**Docker overview**

Docker is an open platform for developing, shipping, and running application. By taking advantage of docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in productions.

**The Docker Platform**

Docker provides abililty to package and run an applications in a loosely isolated environment called container.

Container are lightweight and containe everything needed to run the application so you donot need to rely on what is currently install on the host.

**What can I use docker for ?**

Fast, consistent delivery of your applications

Container are great for continuous integrations and continuous delivery **(CI/CD)** work flow.

**Responsive deployment and scaling**

Docker’s container-based platform allows for highly portable workload.

**Running more workload on the same hardware**

Docker is lightweight and fast.

**Docker Architecture**

docker uses a client-server architeuture.

Docker daemon which does the heavy lifting building running and distributing your container.

**The docker daemon**

The docker daemon dockerd listens for docker API request and manages Docker such as image container volume network.

**The docker client**

The docker client docker is a primary way that many Docker users interact with docker.

**Docker desktop**

Docker desktop include the docker daemon, dockerd docker client docker docker compose docker content trust kuberrnetes and credential helper.

**Docker registries**

A docker registry stores docker images. Docker hun is public registry that everyone can use and docker is configured to look for image on docker hub by default.

**Docker pull or docker run** cmd pulls the image from required registry

**Docker push** cmd image is pushed to the registry**.**

**Docker objects**

**-images**

**-container**

**-networks**

**-volumes**

**-plugin**

**IMAGES**

An image is read only template with instruction for creating a docker container**.**

**Contianer**

A container is a runnable instance of an image. You can create start stop move or delete usinf docjer api or cli.

We can connect a container to one or more network attach storage to it or even a create new image based on its cuurent state.

Always remember /bin/bash and -I interreactive –t terminal –d detach –p port .

**The underlying technology**

Docker is written in the go programming language.

Docker uses a technology called namespaces to provide the isolatied workspace called the container. When you run a container, Docker creates a set of namespaces for that container.

What is container ?

Is a sandboxed process on your local machine that is isolates form all other process on the host machine.

Can be run on local machine virtual machine or deployed to the cloud.

What is container images ?

When running a container it uses an isolated filesystem. This custom filesystem is provide by a container image. Since the image contain the container filesystem. It must contain everything needed to run an applications all dependencies, configurations script, binaries other configuratins like environment variable default command to run and other metadata.

Yarn

Yarn

Java script and dependency management tools that helps users to automates the task installing updating removing and configuring npm pacakgae.

Java script runtime environment

Yarn use node.js to track libraries and dependencies.

Containerize applications

Docker build -t (name ) . (this will run dockerfile to pull images )

Docker run -dp 3000:3000 (name) this cmd will makes container and run it

-d means detach mode and –p means port

Update applications

Update your sources code [path src/static/js/app.js]

Build the image

Run the container

Remove the old container if you don’t remove the old container it shows errors

Share the applications

Create docker id for sign in

Create repo give name similar to your project

Docker login

Cmd docker login –u username

Docker tag getting-started username/getting-started

Docker push username/getting-started

docker run -d ubuntu bash -c "shuf -i 1-10000 -n 1 -o /data.txt && tail -f /dev/null" bash shell

cat /data.txt

**container volumes**

volume provides th ablility to connect specific filesystem path of the container back to host machine

if the directory of the container is mounted, changes in that directory is also see on host machine.

Persist db how ?

Docker create voume todo-db

docker run -dp 3000:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started

--mount = specify a volume mount

Src= todo-db

Target=/etc/todos gettingstarted

Docker volume inspect [id ]

Nodemon tools [remember]

Bind mount

docker run -it --mount type=bind,src="$(pwd)",target=/src ubuntu bash

--mount docker to create a bind mount

Src is the current directory on your host machine

Target is where the directory should appear inside the container /src

docker run -dp 3000:3000 \

-w /app --mount type=bind,src="$(pwd)",target=/app \

node:18-alpine \

sh -c "yarn install && yarn run dev

-dp detach and port

-w /app current working directory

--mount type=bind, src=”$(pwd)”, target=/app \ bind mount the current directory from the host into /app directory

Node doesnot take bash scripting

Docker log –f container id

Multi-container apps

Container Networking

Simply if they are in same network they can talk to each other if they aren’t they can’t.

Mysql

Docker network create [related name like app-db]

docker run -d \

--network todo-app --network-alias mysql \

-v todo-mysql-data:/var/lib/mysql \

-e MYSQL\_ROOT\_PASSWORD=secret \

-e MYSQL\_DATABASE=todos \

mysql:8.0

docker exec -it <mysql-container-id> mysql -u root –p

show DATABASE

environment variable for mysql

* MYSQL\_HOST - the hostname for the running MySQL server
* MYSQL\_USER - the username to use for the connection
* MYSQL\_PASSWORD - the password to use for the connection
* MYSQL\_DB - the database to use once connected

mysql> ALTER USER 'root' IDENTIFIED WITH mysql\_native\_password BY 'secret';

mysql> flush privileges;

docker run -dp 3000:3000 \

-w /app -v "$(pwd):/app" \

--network todo-app \

-e MYSQL\_HOST=mysql \

-e MYSQL\_USER=root \

-e MYSQL\_PASSWORD=secret \

-e MYSQL\_DB=todos \

node:18-alpine \

sh -c "yarn install && yarn run dev"

docker run -it --network todo-app nicolaka/netshoot which ship with a lots of tools