<https://www.youtube.com/watch?v=3c-iBn73dDE>

Hub.docker.com-🡪official docker repository(contains docker image for applications)

Application Deployment: There is no install softwares into deployment machine, container image has packaged with Graphical user interface, text, application, chat or text message

Description automatically generatedstart script, configuration and for installers like postgres(can download postgres image). Environment configured, jar made, dependencies are build in image.

Conatainer is layers of images. The base image is mostly linux base image because of size and application image is on top of it.

**docker run postgres:9.6**-🡪pulls the docker image from hub.docker.com and starts server, configure and start application. Run is pull + start docker.

**docker pull/run/start/stop/ps/exec-it/logs/images**

**Docker Toolbox**-🡪 suppose wants to run linux docker in windows vm. Here os kernel does not support below win10.

Graphical user interface, application

Description automatically generated

docker container is running environment for docker image. container is virtual environment in os.

docker run -d <redis>🡪run in deattach mode

docker ps -a

docker run -p6000:6379 –name redis-older <redis:4.0> -🡪docker redis needs 6379. vm port 6000 is bind to docker’s 6379 port

docker exec -it redis /bin/bash-🡪go inside container-🡪env (list env) --🡪exit

Diagram

Description automatically generated

**Docker in Software development**

1. Create index.html & node.js script. **node server.js** --🡪 localhost:3000/
2. docker pull mongo
3. docker pull mongo-express (this give UI for DB)
4. Docker Network: connecting two docker containers. Container when started running creates isolated docker network in host/vm. So any docker inside host can communicate by name, no need of using specific port of that docker.

Diagram

Description automatically generated

**docker network ls**

**docker network create mongo-network**

1. docker run -p 27017:27017 -d -e MOGO\_INITDB\_ROOT\_USERNAME=admin -e MOGO\_INITDB\_ROOT\_PASSWORD=admin –name mongodb –net mongo-network **mongo**
2. docker run -d \

-p 8081:8081 \

-e ME\_CONFIG\_MONGODB\_ADMINUSERNAME=admin \

-e ME\_CONFIG\_MONGODB\_ADMINPASSWORD=admin \

–name mongo-express \

–net mongo-network \

-e ME\_CONFIG\_MONGODB\_SERVER=mongodb **mongo-express**

1. check status

Docker logs ………………….id------ (to see created status) or in browse **localhost:8081**

Docker ps

1. server.js created to use mongodb docker and fetch & update records from **localhost:3000/**
   1. docker logs <id> -f
   2. docker logs <id> | tail

**Docker-Compose** (multiple run commands grouped- build utility)

Graphical user interface, application, Word

Description automatically generated

In above there is no docker network specified in compose yaml because compose will create a dedicated network so by name each container can communicate within services.

**docker-compose -f mongo.yaml up (up means start the containers)**

**docker-compose -f mongo.yaml down**

when we restart the container, the stored data will be lost, no persistent. The persistence given by docker volume.

**Build own docker container from node.js application**

Diagram

Description automatically generated

Instead of make linux is the base image, here node is the base image of the application.

1. Create dockerfile

Graphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated

1. Build image from Dockerfile
   1. **docker build -t my-app:1.0 .**
2. js will be committed to Git.

Js + Dockerfile will be committed to Jenkins and Jenkins will build the image from Dockerfile.

1. docker rm <id> -🡪delete the docker container (when changes to dockerfile, needs to rebuild container)

docker rmi <id> -🡪delete the docker image

docker images

docker build -t my-app:1.0 .

**Docker Registry**

Create a private repository in aws.>> aws.console.com in browse >> services >> ECR (my-app name of repo created)

* docker login
* aws private repo login
  + docker tag
  + docker push

**Private Docker repo used to deploy on other server**

In other server, create docker compose file mongo.yaml with image point private repo image

Text

Description automatically generated

* docker login to private repo
* docker-compose -f mongo.yaml

**Docker Volume (data to persist)**

Normally data stored in virtual filesystem of the container. When stopped/restarted, data flushed. Docker volumes makes replication of VFS to host/physical FS.

* docker run -v /home/mount/data:/var/lib/mysql/data (map HFS : VFS)
* docker run -v /var/lib/mysql/data (anonymous volume created by docker)
* docker run -v <name>:/var/lib/mysql/data (name gives reference to the volume)-Named volume

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generateddocker volumes present in local machine are

. in windows C:\ProgramData\docker\volumes

. in linux, /var/lib/docker/volumes

**Kubernetes**

Kubernetes is open source container orchestration tool developed by Google. It helps manage containerized applications.

Graphical user interface, text, application

Description automatically generated

**Need of orchestration tool is** 1. When changed from monolith to microservices 2. Lot of containers

* High availability or no down time
* Scalability or high performance. Small size, faster deployment
* Disaster recovery- backup & restore

**Kubernetes Components**: smallest unit is pod. Abstraction over container. Pod contains 1 docker container or 1 app. Each pod gets own ip. Interacting each pod by ip is difficult when one pod crash and recreate. other component called Service used instead of ip. Ingress component helps maintain map of domainname to ip:port. Ingress communicates with Service. ConfigMap helps to map pod, application information. Secret component helps maintain the password, certificates. Volumes components to store data(db, log) into HFS. Deployment can replica the pod of application for HA. But db pod cannot be replicated to avoid inconsistence. This can be overcome with SatefulSet component.

**Kubernetes Architecture:** Node or worker node created per vm may cantain multiple pod. It contains important components(kubelet, kubeproxy, container runtime). Master node manages clusters(no. of worker node, communicate between, create new node). It contains Api Server, scheduler, controller manager, etcd.

**Minikube and kubectl setup -one node cluster**

On dev env, minikube is create virtual box and creates master, worker node in same vm. Kubectl is used communicate between cluster.

minikube start –vm-driver=hyperkit/oraclerVM

kubectl get nodes

minikube status

kubectl get pod

kubectl get services

We are directly creating pod. The deployment creates required pod. Deployment is blueprint for creating pod.

**kubectl create deployment NAME –image=image**

**kubectl create deployment nginx –image=nginx** -🡪downloads docker image from hub.docker

kubectl get deployment

kubectl get pod

kubectl get replicaset

kubectl edit deployment nginx-depl

Thus **Deployment** manages **Replicaset** manges **pod** is an abstract of a **container**.

**Debugguing pods**

Kubectl logs <name>

kubectl exec -it [pod name] – bin/bash

kubectl delete deployment [deply name]

kubectl create deployment [deply name] image option1 option2 [or] kubectl apply -f [conf file]

Text

Description automatically generated

kubectl get pod **-o wide**

kubectl get deployment

kubectl get service

kubectl **describe** service [service name]

yaml file has 3 parts(metadata, spec, status) connecting deployment to service to pods.

Kubectl get deployment [deploy name] -o yaml > file.yaml 🡪outputs in yaml format

Graphical user interface

Description automatically generated

kubectl get all

🡪create mongo.yaml, mongo-secret.yaml

Kubectl apply -f mongo-secret.yaml