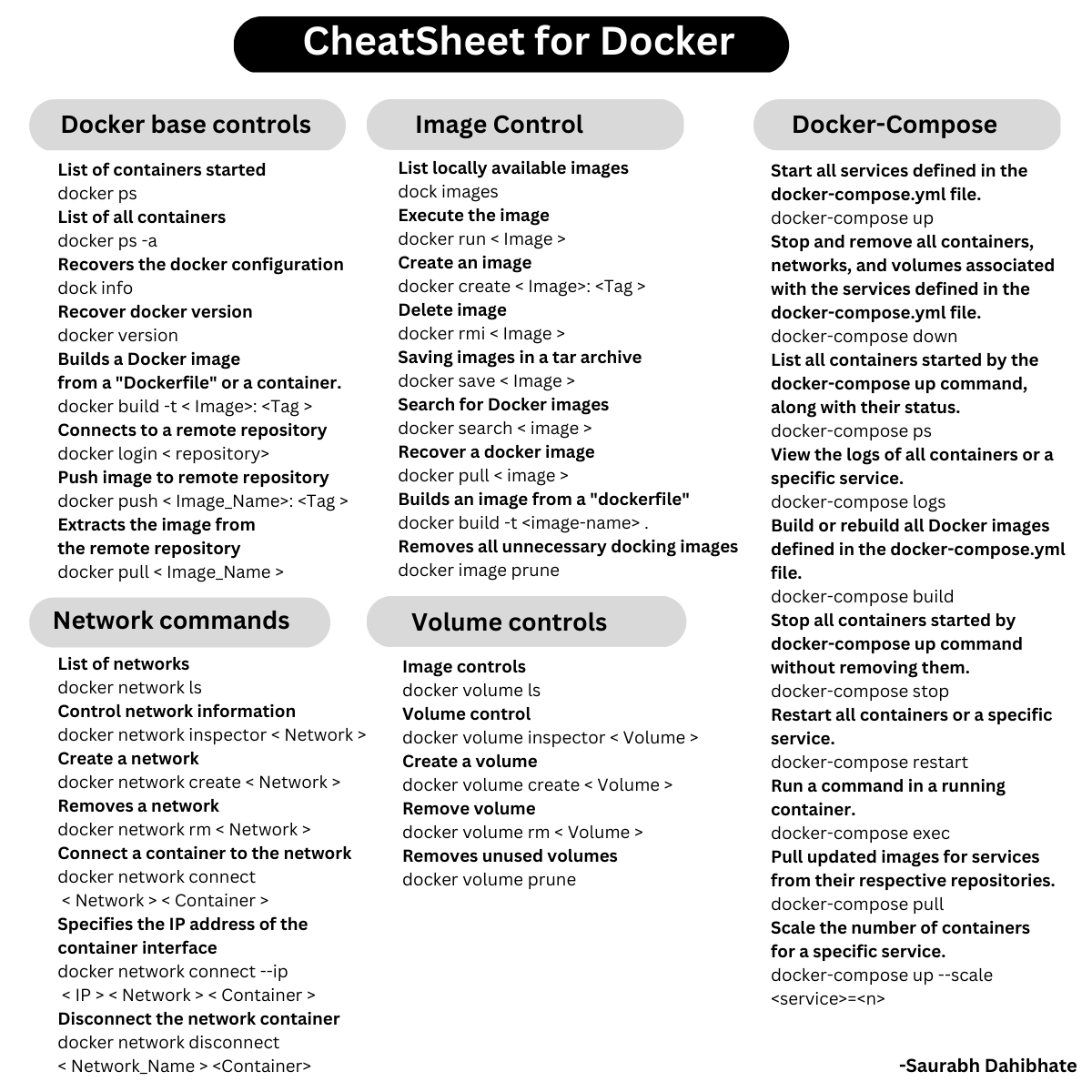
**export**

**source**

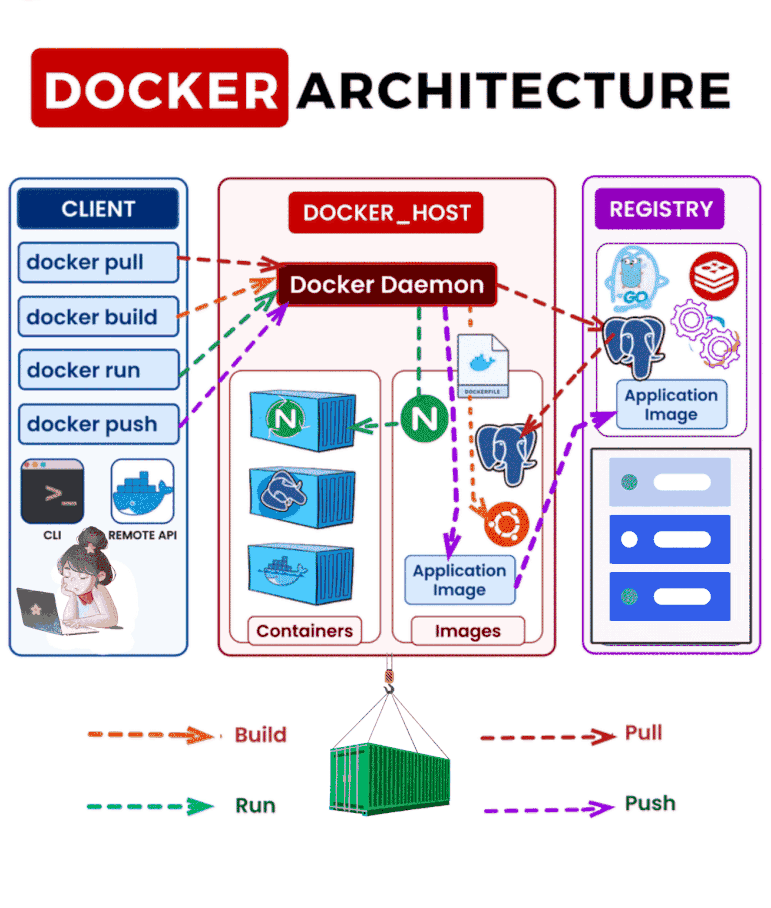
**History : !! , !<no> ,**

**Week 2 :**





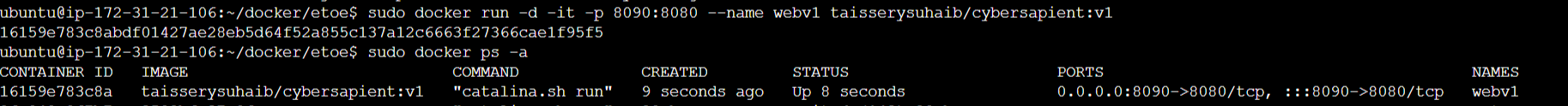
* + - **Docker architecture**



**Docker is containerized platform , lightweight .**

**Sudo apt install docker.io**

**Docker run :**

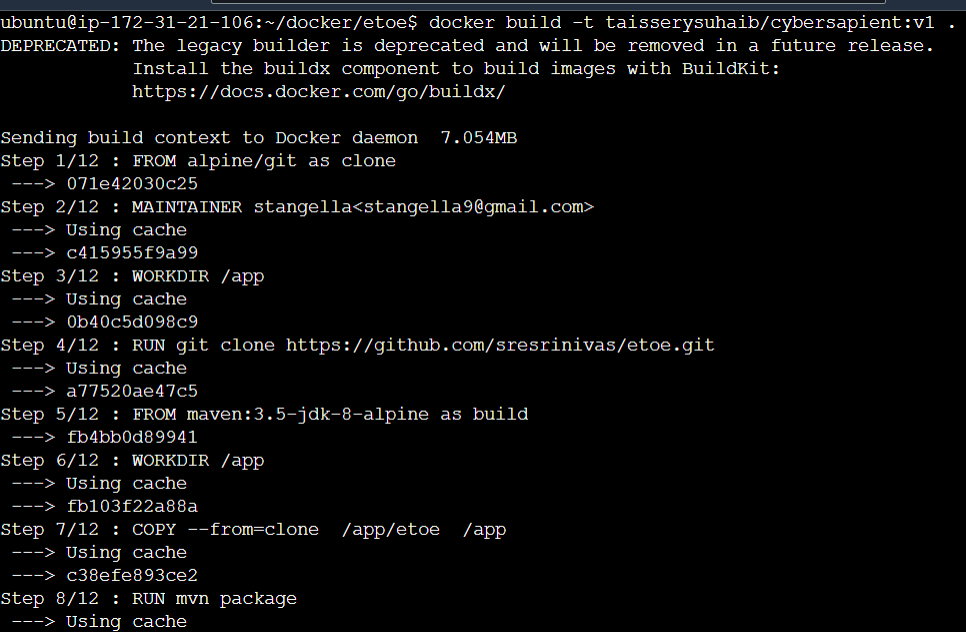
****

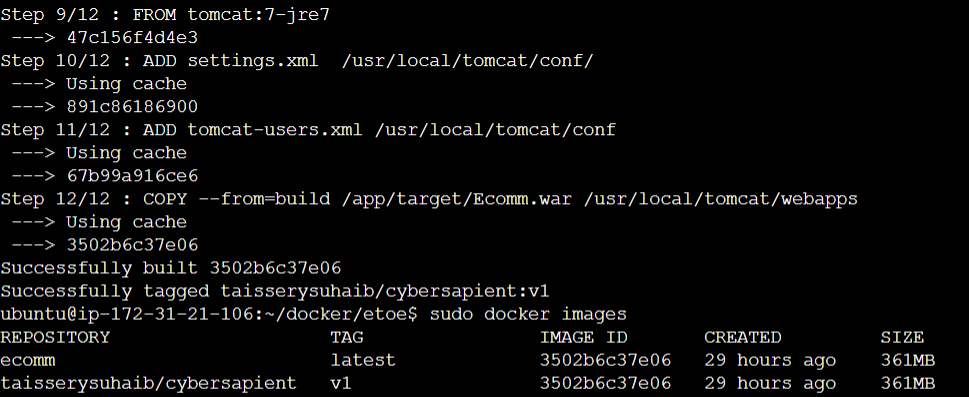
**Docker commit :**

**creates images from the container**

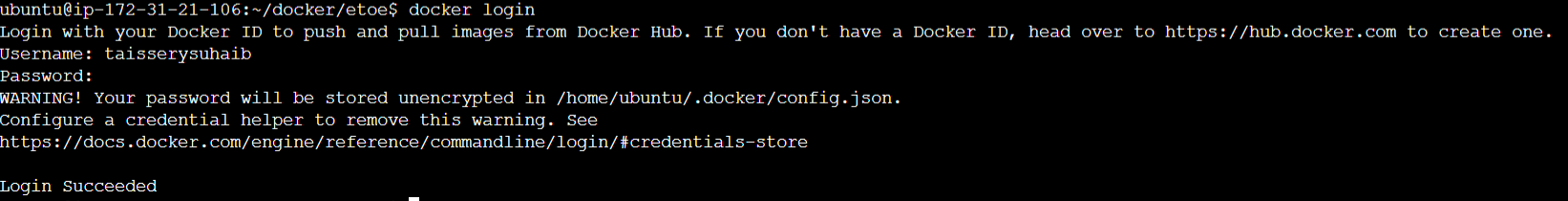
**Syntax : docker commit <cont id> <imagename:tag>**

**Docker tag with build :**

****

****

**Docker login :**

****

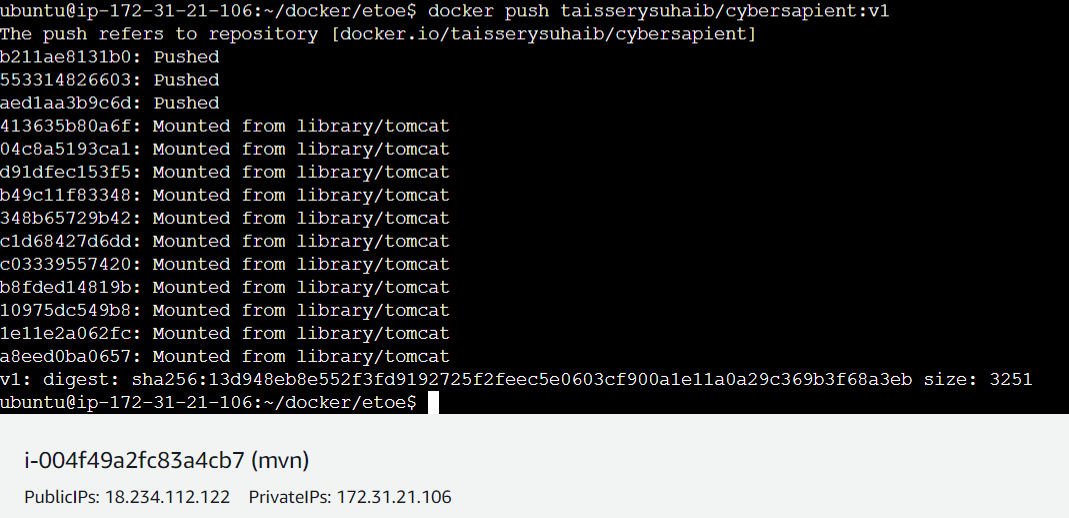
**Docker push :**

**docker push taisserysuhaib/cybersapient:v1**

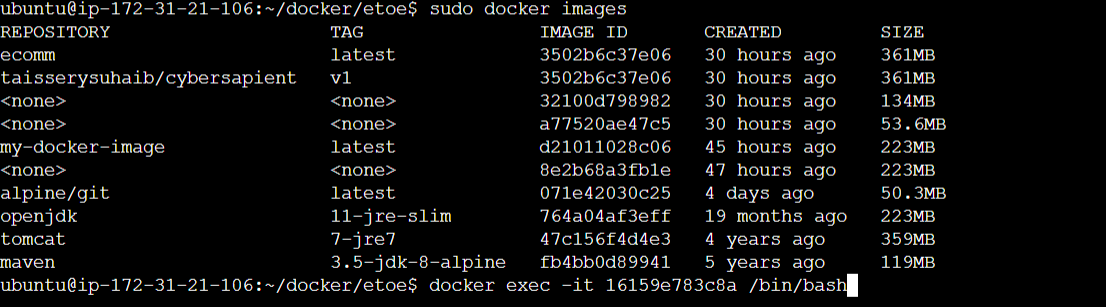
**docker username : taisserysuhaib**

**repo name : cybersapient**

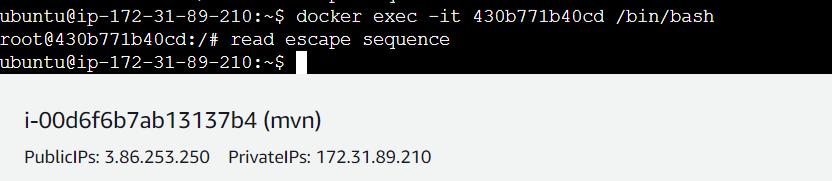
**tag : v1**

****

**Docker exec :**

****

**Exit the container without actually exiting it .**

**Ctrl + p , ctrl + Q :  
  
**

**Docker Inspect :  
  
to debug or read about container and images**

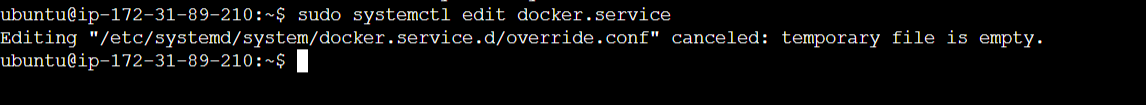
**Syntax : docker inspect <cnt id or image id>**

**Docker pull :**

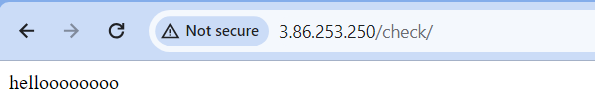
**Syntax : docker pull <username/repo:tag> or <image name>**

**Docker stats :  
I will show the container resources**

**Syntax : docker stats <cnt id>  
**

**Edit docker configure :  
**

**Install appache2 and under host a static web page**

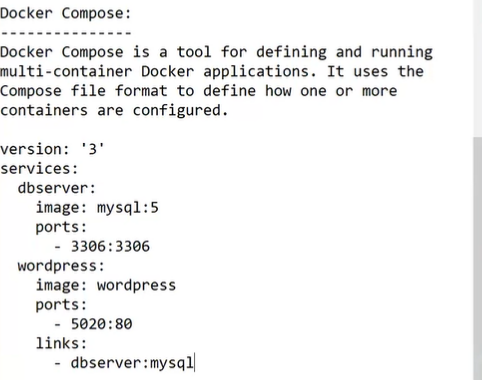
****

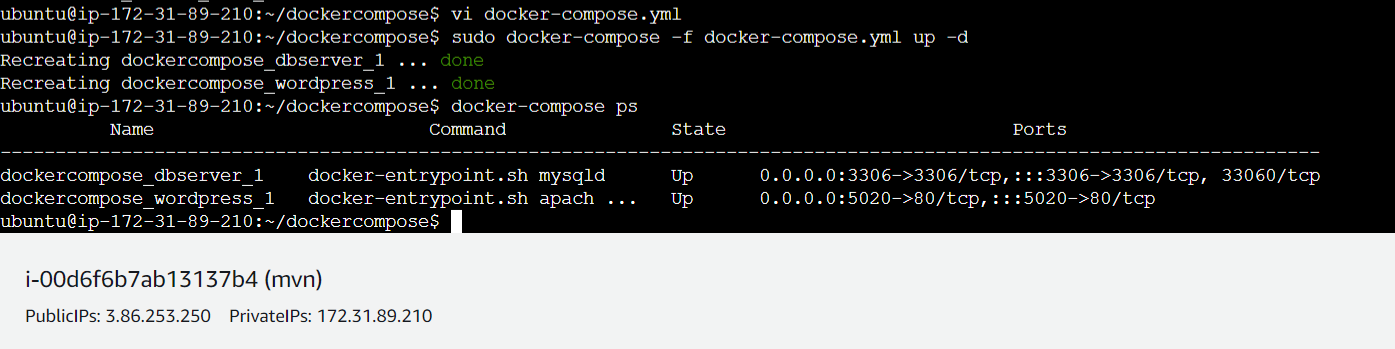
**Run a static webpage base a nginx using docker :**

****

**Output :  
**

* **Docker compose :** 
  + **To create or run multiple container we write in a single code file**
* **Go to cli mode :**
  + **Vi docker-compose.yml**
  + **Paste the below code**

****

****

* **Install docker compose :**
  + **Apt install docker-compose**
* **Run cmd :**
  + **Docker-compose up -d**
  + **Or if the file has different name the ti up the compose** 
    - **Cmd : docker-compose -f <nameofthe docker compose file> up -d**

**Note : to remove all the running container at once**

* **Docker rm -f $(docker ps -aq)**

**Agenda : attaching volume to a container :**

**2 types :**

**1) simple docker volumes**

**2) docker volume containers : shared storage**

* ****
* **This is the path in which the we can check the info of log for application installation .**

Centos = image in which container has to be created

* **Create or attach a volume to a container :**
  + **Docker run –name <name> -it -v /data centos**

-v /data :

Attaching a volume to an container

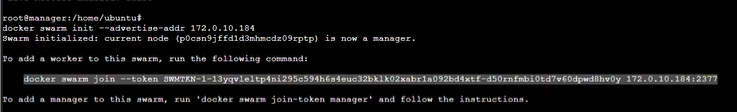
* **Enter inside the container :** 
  + **And run the cmd by: ls** 
    - **Check for the folder data which is volume**
* **Now copy the path of the volume of the container :**
  + **Docker inspect <container id / name>**
    - **Search for volume or mounts :**
    - **Copy the source path**
  + **Run the cmd :**
    - **Ls -lrt <path>**
* **To come out of the container with out exiting(stoping) :**
  + **Ctrl +P+Q**

* **Create a container using a docker volume container**
  + **Create a container :**
    - * **Docker run –name mycon -it –volumes from < container name of reference >**
* **Networking :**
  + **Host type - uses the public ip of ec2**
  + **Bridge type -- ip to ip ( default bridge given in docker container )**
  + **Null -- if the container should not communicate to other**
  + **Overlay – used in docker swarn**
* **We can specify the kind of or type of networking should be given to the container while creating** 
  + **Docker run -it –name \*\*\*\*\*\* --net host ubuntu**
* **To know the logs of the container :**
  + **Docker log <container name or id >**
* **Docker swarn :**
* **Create a 3 ec2 or any :**
* **One as master and others as worker**

**Note : so to identify the which ec2 we r considering it as master or worker > go to its cli**

* + **VI etc/hostname**
  + **Init 6**
* **Docker swarm init**
* **In master ec2 : docker swarm init - -advertise-addr <private ip of master> { to make it a manager of swarm }**

****

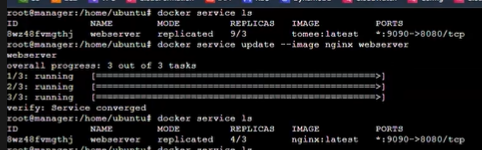
****

* **Copy the cmd and paste in the worker ec2**
* **Go to master ec2 :**
  + **Docker node ls ( to check if the worker node is added )**
  + **From master : docker service ps <container name>**
* **Create a a container :**
  + **Docker service create –name <\*\*\*> -p 9090:8080 --replicas 1 <image name>**
* **To run more of it** 
  + **Docker service scale <containername> = number of containers**
* **Go to worker :**
  + **Docker ps -a**
* **To delete all the master and worker then :**
  + **Docker swarn leave –force**
* **To remove node or delete :**
  + **Can be done in manager/master :**
    - **Docker node - - availability drain worker -1**

**Note :**

**To get back the node :**

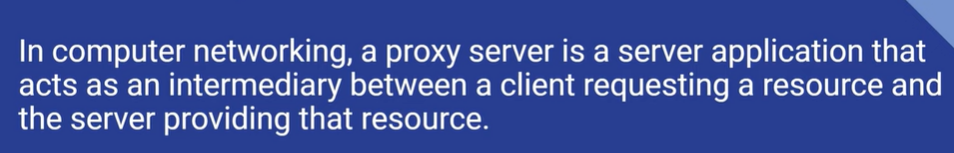
**> = docker node - - availability active worker -1**

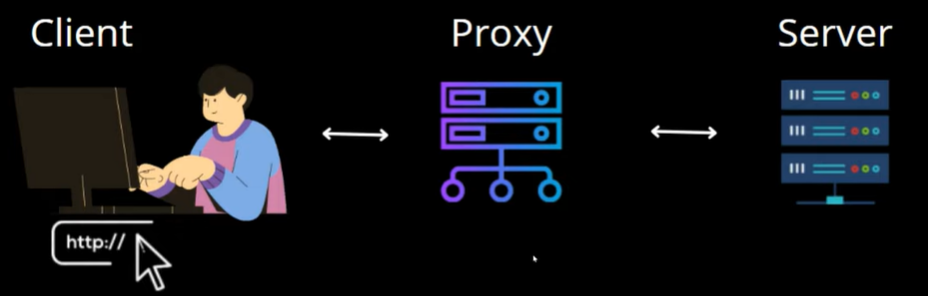
* + **Can be done from worker :**
    - **Docker swarn leave**
* **Rolling updates :**
  + **Updating the existing container image with latest images**
  + ****
  + **Docker service ls :**
  + **Docker service update - - image nginx webserver/id**

Container id or name

New image

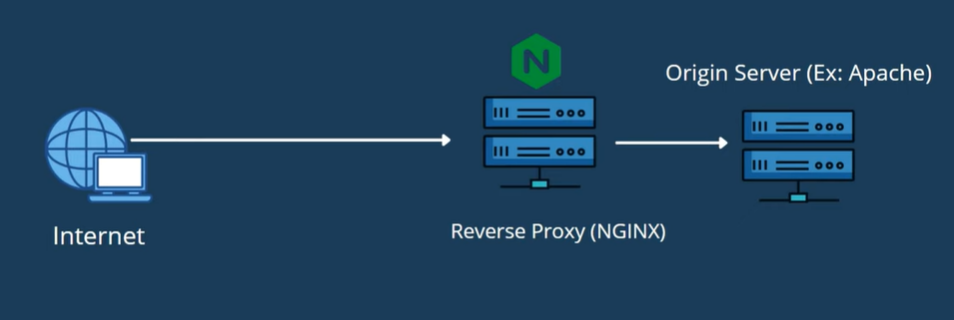
* **To add a another worker to the manager :**
  + **The manager should have the token , so to get the token** 
    - **Docker swarn join–token manger**
* **Making docker worker as manger : promote** 
  + **Docker node ls**
  + **Docker node promote worker-1**
* **Reverse proxy method : using nginx :**

****

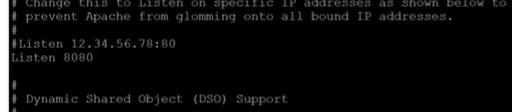
****

**Forward proxy : on client side**

**Reverse proxt : on server side**

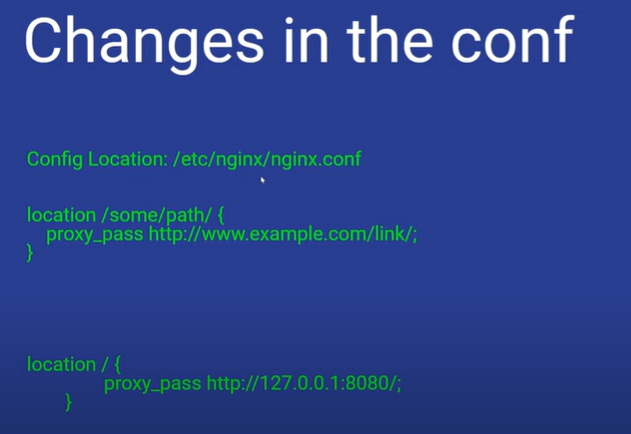
****

* **Open ec2** 
  + **Install appache httpd**
    - **To change the port of appache : cd /etc/httpd/conf > cat httpd.conf**

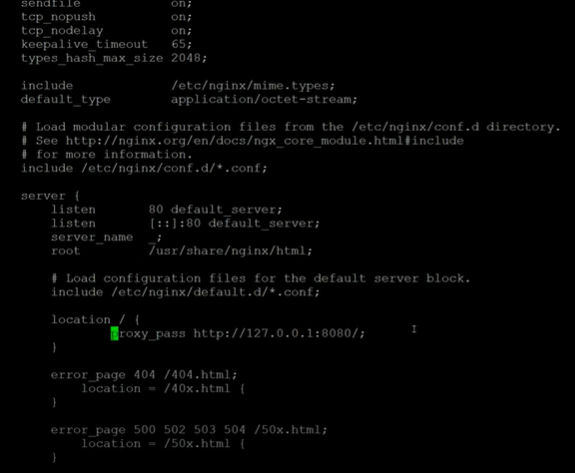
****

* + **Install nginx**

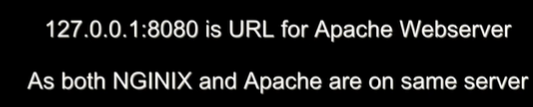
**For reverse proxy :**

****

* **Cd /etc/nginx/**
* **Vi nginx.conf**

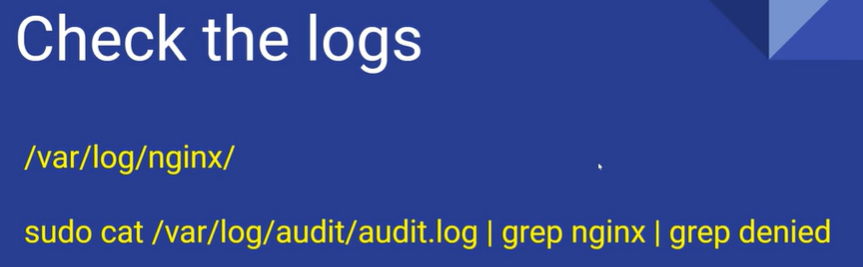
****

**All the content under location block**

****

* **Systemctl stop nginx.service**
* **Systemctl start nginx.service**

**Refresh the browser page : of error**

****

* **Cd /var/log/nginx**
* **Cat error.log \**

****

****

**Versal = webpage / frontend**

**Render = backend**

* **Create a repo in github**
* **Push the code**
* **Npm run build** 
  + **It will create a folder called build > under this there is a static code**
* **Npm start > open browser: localhost:3000**
* **Open render.com : render for backend**
* **Select webservice : connect with github** 
  + **Coneect the repo**
  + **Name**
  + **Branch :**
  + **Root directory : folder or location of backend**
  + **Runtime : node**
  + **Build cmd : npm install**
  + **Nom start**
  + **Open advance setting :**
* **Versel :**
  + **Connect with github**
  + **Create a new project**
  + **Select the project repo > import**
  + **Name :**
  + **Root directoy : ./**
* **Multi stage docker file :**

ubuntu@ip-172-31-89-210:~/etoe$ cat Dockerfile

FROM alpine/git as clone

MAINTAINER stangella<stangella9@gmail.com>

WORKDIR /app

RUN git clone https://github.com/sresrinivas/etoe.git

# stage-two

FROM maven:3.5-jdk-8-alpine as build

WORKDIR /app

COPY --from=clone /app/EcommWeb /app

RUN mvn package

# stage-third

FROM tomcat:7-jre7

ADD settings.xml /usr/local/tomcat/conf/

ADD tomcat-users.xml /usr/local/tomcat/conf

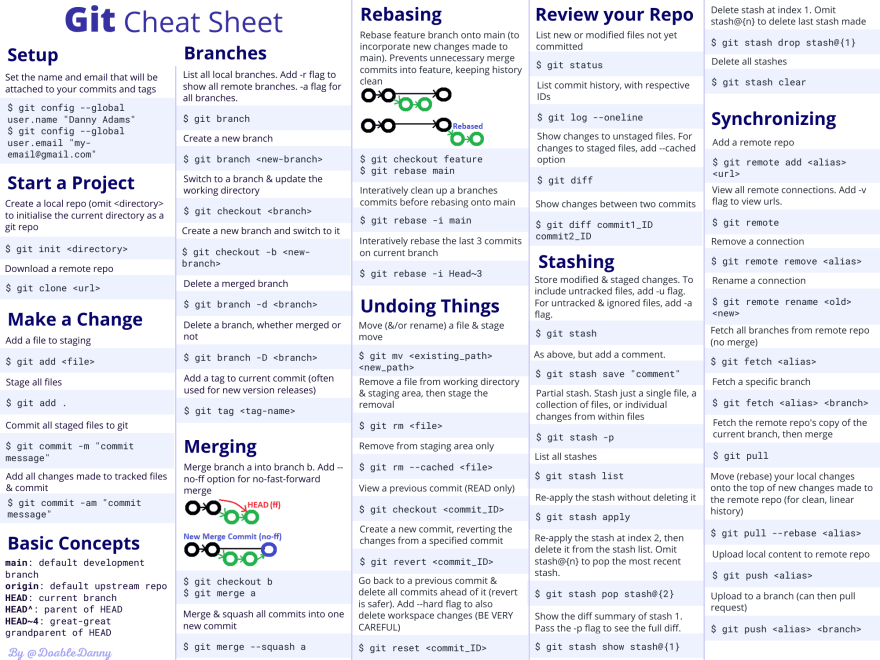
COPY --from=build /app/target/Ecomm.war /usr/local/tomcat/webapps

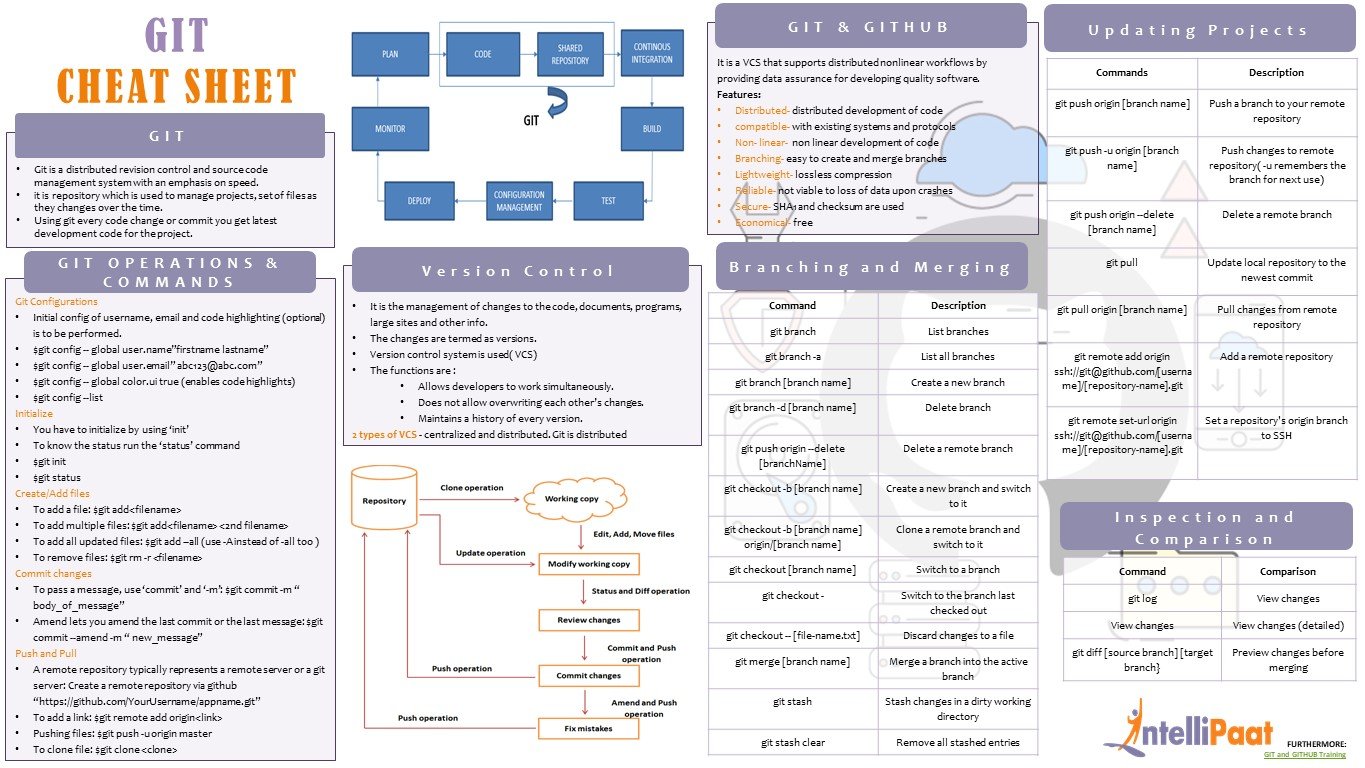
* **Distroless image :**

Distroless docker images are just way to lighten the application image by removing unnecessary binaries, artifacts and functionalities. Since unnecessary components are removed, its size will be less and attack surface will be smaller. And there is a chance for reduces vulnerabilities as well.

**Week 3: CI/CD:**

* **Git :**





**Week 3**

**Learn Github Actions, create CICD, use caching, create secrets, send mail when build fails, create a repo with dev, stage, prod branches, create cicd for each branches.**

**git commands, create branch, pull , add, commit, fetch, diff btw merge and stash, git branching strategies,**

**learn sonarcloud, integrate with github actions.**

**Learn Jenkins, create CICD, use caching, use secrets ,send mail when build fails, create a repo with dev, stage, prod branches, create cicd for each branches.**

**"create github actions pipeline to deploy nodejs application in bare server.**

**create github actions pipeline to deploy python application in bare server**

**create github actions pipelinet o deploy react application in bare server"**

**integrate sonarcloud with jenkins, learn dependency check and integrate it with jenkins**

**"create jenkins pipeline to deploy nodejs application in ec2**

**create jenkins pipeline to deploy python application in ec2**

**create jenkins pipelinet o deploy react application in ec2"**

**use docker as a agent in jenkins worker nodes**

**https://owasp.org/www-community/Free\_for\_Open\_Source\_Application\_Security\_Tools OWASP tools.**

**version manager for node application and python.**

**pm2 process manager**

**Build fail should be notified in slack channel in github actions and jenkins**

* **Git :**

Git init

Git status

Git add <file name>

Git commit -m “any”

Git remote add origin <github-URL> ( origin is an alias name not a fixed name can give any )

Git remote set origin <github\_url>  
Git remote remove origin

Branch creation :

* Git branch <name>
* Git checkout -b <name>
* Git branch ( will show on which branch we r at present )

Git push :

* Git push origin main ( or ) --force

Git stash :

If we are working in some branch named <mybranch> and you want to go to another branch by keeping the changes there it self

* Git stash -u <will stash the work in present branch > we can checkout to another branch
* Git stash list <will show all the stash list >
* Git stash pop <will unstash > (before unstash go to the particular branch)

Delete branch on local :

* git branch -D f2 ( you should not be working under the branch which has to be deleted , check out to another branch )

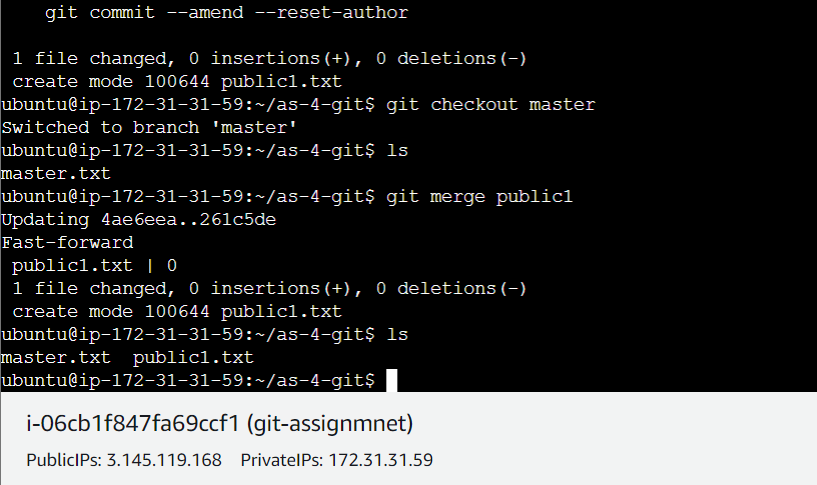
on remote (github)

* git push origin --delete f2
* git push origin –all ( will even push branch to remote )

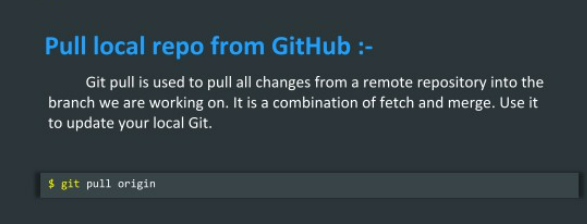
git merge :

note : we need to be in base branch before merging for suppose if I have to merge public1 with main

* git merge public

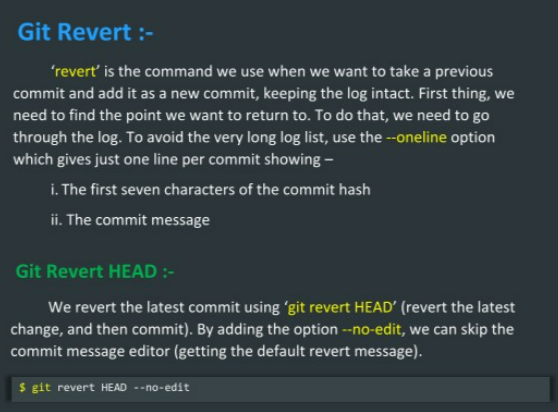


* git log
* git log –online



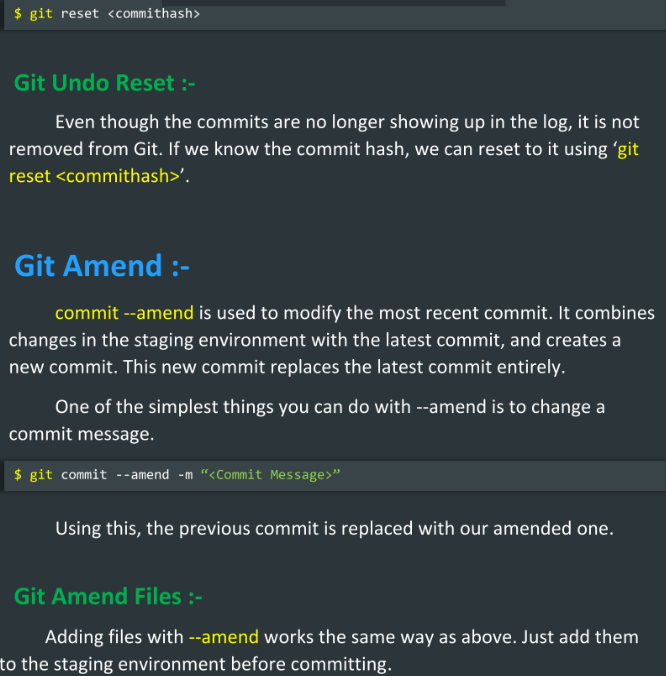
Git clone :

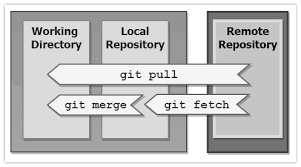
* git clone <url >
* git clone <url > <folder name>
* git revert :



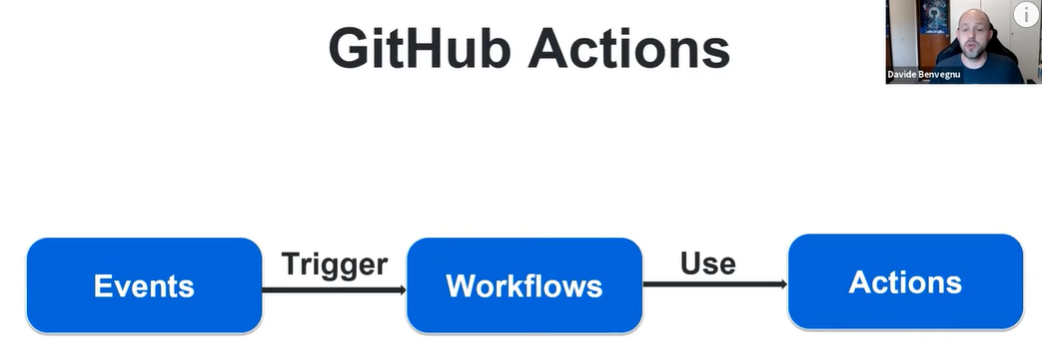
* git log –online ( copy the id )
* git revert <log id >

git reset :



Git pull vs fetch   
  


* **github actions :**

****