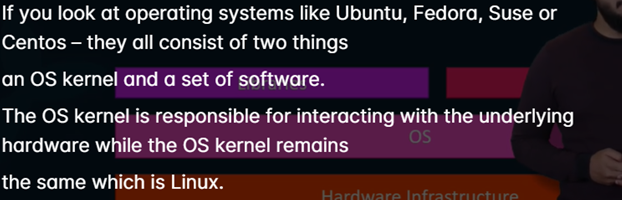
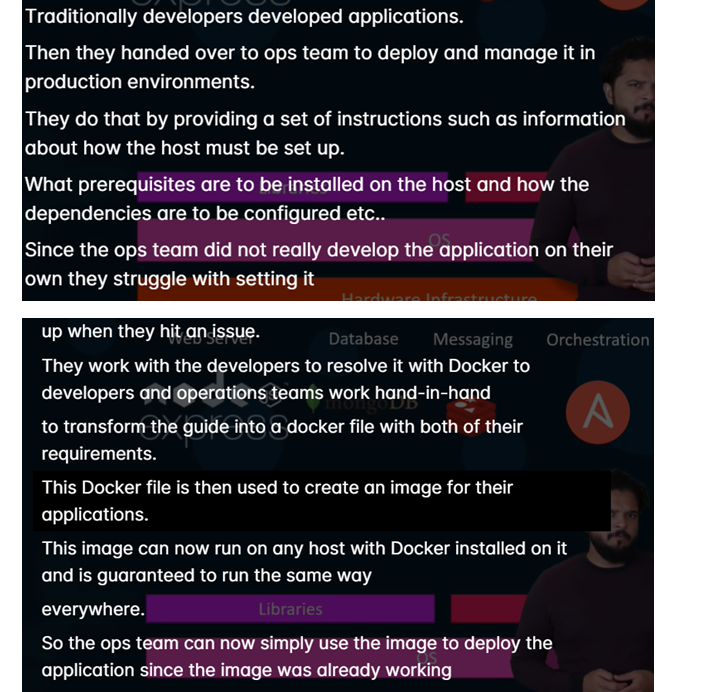
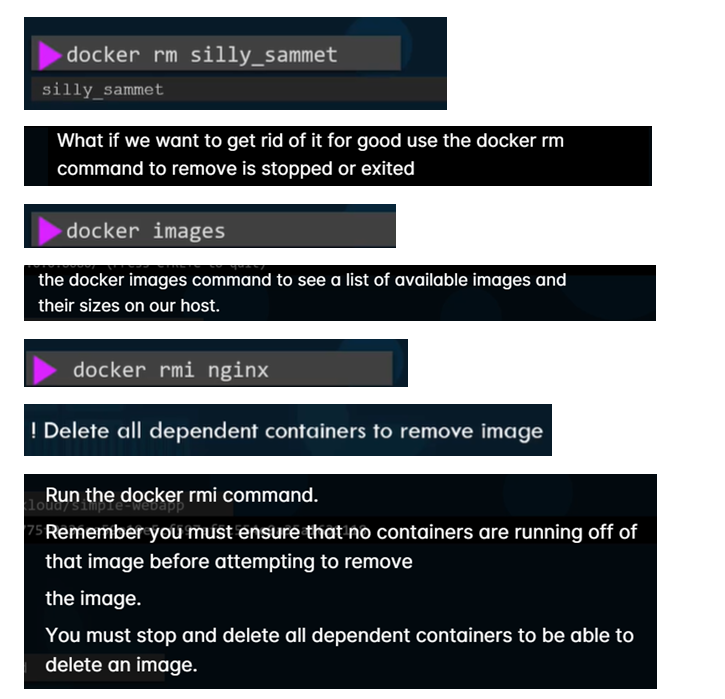
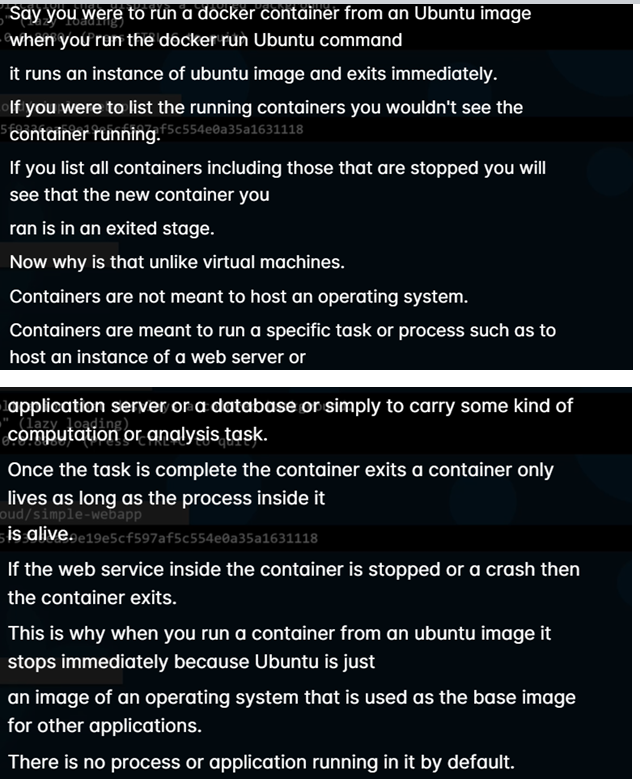
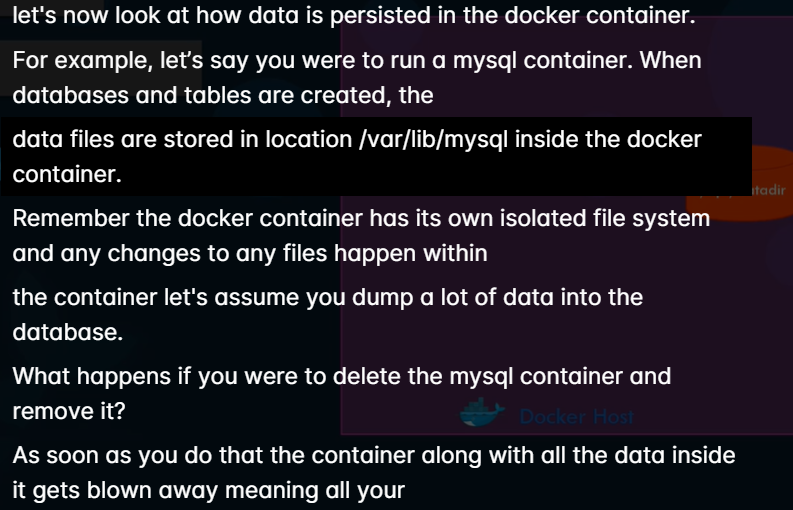
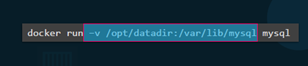
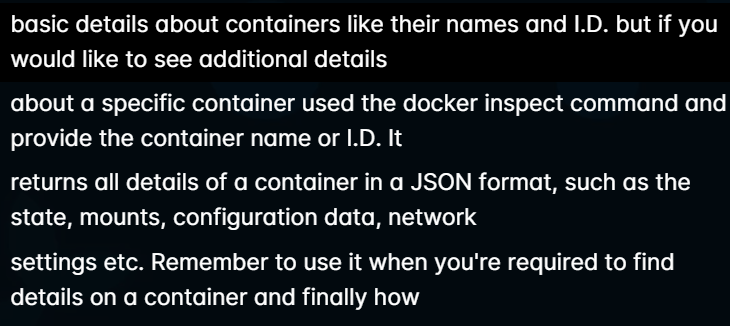
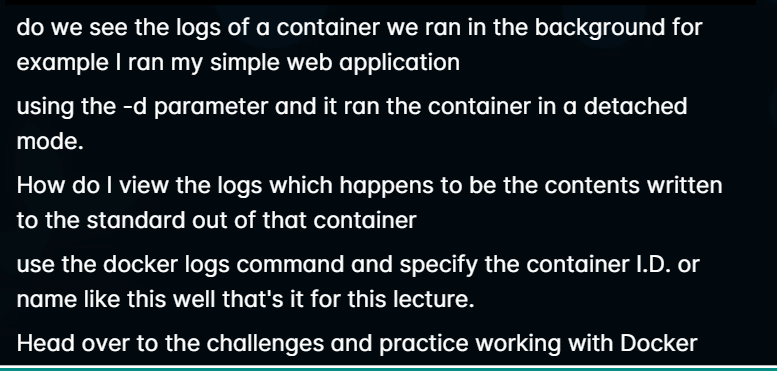
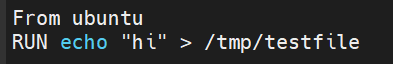
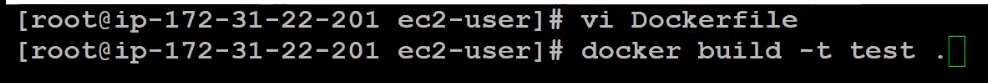
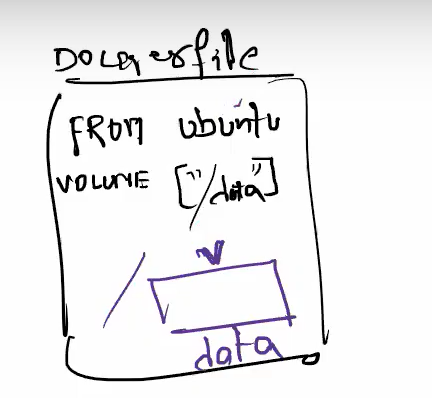
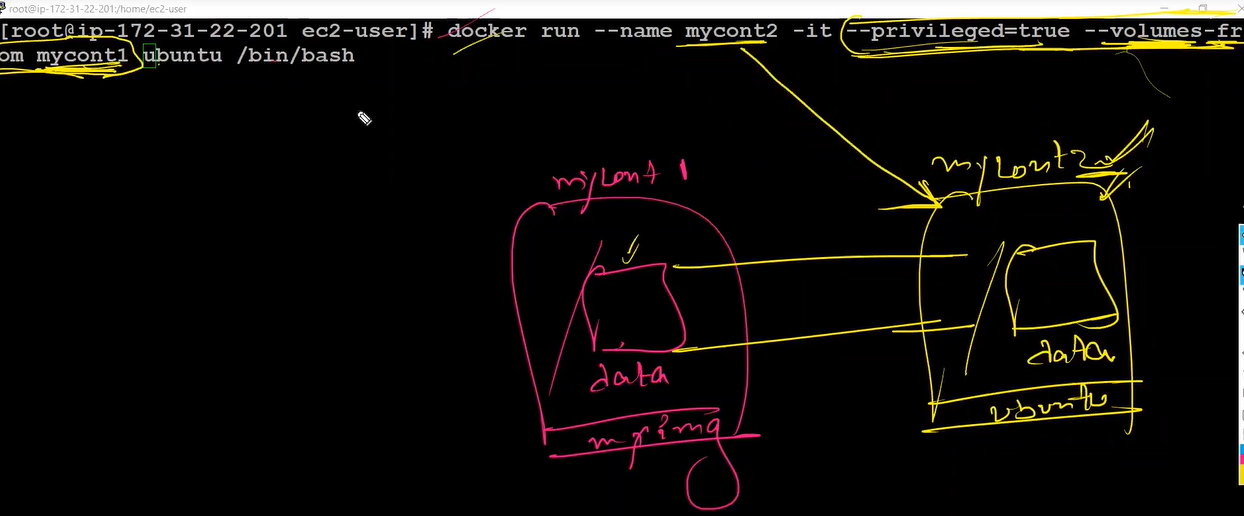
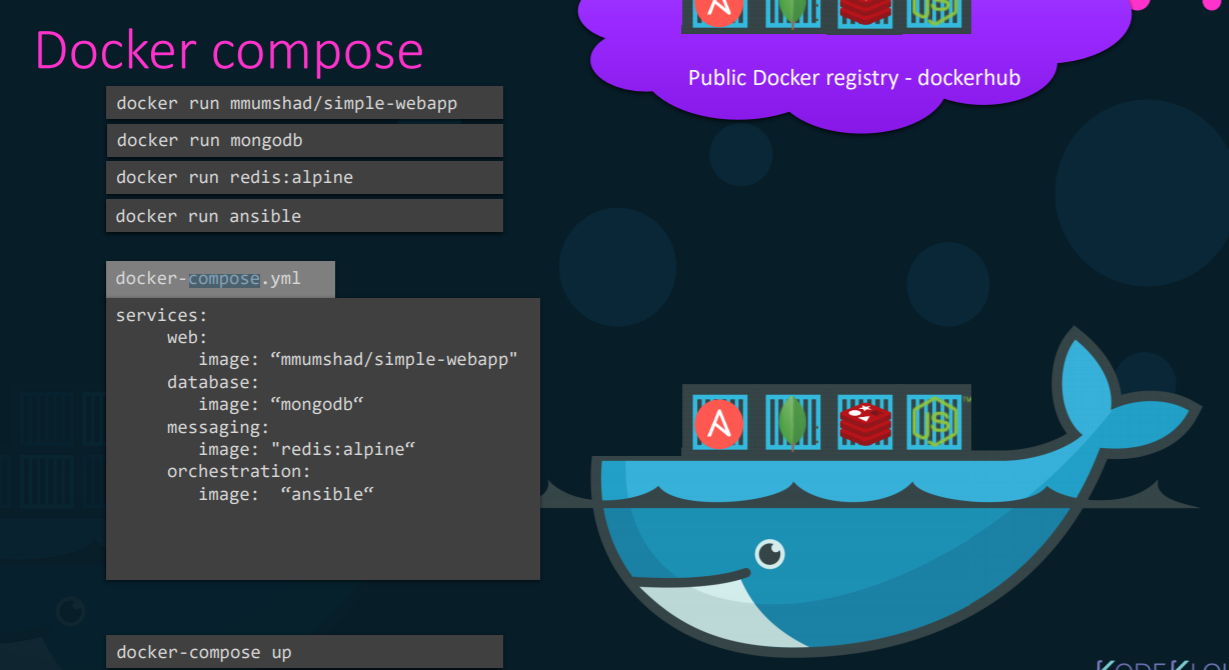
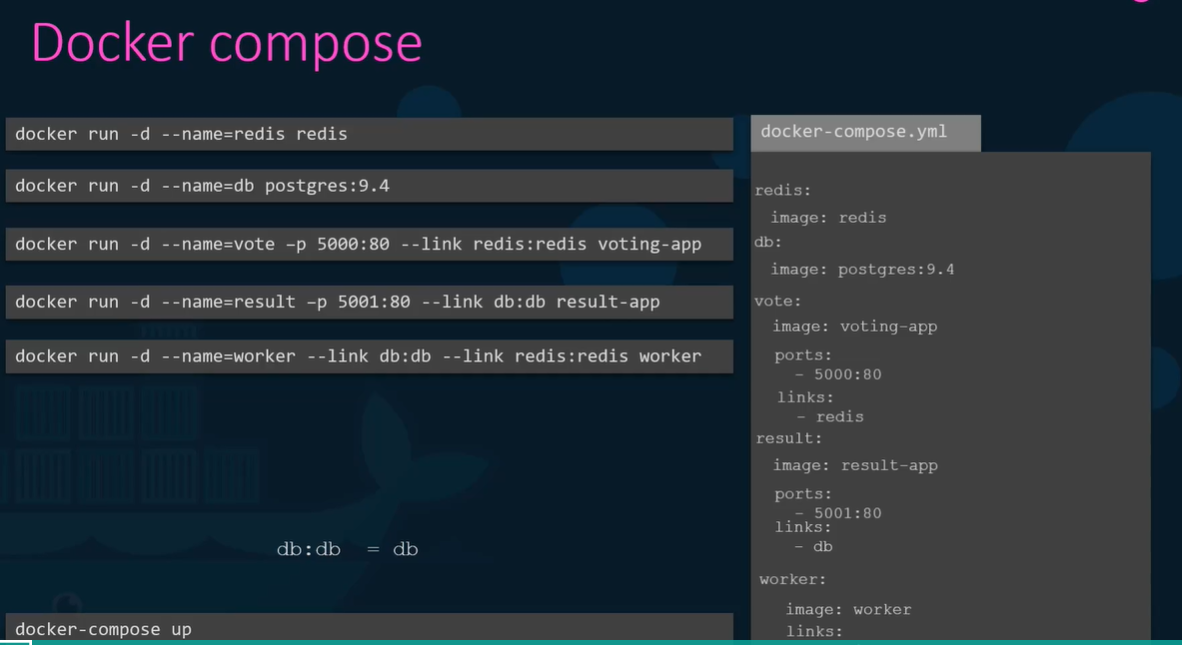
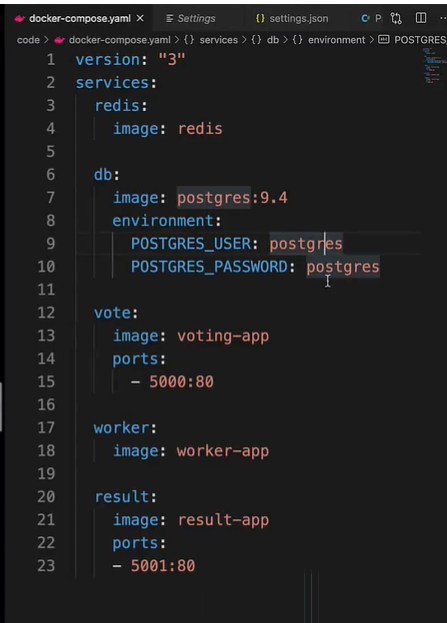
Docker Inshort:

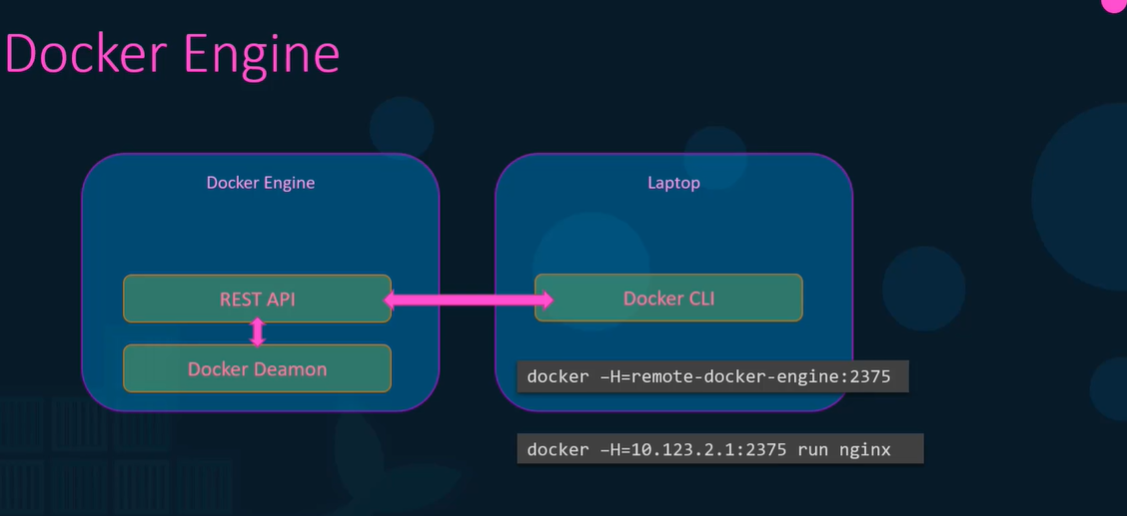
* Why Docker came into picture: Application Stack might have multiple technologies like Webserver using NodeJS, Mongo DB and Messaging service like redis, there are issue w.r.t compatibility with underlying OS and compatibility between services and whe ever we upgrade underlying OS or some services again it might lead to compatibility issue. And also we need to run lot of command inorder to setup in new system, inorder to solve all these issues docker is used
* Containers are completely isolated environments which has there own processes and services and network interfaces similar to a virtual machine
* How Docker Works: 
* 
* 
* **docker ps** 🡪lists all running containers and some basic info like container id, name , status
* **docker ps –a** 🡪 will list all containers including containers which are stopped or exited
* **docker stop <id/name>** -> to stop a container
* **docker rm <id/name>** -> to remove a stopped or exited container
* 
* 
* 
* docker run redis:4.0 🡪 here 4.0 is tag..if we want to run specific version of redis then we use tags, if we don’t specify by default it will consider latest
* Docker Instllation
* Docker package will be already available we don’t need to download, we just need to install
* Yum install docker –y
* After installing docker we need to start docker service
* Service docket status 🡪 to check docker deamon is active or not
* Service docker start
* docker run –I Ubuntu i-> means once container is created it goes inside container –d-> detached mode , -t-> means psudo terminal means when we want container to keep running then we need to provide docker run –dt <image\_name>
* Docker will have internal IP which is accesable only by docker host
* we can map port using the command option –**p docker run –p <docker\_host\_port>:<docker\_container\_port> <image\_name>**
* **docker run –p 8306:3306 mysql**

Docker Volumes:



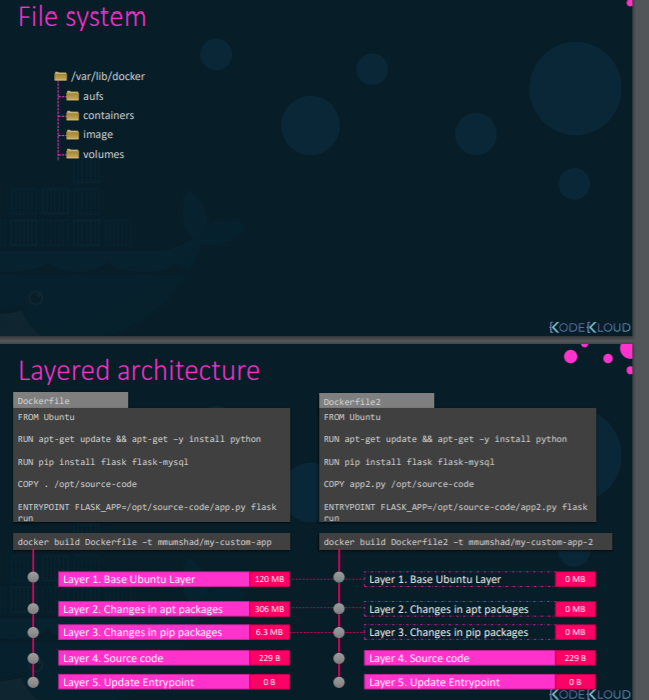
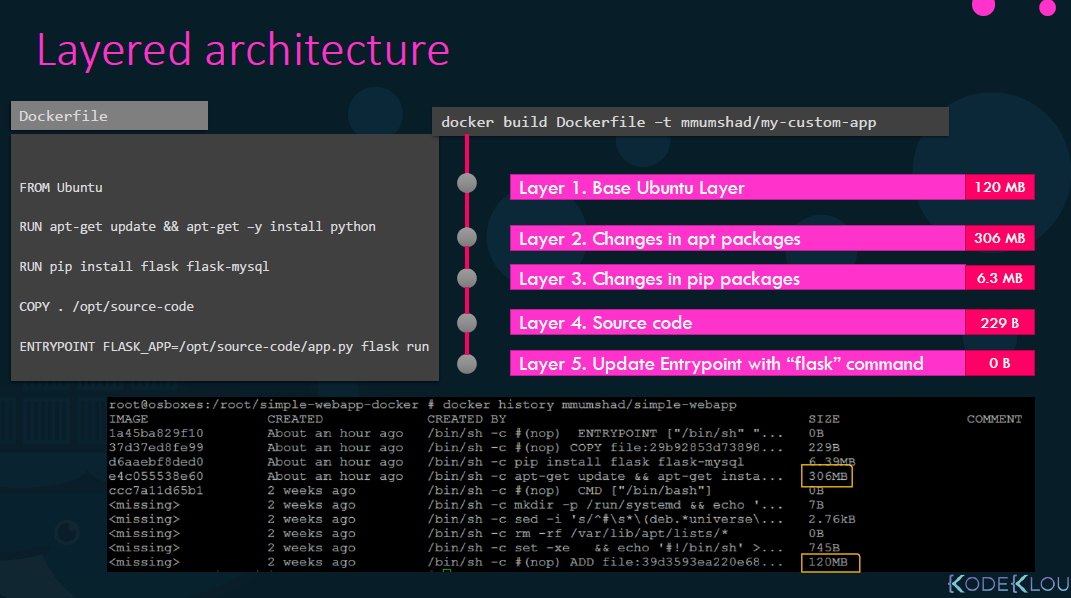
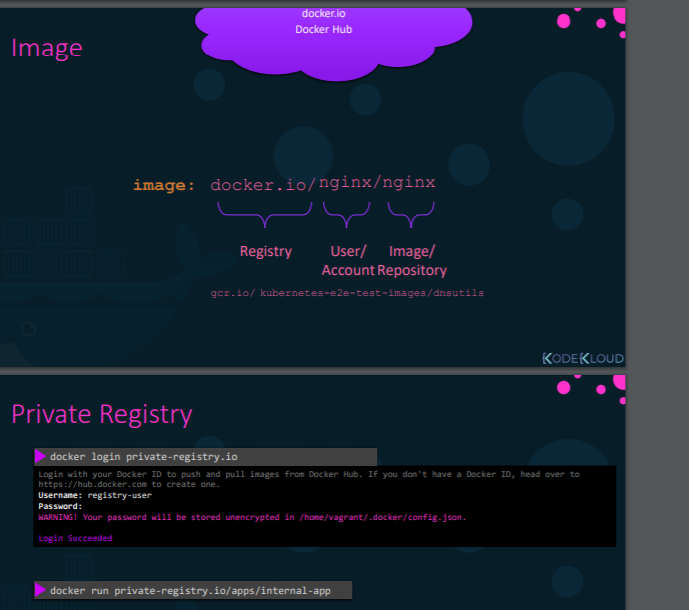
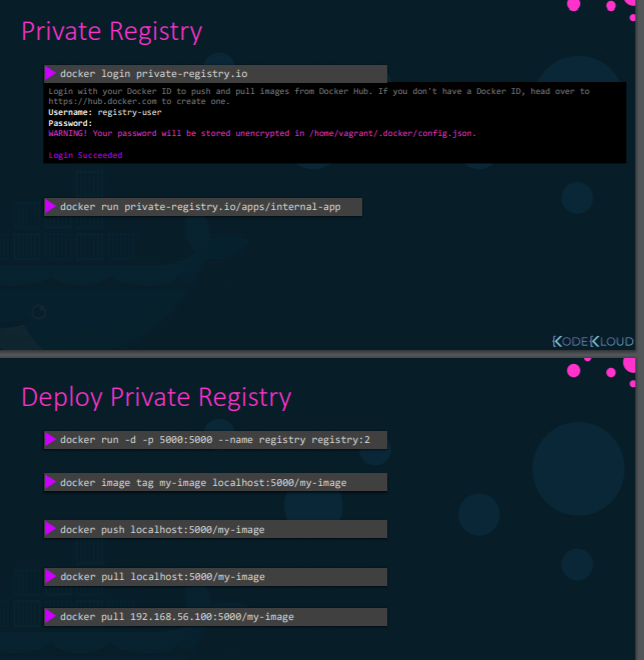
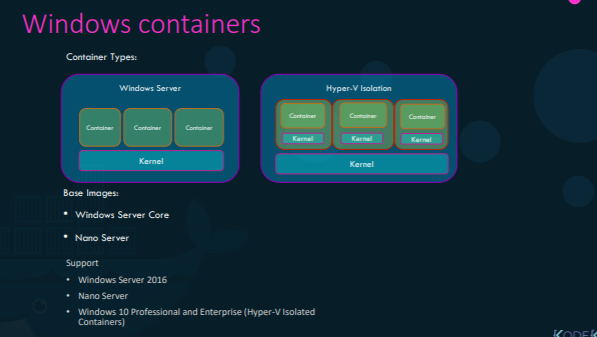
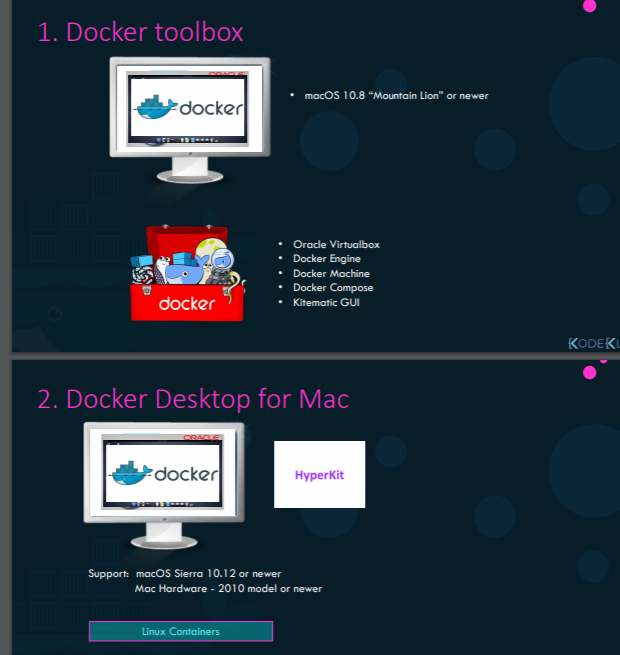
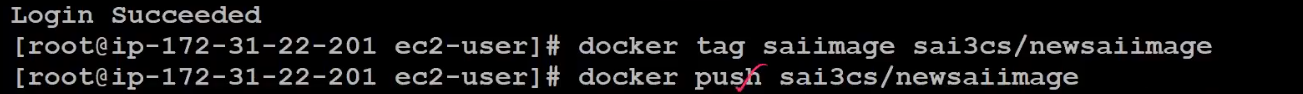


* 
* docker inspect <image name/id>
* 
* 
* 
* Why we need to create an own Image: Either bcz u don’t find a component and
* 
* Dockerfile is a text file written in a specific format that docker can understand
* Everything on the left should start with CAPS
* Every DOcker Image must be based of another image
* Entrypoint allows us to specify a command that will be run when the image is run as container
* 
* -e stands for environment variables
* Docker Compose: Instead of running multiple times docker run command we can provide all the info in docker compose file and run it
* To go inside docker
* Docker attach <docker\_name>
* To rename container docker rename <old\_name> <new\_name>
* Give cutom name while creating container
* Docker run –it –-name saicontainer Ubuntu /bin/bash
* Docker stop <docker\_name>
* docker rm loving\_hellman
* docker rmi <image\_name>
* Dokcer diff <container\_name> will shows all changes done on container
* Docker commit <container\_name> <image\_name>
* There are total 4 steps:
* create file with name Dockerfile
* add instructions
* create image
* create container out of image
* 
* 
* command to create docker image from docker file
* docker build –t test .
* t means tag
* to check docker image is created execute
* docker images
* Volume Sharing between container to container
* 
* 
* 
* 
*  Privileged is true so we get all access from container 1
* 
* 
* 



* Docker Architecture
* When docker is installed three components are installed in it
* docker cli, REST API and docker deamon
* Docker deamon is a backgroud process that manages images, containers volumes
* Docker CLI is the command line where we provide all the docker commands



* There is a way to restict amout of harwdare resource alloted to each container by setting CPU and memory size
* Docker User namspace to create isolation where it creates a child proceed from the parent process id of docker host
* 
* 
* Once docker is installed it creates a folder structure under /var/lib/docker
* All files related to container stored in containers, images in image folder
* Any volumes created under volumes folder
* Each line of instrcution in dockerfile creates a new layer in the coker images with just the changes from previous layer
* These layers are reused if required there by making docker build faster
* 
* 
* Docker registry is places where docker images are kept, It is a cnetral repo for all docker images
* 
* we have two versions of docker in windows
* docker toolbox and docker desktop for mac
* Docker toolbox is a docker on linux VM created using virtial box,
* It run linux containers on widnows system, it uses virtaul box, docker engine and UI called kitematic
* 
* In Docker desktop we remove oracle virual box and use native virtualization technology called Microsft Hyper V
* Hyper V is available in Win 10 Enterprise and Professional Edition or windows Server 2016
* In Docker Desktop the default option is to run a linux container, but if we would like to run a windows container, we need to switch to windows container
* Using this we can create windows based images and run windows containers
* There are two types of containers in windows
* windows server container whcih works like linux container where underlying OS is shared
* Second option is Hyper V Isolation where each container runs with a higly optimized VM guarnteeing complete isolation
* Where as in linux we have multiple base images where as in windows we have two base images
* Webserver Core and Nano Core
* nano core is a headless deployment option for windows server whcih runs at a fraction size of full OS
* where was Webserver core is not that light weight
* Hyper V and Virual Box cannot co-exists parallel
* 
* we have two versions of docker in mac
* docker toolbox and docker desktop for mac
* Docker toolbox is a docker on linux VM created using virtial box, Support 10.8 above
* It run linux containers on mac system, it uses virtaul box, docker engine and UI called kitematic
* Docker desktop is similar to docker tool box but linux VM is created using hyper kit virtualization technology
* Support 10.12 above
* DOCKER HUB:
* First create account in docker hub
* From ec2 enter below command
* 
* Enter username and password
* We need to create repository inorder to push docker images
* It is similar to github, we can push images to hub and it can be shared with others
* Docker tag <image\_name\_present\_in\_ec2> username/<repo\_name>
* docker tag ubuntu atulluta/ubu 🡪 to create repo in dockerhub
* docker push atulluta/ubu 🡪 to push image to docker hub
* 
* docker pull atulluta/ubu