**Docker**

### What is a container?

A way to package application with all the necessary dependencies and configuration. Thus making developing, running and deployment of applications easier. Container is the runtime environment for image

### Why to use a container?

As container is an isolated layer thus it has all the dependencies required, therefore we do not need to install OS or other dependencies individually on each and every machine to develop or deploy the application.

### Docker VS Virtual Machine

|  |  |
| --- | --- |
| Docker virtualizes only the Application Layer | VM virtualizes both Application and OS layer |
| Docker Images are much smaller | VM images are way bigger in size |
| Docker containers starts and run much fast | VM containers starts and run slower as they have to boot the OS every time |

### Basic Commands

docker pull redis (pulls the image of redis from docker hub)

docker images (lists all the images in the local machine)

docker run redis (starts image in the container)

docker ps (lists all the running containers)

docker ps –a (lists all the running as well as stopped container)

docker run –d redis (runs the container in detached mode)

docker stop <container\_id>

docker start <container\_id>

docker run –d –p8000:6379 redis (redis container now runs on port 8000 of your machine)

docker logs <container\_id> (to get logs for the container running)

docker run –d –p8000:6379 –name redis-cont redis (to name the running container as redis-cont)

docker exec –it <container\_id> /bin/bash (to have control of the terminal of the container)

docker container prune (delete all the stopped containers)

### Docker Network

It is an isolated area where container within it can communicate with each just by container name and containers outside it will need port number and id to communicate.

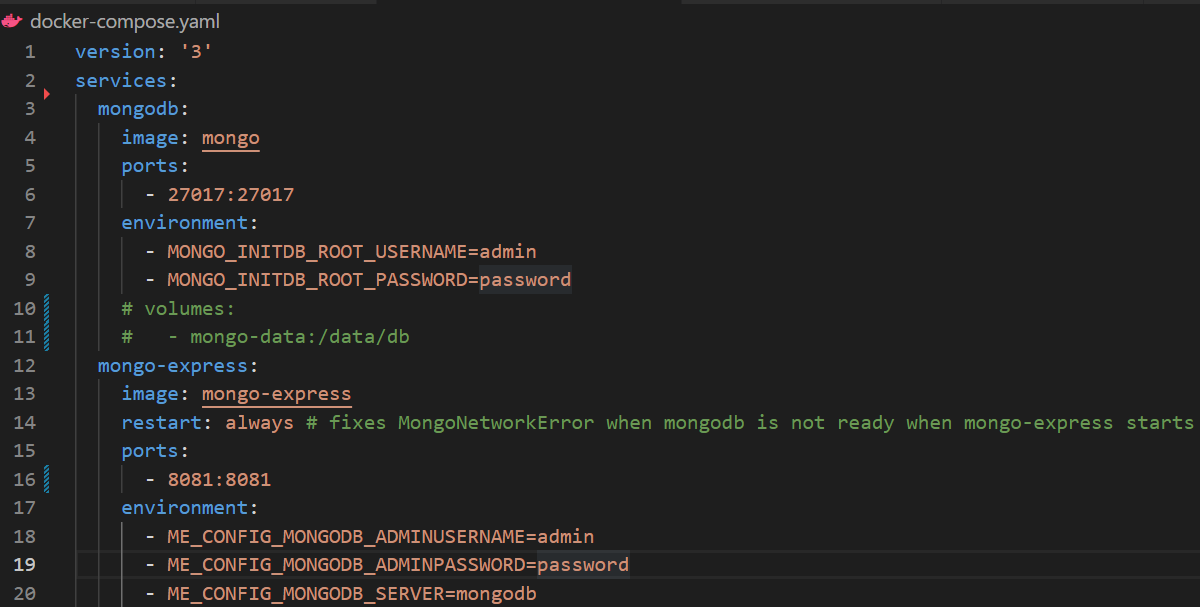
docker network create mongo-network (creates a custom network)

docker network ls (lists all the networks)

### Docker Compose

It is a better alternative to run multiple container than running it via CLI.

#### docker-compose.yaml

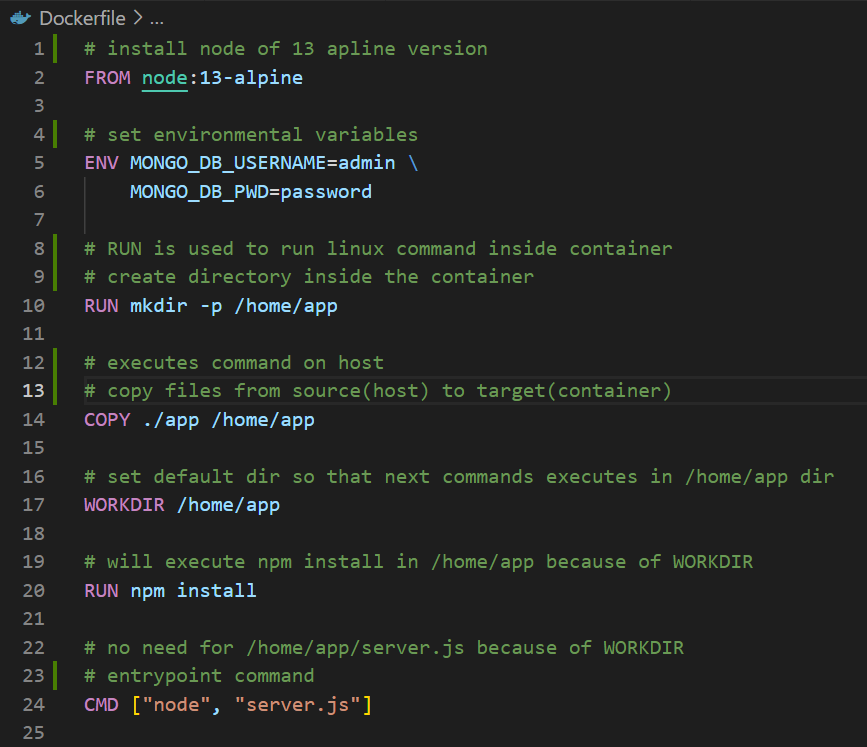
Docker Compose automatically creates a network.

docker-compose –f docker-compose.yaml up (to start up the containers)

docker-compose –f docker-compose.yaml down (to stop the containers)

### Dockerfile

Blueprint for creating docker image



### Build Image from Dockerfile

docker build –t my-app:1.0 . (this has two values, **-t my-app** which is the name of image build and **.** which is the location of Dockerfile in directory)

docker images (to verify if the image “my-app” is created)

### Docker Registry

#### Private Repository (AWS)

Login in to AWS account

Create a repository in AWS ECR

Install AWS CLI in your local machine

Configure AWS credentials by aws configure

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 677266202697.dkr.ecr.ap-south-1.amazonaws.com (Authenticate your Docker Client to your registry)

docker build –t my-app:1.0 . (Build the docker image)

docker tag my-app:1.0 677266202697.dkr.ecr.ap-south-1.amazonaws.com/my-app:1.0 (Tag the image)

docker push 677266202697.dkr.ecr.ap-south-1.amazonaws.com/my-app:1.0 (Push the image)

#### Public Repository (Docker Hub)

docker login (login into docker hub via CLI)

Create a public repo in Docker hub (dockerforrishabh/my-app)

docker tag my-app:1.0 dockerforrishabh/my-app:1.0 (Tag the image)

docker push dockerforrishabh/my-app:1.0 (Push the image)

### Deploying the Application

Update server.js with mongodb for localhost and rebuild the image and push it in repo

Use docker-compose.yaml with three services –> my-app (AWS), mongodb and mongo-express (Docker Hub) to start the application

### Docker Volumes

Docker Volumes are used for **Data Persistence**. Folder in physical file system is mounted into virtual file system of Docker. Thus whatever you write on container file system automatically gets written on host file system and vice versa.

##### Types of Docker Volumes

Host Volume: docker run –v {host\_path}:{container\_path}

Anonymous Volume: docker run –v {container\_path}

Named Volume: docker run –v name:{container\_path}

### Best Docker Practices

* Use official Docker Images as Base Image
* Use specific version of image
* Use small-sized official image based on leaner and smaller OS distros
* Optimize Caching Image Layers (go from least changing to most changing commands)
* Use .dockerignore file and list files you want to ignore from your image
* Make use of “Multi-Stage build” which separates the build dependencies from the final image
* Use Least privileged user
* Scan your image for vulnerabilities using docker scan command

## Key Points

* OCI - The Open Container Initiative is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes
* Docker Runtime: allows us to start and stop containers
* run c: works with OS to start and stop container (low level)
* container d: manages run c. How to get data into container from internet (high level)
* Docker Engine: Docker CLI -> rest API -> server (Daemon)