Docker:

- docker version

List docker version with docker engine version and its all details.

- docker info

List all config information of docker. Containers, images, running, paused, stop, networking etc.

- docker <commands> <sub-commands> options

Run commands on docker using this syntax.
e.g docker container run

Image VS Container:

Image :
Image is the binaries, source files of the application we want to run.
E.g Nginx, node's etc

Container:
Container is an instance of running image as a process.
Create many containers of same image.

-- docker container run --publish 80:80 nginx

Named container:

— docker container run --publish 80:80 —name webserver nginx

This commands pull nginx image from docker hub and using publish commands map to port 80:80.

If port is busy then use other port like
8888:80 , 8989:80
then open localhost:8888.

— docker container run --publish 80:80 —detach nginx

Above command run container in the background.

List Container:
— docker container ls
— docker container ls -a

Stop container:
— docker container stop <container_name/id>

Logs:
-- docker container logs <container_name>

See process running in docker container os:
— docker container top web server

Remove Container:
Stop container first and then remove containers:

— docker container rm containerid1 container2

Force delete:
— docker container rm -f containerid1

END SESSION

Details Listed in 1 container:
Docker container top

see 1 container logs.
— docker container run inspect.

Performance stats of all containers
— docker container run stats

Work in container shell

-- docker container run -it (run new container interactively)

Sometimes we need to work inside container using cli or bash. For work in container bistro we need to run this command.

— docker container run -it bash

You can do everything you can do in linux shell using this. Downloading extra package, custom config settings. Etc

E.g:

Pull ubuntu image and interact with its shell.

-- docker container run -it ubuntu ubuntu

-- apt-get update

Stop:
— exit

Restart previous container with settings:

— docker container start -ai ubuntu

MYSQL:

See existing running container of mysql and do config in it:

— docker container exec -it mysql bash
Ps aux to check mysql running process.

— docker container exec -it (run additional commands into existing container)

Note:
This whole thing is best for trouble shooting container, custom configuration etc

---

END

Docker Networks: Concepts

We can create a separate single network for applications so they can communicate to each other instead of exposing to outer world.

E.g mongo_db and node’s containers

“Batteries include, but removable”

Make new virtual networks.
Attach containers to more than one virtual network.
Skip virtual networks and using custom host IP.

Commands:
 -p : Publish <Host: container port>

— docker container run -p 80:80 —name nginxweb -d nginx

— docker container port nginxweb
80/tcp 0.0.0.0:80

-- docker container —format '{{ .NetworkSettings.IPAddress }}'

COMMANDS

Show networks:
-- docker network ls

Inspect network:
-- docker network inspect

Create a network:
-- docker network create —driver

Attach a network to container:

— docker network connect

Detach:

—docker network disconnect

Remaining section will continue from some other day.

Containers Images

---

1. Basic of images
2. What is an images.
3. How to find images on internet.
4. Manage images on machine
5. Create own images.

What is an image and what is not?

App binaries and dependencies.
Metadata about image data and how to run image.
Not complete OS just binaries that is needed by application.
Small as one file like gaoling static binary.

—> UBUNTU

DOCKER HUB:
Good public images
Download images.

Search images on docker hub.
Go with official one.
Go image documentation and their version.

— docker image ls

— docker pull nginx // get latest image.

— docker pull nginx:1.11.9 // get image by tag

Image Layers

Image Layers:
Union file system is used to design images.

-- docker image ls

Give image history: who create it and change it modified it.
— docker history nginx:latest

Every image start from blank layer. Layer maybe dozen or one.

— docker image inspect
Give us all the details of image (metadata, binaries)
Exposed ports etc etc

TAGING:

— docker image tag —help

Retag existing image:

— docker image tag nginx salman/nginx

If tag not provide then it consider it latest

-- docker image tag —help

— docker image push salman/nginx

-- docker login
— docker logout

— update existing
— docker image tag salman/nginx salman/nginx:testing

Docker File

Default name: Dockerfile

Commands:
FROM Debian:jessie # image of jessie release

ENV NGINX_VERSION 1.11.10-1 # settings nginx version using environments variable

RUN # file edit, run shell script, anything you want to run at that time in terminal

RUN ln -sf /dev/stdout /var/log/nginx/access.log # logging

EXPOSE 80 443 8080 # exposing port to outer world

CMD [‘nginx’, ‘-g’, ‘daemon off;’] # final command to run after start and stop container.

What we do we this file:

Build image:

— docker image build -t custom-nngix . # . current directory

— docker image ls # show image in list

Sample 2:

FROM nginx:latest # official version of nginx image

WORKDIR /usr/share/nginx/html # change directory instead of running RUN cd to path.

COPY index.html index.html

— docker image build -t nginx-html .

-- docker container run -p 80:80 —rm nginx-html

Give Tag to image:

— docker image tag nginx-html:latest salman/nginx-html

— docker image push

ASSIGNMENT:

Build your own image using docker file.
Take node.js app and dockerize it.
Build file
Test it
Push it
Run it

Use alpine version of official node.
See result on localhost
Tag image and push docker hub
Remove from local and pull it from docker and then run.

Persistent Data

---

1. Using Data volumes
2. Bind Mounts Data volumes.

Containers are usually immutable and ephemeral. (Means disposable)

Limitation of containers.

“Immutable infrastructure”: Only re-deploy containers, never change.

Here is a trade of:
What about your databases, unique data which containers are producing.

Docker gives us features to ensure “Separation of concerns” here.

When we remove container only then data removes as binaries of container ,
UFS files also remove.

Here comes a persistent data requirement.

Docker allow us to use Volumes and Bind Mounts outside of container to store unique data.

Bind Mounts: Link container path to host path.

Volumes:

VOLUME command in Dokcerfile

Voumne need extra step to remove it. It cannot remove on container cleaning.

- docker container inspect <contianer-name>
- docker volume ls

In linux machine we can see these volume places by going through it.
But in MACOs and Windows docker is doing hidden things at backend using Linux VM.

If we remove container our volumne/data is saved.

Here is one problem:
Data is available as an executable.

We need to assigned a name to volumes using named volumes.

- docker container run —name mysql -e MYSQL_ALLOW_EMPTY_PASSOWRD=true -v mysql-db:/var/lib/mysql mysql

Create a volume in the host system using name.
If we create a new container and assigned the same volume as we created previously it will assigned easily.

Sometimes we need to create a docker volume ahead of time.

- Docker volume create

ASSIGNMENT:

- Database upgrade with containers
- Create a Postgres container with named volume sql-data using version 9.6.1
- Use docker hub to learn VOLUME path and versions needed to run it.
- Checks logs, stop container.
- Docker volume ls.
- Create new container with previous volume.

Bind Mounting:

For local development:

Maps a host file directory to a container file or directory.
Basically just two locations accessing/point to the same file(s).
Again, skips UFS and host files overwrite any in container.

Cannot use in Dockerfile, must be at container run

—— run -v /users/esketchers/stuff:/path/container

Serve nginx html from host system: (Mac in my case)

- docker container run —name nginx -p 8080:80 -v $(pwd):/usr/share/nginx/html nginx

Open CLI of container:

- Docker container exec -it nginx bash

Docker Compose

Containers run single, independent using dockerFile. What is the solution to run all the web stack container together. Here comes the docker compose. Which is basically a configuration file which help us to run all container together. Handle relationships between different container.

1. YAML file formatted. (Configuration language)

A YAML file contains

1. Containers
2. Networks
3. Volumes

2. CLI tool docker-compose used for local dev/test automation with those YAML files.

Let discuss docker-compose.yml:

Version : 1, 2, 2.4, 3.1 etc

YAML file used with docker-compose command for local docker automation.

With docker directly in production with swarm.

- Docker-compose.yml

```
Version: ‘3.12’

Services: # containers run
       servicename: # friendly name
              image:
              Command
              Environment
              Volume
              Ports:
                 - ’80:8080’
       Servicename2:

Volumes: # docker volumen create

Networks:
```

DOCKER-COMPOSE CLI:

- Docker compose download separately for linux.
- It is best for development and test not for production.
- Two most common commands are:
- — docker-compose up # setup volumes/networks and start containers
- -- docker-compose down # stop all container and volumes

ASSIGNMENT:
Write a compose file

Drupal CMS. (Dokcer hub image)

Use Drupal and Postgres image
Use ports to expose 0n 8080
Set POSTGRES_PASSOWRD for Postgres.
Walk through Drupal setup via browser.

Drupal assumes DB is localhost, but it service name.
Use Drupal unique data.

Solution:

Make new file: docker-compose.yml

Version: ‘3.1’

```
Services:
   Drupal:
        Image: Drupal
        Ports:
          - “8080:80”
        Volumes:
           Drupal-modules:/var/www/html/modules
           Drupal-modules:/var/www/html/profile

   Postgres:
          Image: postgres
          Environment:
             - POSTGRES_PASSWORD=mypassword
```

Swarm

How do we automate container lifecycle on the heroic, amazon, azure etc?
How we scale up, manage it with few member of team?
How to replace container without stoping anything or downtime?
How create cross-node virtual network?
How to control track where container get started?
How to sure that machines on which we run containers are valid or trusted ones.?

How we can store secrets, keys, passwords ?