**Cheat sheet for DOCKER**

* Docker is a tool that makes it easy to run and manage applications in isolated environments called **containers**.
* It automates the deployment of applications within lightweight, portable containers that include all the necessary dependencies, libraries, and configuration files, allowing applications to run consistently across different computing environments.
* To get different images we use Docker hub, to install Docker sudo apt-get Docker.io
* Container works as application
* The main difference between image and container is image is read only but container can be edited because of free space inside of it.
* The data of images and containers are in /var/lib/Docker & /var/lib/Docker/container
* The limitation of Docker is, it is only used to create container but not to manage.
* Docker can share containers from one to another but Docker can’t auto scale for that we use K8s
* TO CREATE .TAR FILE AND SHARE - There are three ways to share/transfer the image, we can use .tar-file, Docker hub OR ECR (elastic container registry)

1. From scp -i we can get the .tar file from instance to desktop
2. From Docker hub, we have to create a/c in Docker hub then to log in from Docker of instance and give tag as per our account name and repository name
3. From ECR, first change the configuration as our AWS account and give the access key and private key and region then we have to create repository in ECR, then click on view push command, and follow the commands step by step.

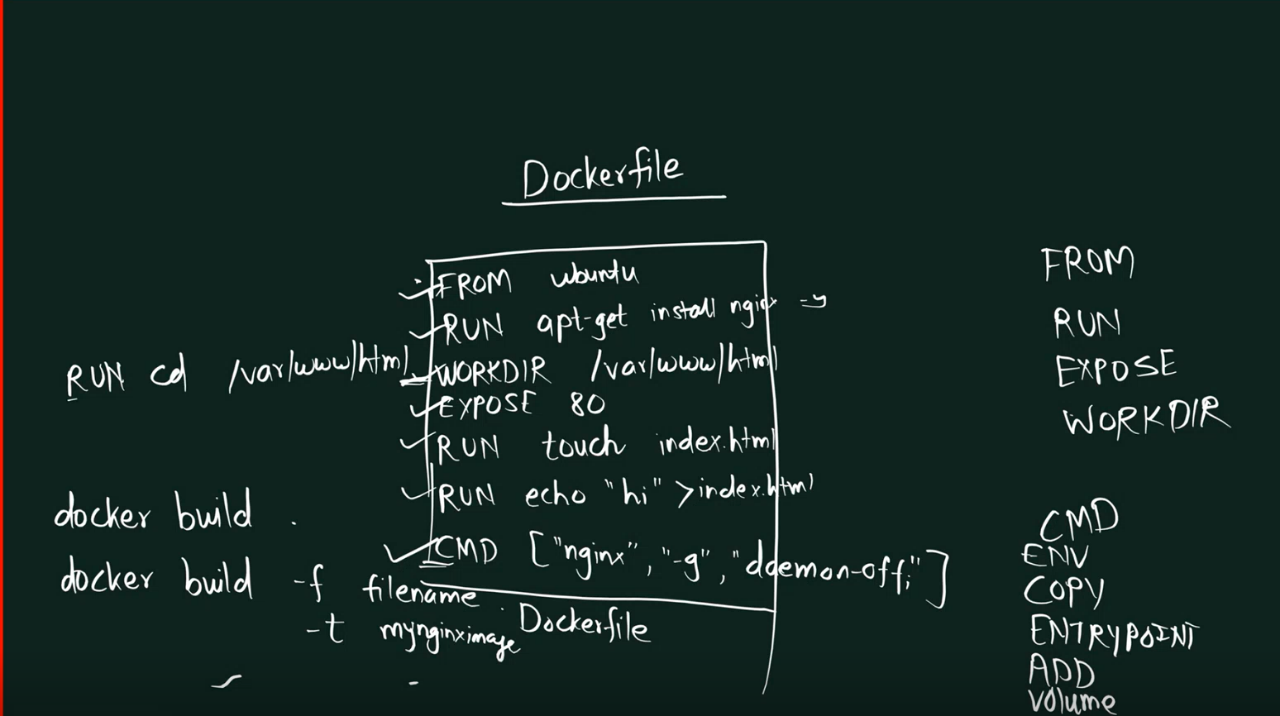
* To install aws-cli in instance

1. sudo apt install curl unzip
2. curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"
3. awscliv2.zip
4. sudo ./aws/install
5. aws --version

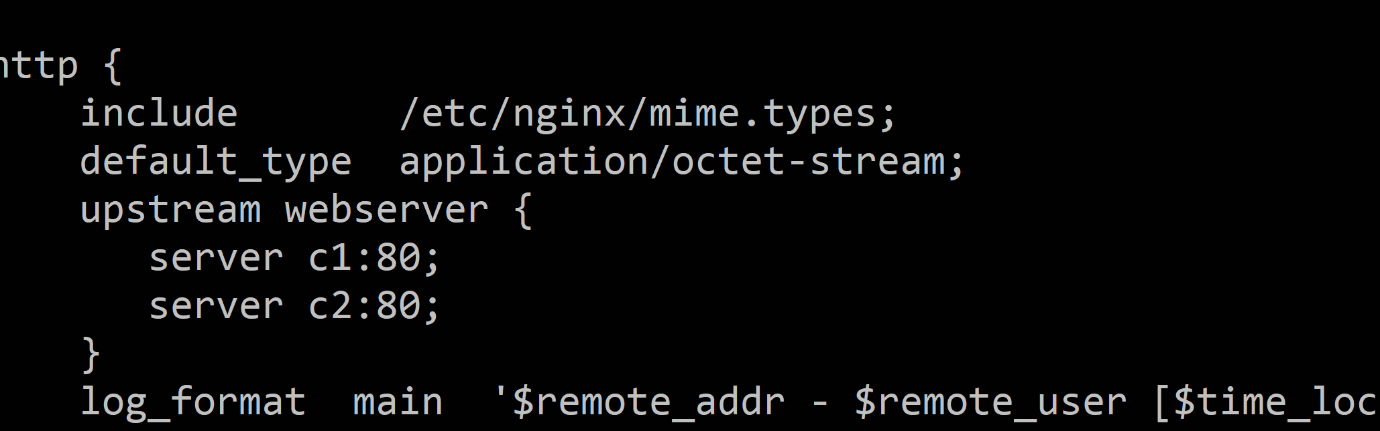
* DOCKER VOLUME – we can bind volume to number of containers, there are Three types of volumes, bind volume, named volume & anonymous volume

1. Bind volume – In this type we have to create directory to be used as volume and to bind it we have to give the full path of directory which will be used as volume
2. Named volume – In this type we just use the command, docker volume create volume-name to create volume then to bind we just use the name of volume and don’t have to give the path
3. Anonymous volume – In this type we don’t give any path or volume name the server will create a volume anonymously.

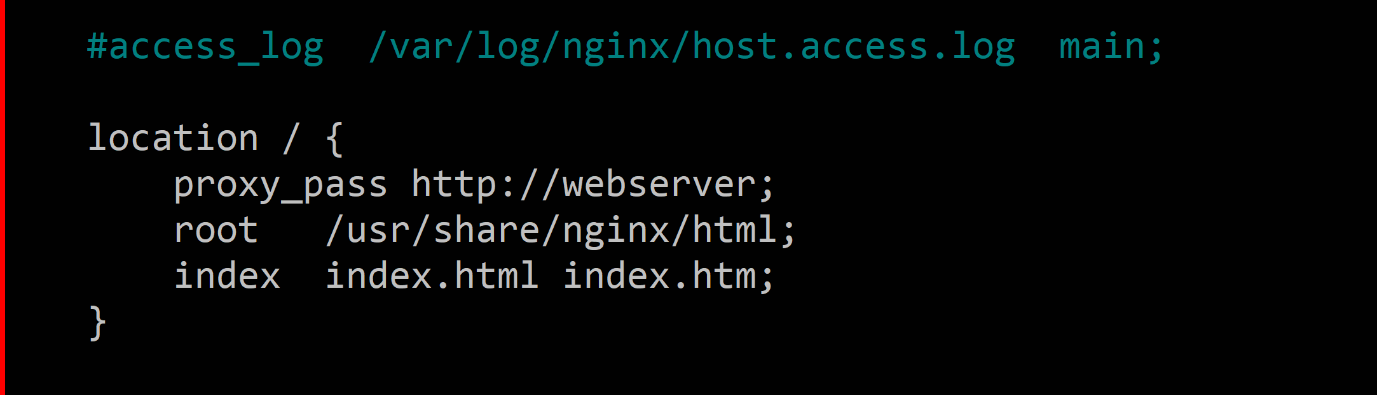
* DOCKERFILE – To create our own custom image



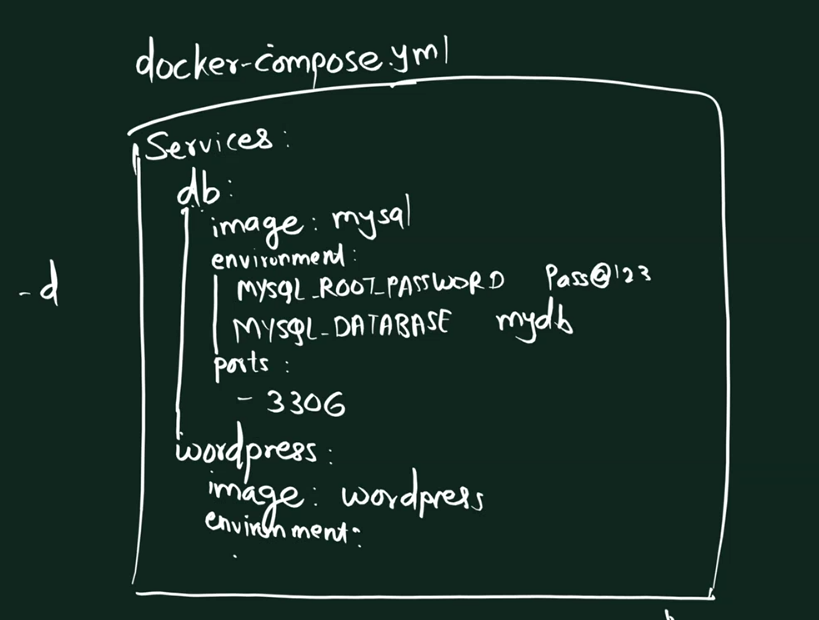
* There are lots of keyword – FROM (To pull base image rom docker hub) RUN (To run any command) COPY (To copy any files or directories from host to image) WORKDIR (working directory) EXPOSE (To assign the port) CMD (To specify the instruction when the image becomes the container) (-g To start service in background) ENV (To specify the environment variables) ENTRYPOINT (To specify the command to be run after container creation but allows arguments to be pass) ADD (copies files/directories from host to image also copies .zip & .tar files from link and also export it) VOLUME (To create volume) ARG (To define attribute to be passed) MAINTAINER (To specify the name and email of auther/user) LEBAL (To add metadata[The data od data is metadata]) USER (To set user) HEALTHCHECK (To specify path to health check) SHELL (To specify shell to be used to run command [/bash /ksh /sh]) STOPSIGNAL (specify the signal to container when to stop gracefully) ONBUILD (specifies the instruction to be use when we use this as a base image for another image)
* We can also create docker file using shell-script by copying the shell-script file inside docker file then build the docker file create container then we will get the content which is written in .sh file
* NETWORK – there are three types of networks in docker
  + Bridge network – default, A bridge network in Docker is a virtual network that allows containers running on the same Docker host to communicate with each other
  + Host network – insecure, A host volume is a file or directory on the host machine that is mounted into a container
  + Overlay network – between two different instances, The overlay network driver creates a distributed network among multiple Docker daemon hosts
* It is used to build & restrict connection between two containers
* NGINX PROXY –
* We can use nginx proxy as a load balancer in which we connect two containers with one proxy container in same network then we have to make changes in proxy containers configuration files.
* /etc/nginx/nginx.conf



* /etc/nginx/ conf.d/default.conf



* There is another type of nginx proxy HAproxy (highly available)
* DOCKER COMPOSE – It is tool for defining and running multi-container application, the main difference between docker-file and docker-compose is docker-compose can create multiple images with running container
* Docker compose file is written in YMAL language, in which line/word spacing is very important.



* Containerization tools – podman, container-d, LXC, Elastic Container Service (AWS), ACS(Azure)

**Commands**

* Docker pull image-name – To download image
* Docker images/ image ls – To list the image which are downloaded
* Docker history image-name – gives the history of all the layers of image
* Docker rmi/image rm image-name – To delete image
* Docker rmi/image rm image-name --f – To delete images forcefully
* Docker run image-name – To create container
* Docker run -d/ --detach image-name – To continue the process in background of creating the container
* Docker run -d -p 80:80 --name container-name image-name – To create container with connection with port 80 of instance and name.
* Docker run -p3306:3306 --name container-name -e MYSQL\_ROOT\_PASSWORD=sumit2606 -d mysql – To create mysql container
* Docker ps – To get details of running container
* Docker ps -a – To get details of all container
* Docker inspect container-ID – To get detailed info about container
* Docker start/stop – To start and stop the container
* Docker Stop $(Docker ps -aq) – To stop all container (this works in loop)
* Docker exec -it container-id /bin/bash – To get inside the container
* Docker rm container-name – To delete container
* Docker rm $(Docker ps -aq) – To delete all stopped container
* Docker rm C1 C2 – To delete multiple container
* Docker container rename old new – To rename the container
* Docker commit container-id name – To create image from container
* Docker save image-id > myfile.tar – To create .tar file of image
* Docker export container-id > mycon.tar – To create .tar file of container
* Docker import .tar-file new-image-name – To create image from .tar file
* Docker login – To login into Docker hub
* Docker tag old-name new-name – To change the name of image
* Docker tag old-name new-name:v1 – To give tag to image
* Docker push image-name – To push images in repository
* Docker run -d -p 80:80 --name container-name -v /home/ubuntu/mydata:/usr/share/nginx/html nginx – To bind the volume in nginx
* Docker run -d -p 3306:3306 --name container-name -e MYSQL\_ROOT\_PASSWORD=sumit2606 -v /home/ubuntu/mydata:/var/lib/mysql mysql – To mount the bind volume in mysql
* Docker logs container-name – To check errors of container
* Docker volume create volume-name – To create named volume
* Docker volume rm volume-name – To remove volume
* Docker volume prune – To remove unused(unmapped) volumes
* Docker image prune – To remove unused(unmapped) image
* Docker build . – To create image from Dockerfile (. Stands for current directory)
* Docker build -t image-name -f file-name . – To create image from custom docker-file and to give the name for image
* Docker network create network-name – to create network
* Docker network ls – to check networks
* Docker run -d -p 80:80 --name container-name --network N/T name image-name – To create container with network
* Docker network inspect N/T name – to check the details
* Docker network rm N/T name – to remove network
* Docker network connect N/T name container-name – to connect the network after creation of container
* Docker-compose up -d – To run the compose file with default name
* Docker-compose down – to stop and remove the containers & network