DevGeekWeek 2020
INSPIRED BY INNOVATION

matrix DevOps

ATLASSIAN Platinum Solution Partner

Docker admin & Hands-On

Nahshon (Nash) Paz
nash@matrix.co.il

- Docker was developed by Docker Inc. but relies heavily on the docker community, Redhat in 2013, MS in 2014
- Uses the resource isolation features of the Linux kernel
- We're talking linux here, not windows (Docker desktop…)

On our list are: DevGeekWeek 2020 INSPIRED BY INNOVATION

- Layers, Images and Containers

  Terminology
- Why Docker
- Microservices
- Layers, Images and Containers

  Examples + hands on
- Volumes and ports
- Use cases
- Extra - Network, multi stage
- Docker Compose intro

Play

Install Docker & Docker Compose - Centos 7

Step 1 — Install Docker

#Install needed packages:

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

#Configure the docker-ce repo:

```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

#Install docker-ce:

```
sudo yum install docker-ce
```

#Add your user to the docker group with the following command.

```
sudo usermod -aG docker $(whoami)
```

#Set Docker to start automatically at boot time:

```
sudo systemctl enable docker.service
```

#Finally, start the Docker service:

```
sudo systemctl start docker.service
```

Step 2 — Install Docker Compose

```
curl -L "https://github.com/docker/compose/releases/download/1.26.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Or just https://labs.play-with-docker.com/

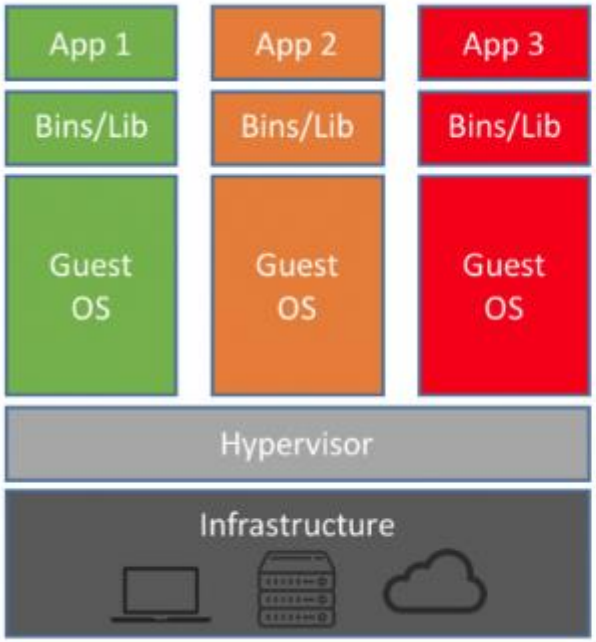
What is a container?

Containers wrap up a piece of software in a complete filesystem that contains everything it needs to run:

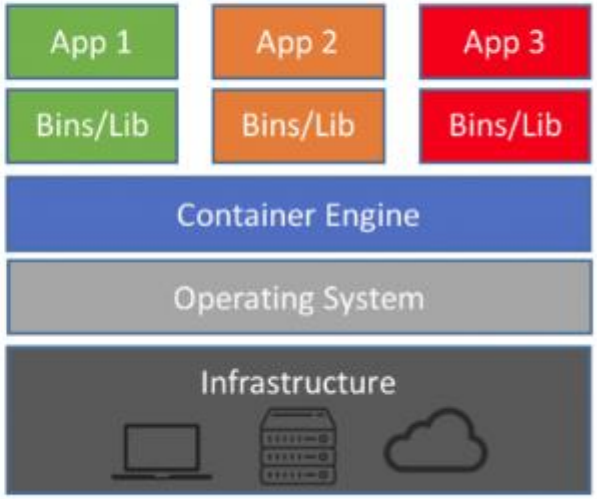
- Code
- Runtime
- System tools
- System libraries

Container vs VM

Whaduyou need an OS for?



Machine Virtualization



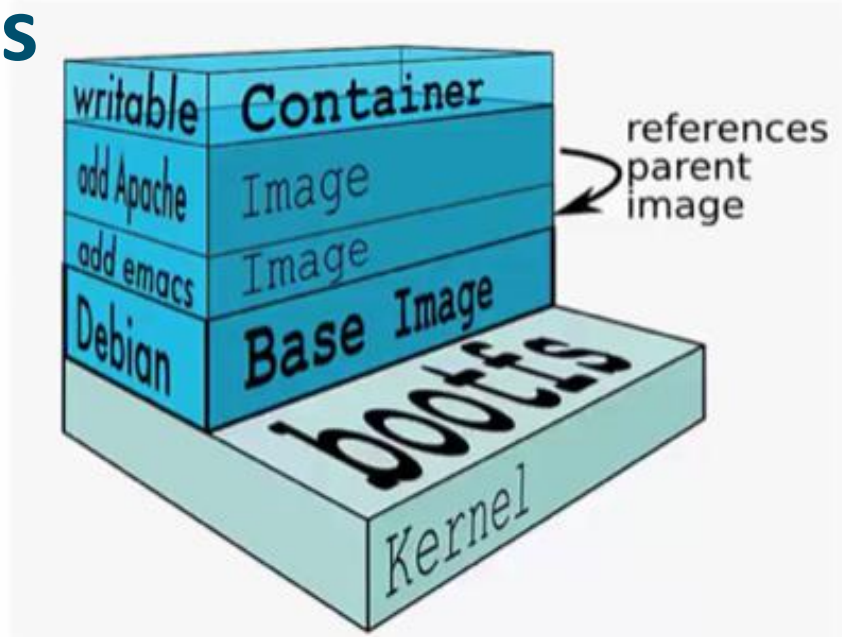
Containers

Python, Java, node, .Net

Docker installed on host

Docker host

Image→ container = Layers (Dockerfile)



- Comprised of multiple layers
- A layer is just another image
- Every layer contains a base layer
- Docker uses a copy on write system
- Layers are read only

Try:

docker history <name of an image>

Lightweight

- Containers running on a single machine share the same operating system kernel;
- They start instantly and use less RAM.
- Images are constructed from layered filesystems and share common files, making disk usage and image downloads much more efficient.

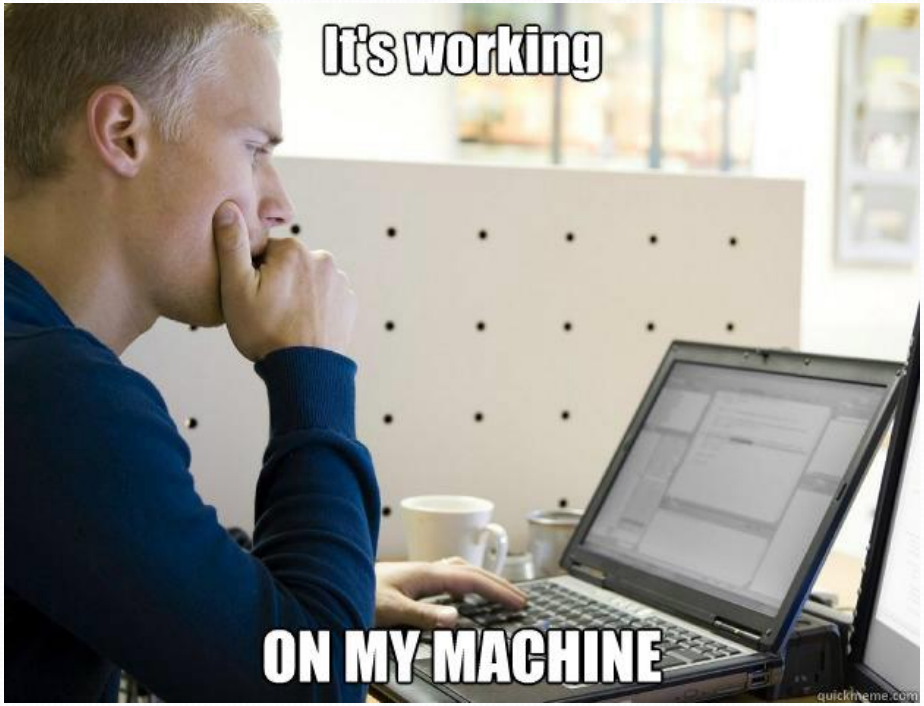
"Ever since we glued on the lightning bolt, he's been working faster."

Environment (as a service)

Packaging an application in a container with its configs and dependencies guarantees that the application will always work as designed in any environment.

Same across multiple deployments

Eliminate "Works on my machine"



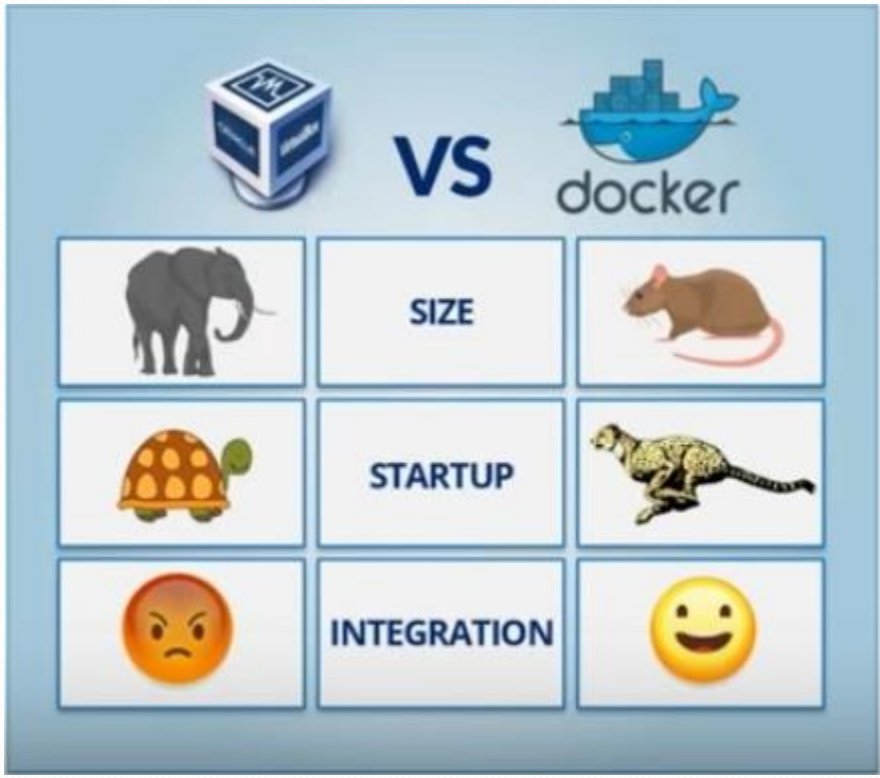
Docker Use Cases

- Development Environment
- Quick evaluation of software
- Microservices
- Multi-Tenancy
- Unified execution environment (dev → test → prod (local, VM, cloud, …)

Micro

Microservices vs monolith

- Environment is packaged with service
- Separation of dev environments, teams
- Same across multiple deployments
- Speed (size, layers)



Docker Hub

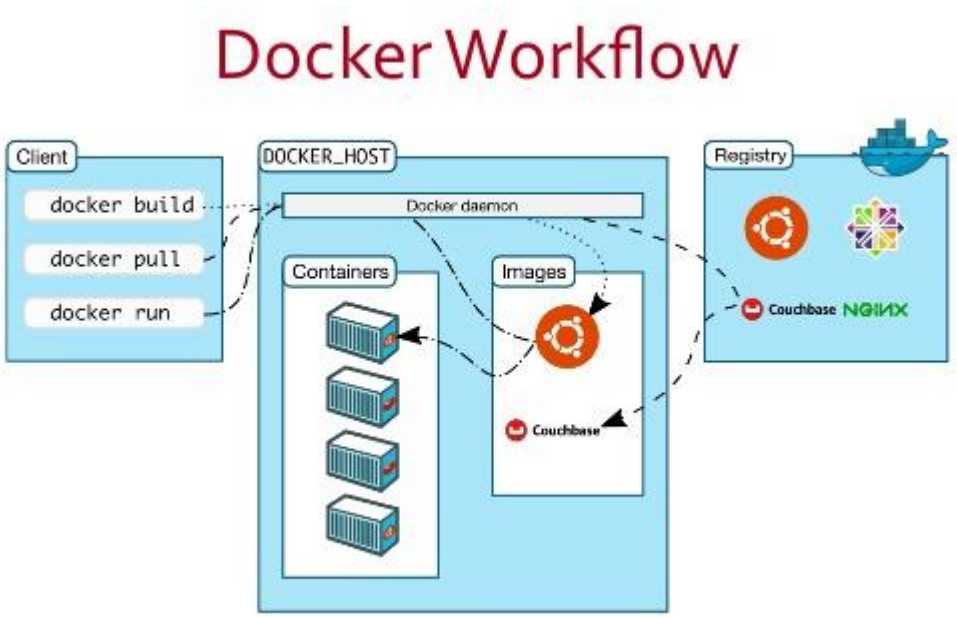
Public repository of many, many Docker images:

https://hub.docker.com/

Google it!

though you can always run:

docker search centos



Hello World

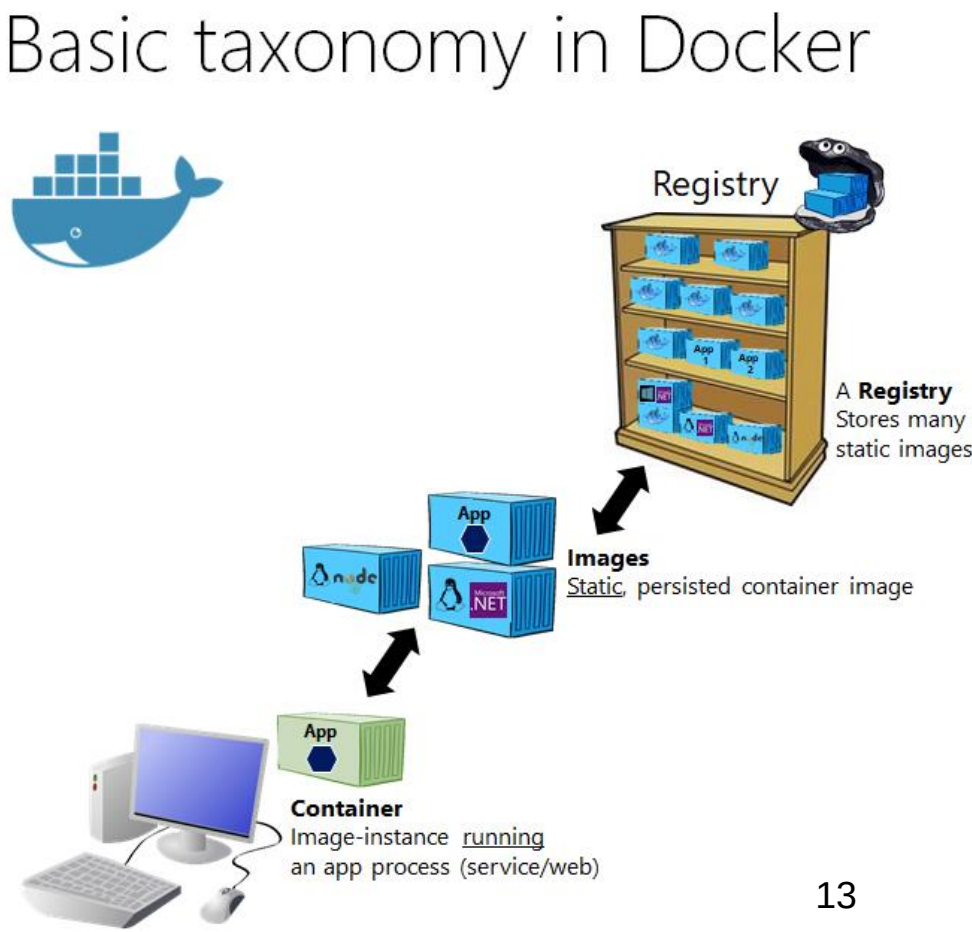
Simple Command - Ad-Hoc Container:

Try:

```
docker run alpine echo Hello World
docker run -d alpine sleep 60

docker images

docker ps
docker container
```



Terminology - Image

Persisted snapshot that can be run

- images (or image ls): List all local images
- image tag: Tag an image
- image pull: Download image from repository
- image rm: Delete a local image
- image save: save to file

Terminology - Container

Runnable instance of an image

- ps (or container ls): List all running containers
- container ls –a: List all containers (incl. stopped)
- container top: Display processes of a container
- container start: Start a stopped container
- container stop: Stop a running container
- container stats: host statistics (networking, RAM, CPU)
- container rm: Delete a container
- container commit: Create an image from a container
- container inspect: full configuration of a container

Terminology - Container

Execute on a running instance of an image

docker exec (for debug purposes)

```
docker exec -it name_or_id sh
docker exec name_or_id sh -c "ls"
```

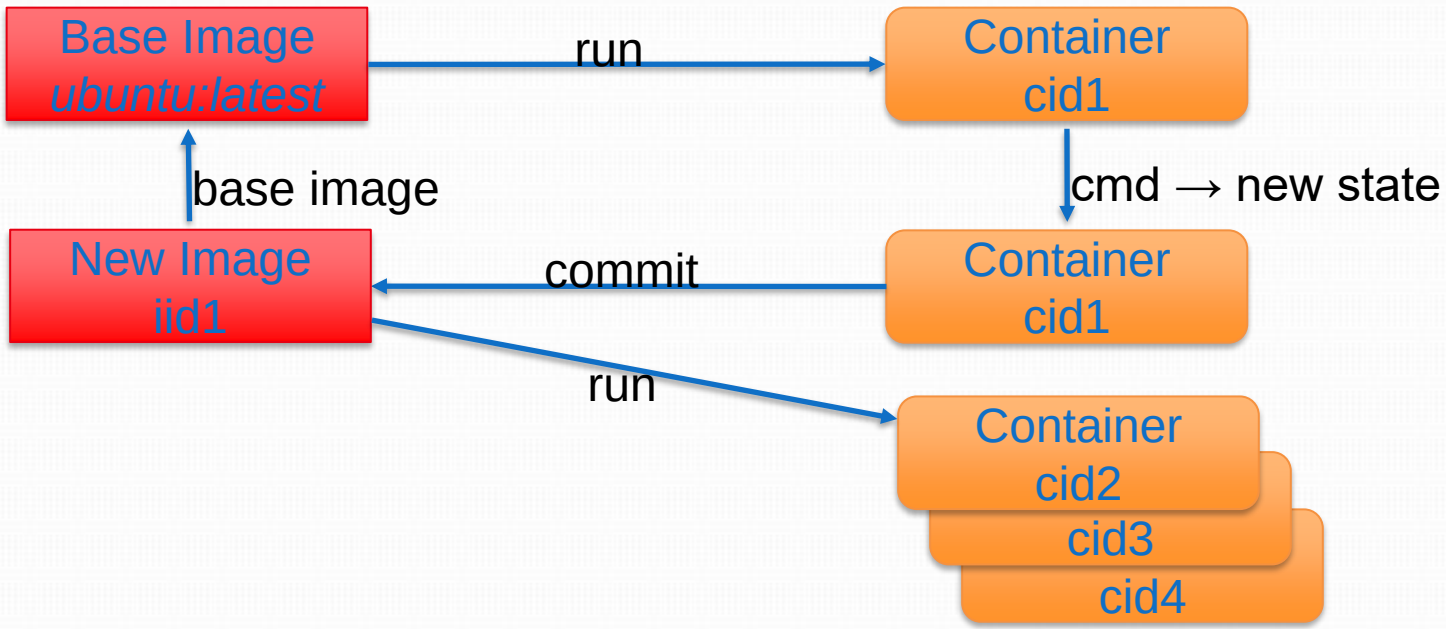
docker logs

```
docker logs -f name_or_id
```

Tip:

```
id = $(docker ps | grep part_of_a_container_name |awk '{print $1}')
```

Image vs. Container




ANY
QUESTIONS
?

Dockerfile

Create images automatically using a build script: «Dockerfile»

Can be versioned in a version control system like Git, along with all dependencies

Dockerfile Example

Try:
Create a directory name Docker-Ex1

```
mkdir Docker-Ex1

cd Docker-Ex1
```

Create a file named message.sh in the Docker-Ex1 directory:

```
#!/bin/sh

echo This is from my dockerfile
```

Create a file named Dockerfile under Docker-Ex1 directory:

```
FROM alpine

COPY message.sh /

CMD ["sh", "message.sh"]
```

```
cd ..  #to go back to the root folder
docker build -t alpine:MyFirstImage Docker-Ex1/ #Note that you must always state the directory in which your Dockerfile exists, . (dot) for the current directory
docker run alpine:MyFirstImage
```

Dockerfile Selenium

from https://github.com/SeleniumHQ/docker-selenium/blob/master/NodeBase/Dockerfile

```
FROM selenium/base:3.141.59-palladium
LABEL authors=SeleniumHQ

USER root

#==============
# Xvfb
#==============
RUN apt-get update -qqy \
  && apt-get -qqy install \
    xvfb \
  && rm -rf /var/lib/apt/lists/* /var/cache/apt/*

#==============================
# Locale and encoding settings
#==============================
ENV LANG_WHICH en
ENV LANG_WHERE US
ENV ENCODING UTF-8
ENV LANGUAGE ${LANG_WHICH}_${LANG_WHERE}.${ENCODING}
ENV LANG ${LANGUAGE}
# Layer size: small: ~9 MB
# Layer size: small: ~9 MB MB (with --no-install-recommends)
RUN apt-get -qqy update \
  && apt-get -qqy --no-install-recommends install \
    language-pack-en \
    tzdata \
    locales \
```

Best practices

- Use cache – changeable layers at the end
- RUN several commands with && as one layer if expecting changes
- .Dockerignore :
- Entrypoints as .sh script rather than a command

```
**/.classpath
**/.dockerignore
**/.env
**/.git
**/.gitignore
**/.project
**/.settings
**/.toolstarget
**/.vs
**/.vscode
**/*.*proj.user
**/*.dbmdl
**/*.jfm
**/azds.yaml
**/bin
**/charts
**/docker-compose*
**/Dockerfile*
**/node_modules
**/npm-debug.log
**/obj
**/secrets.dev.yaml
**/values.dev.yaml
LICENSE
README.md
```

CMD and ENTRYPOINT

CMD = command to run if no arguments replace it,
ENTRYPOINT = image behaves like a binary. (docker run or exec start a service
If the Dockerfile declares both ENTRYPOINT and CMD and no arguments are passed to docker run, then the argument(s) to CMD are passed to the declared entrypoint.

```
FROM ubuntu:trusty

ENTRYPOINT ["/bin/ping","-c","3"]

CMD ["localhost"]
```

Try:

docker build -t ping .

docker run ping

docker run ping docker.io


ANY
QUESTIONS
?

More options

```
docker container run -t -p 8080:80 ubuntu
```

Map container port 80 to host port 8080

```
docker container run -t -e ENV_VAR_A=value1 ubuntu
```

Add environment variable to the container

```
docker container run -d ubuntu
```

Detached mode

Mount Volumes

Volumes help save data outside the container so the data won't get lost when the container stops running.

docker run –ti -v vol_name:/<path inside container> alpine

Terminology - Volume

create	Create a volume
inspect	Display detailed information on one or more volumes
ls	List volumes
prune	Remove all unused local volumes
rm	Remove one or more volumes

Mount Volumes - Cont'

Try:

```
docker container run -ti -v myVol:/home --name alpine alpine sh

cd /home

echo "This is my file" > myfile

ls -l

exit
```

```
docker container rm alpine

docker container run -ti --name alpine alpine sh

cd /home

ls -l

exit
```

Publish Port

Try:
Create a directory named Dockerfile-Ex2

Create a Dockerfile file under Dockerfile-Ex2 :

```
FROM busybox

COPY index.html /www/index.html

EXPOSE 8000

CMD trap "exit 0;" TERM INT; httpd -p 8000 -h /www -f & wait
```

Create a index.html file under the folder Dockerfile-Ex2 –
vim Dockerfile-Ex2/index.html

```
<xmp>Hello World</xmp>
```

Docker image build -t busybox:HelloWorld Dockerfile-Ex2

docker container run -d --name web-test -p 80:8000 busybox:HelloWorld

Running Jenkins on a container

Try:

Google "jenkins docker"

Go to docker hub, https://hub.docker.com/_/jenkins

Oh! DEPRECATION NOTICE

Go to the jenkins/jenkins:lts link → documentation
https://github.com/jenkinsci/docker/blob/master/README.md

Run:

```
docker run -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts
```

In the console you'll see the "initial password"

Wait for it…

Go to http://<your machine's ip>:8080

Micro DB, loads of data

Try:

```
mkdir -p /storage/docker/mysql-data

docker run  --detach  --name=mysql_server_1  --
env="MYSQL_ROOT_PASSWORD=my_password"  --publish 6603:3306  --
volume=/storage/docker/mysql-data:/var/lib/mysql mysql

docker run --rm -it bitnami/mariadb bash

mysql -uroot -pmy_password -h<ip of the docker host> -P6603
USE mysql
CREATE TABLE IF NOT EXISTS tasks (task_id INT AUTO_INCREMENT
PRIMARY KEY,title VARCHAR(255) NOT NULL,start_date DATE,due_date
DATE,status TINYINT NOT NULL,priority TINYINT NOT NULL,description
TEXT,created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
exit
exit
ls -l /storage/docker/mysql-data/mysql
```

Web server random ports

Try:

```
docker container run --name my-nginx-container-1 -v /tmp/docker-host/html:/usr/share/nginx/html:ro -P -d nginx
```

```
docker container ps (note the ports)
```

```
curl http://localhost:<port that's mapped to 80 of the container>
Or browse http://<ip or docker host name>:<port that's mapped to 80 of the container>
```

Try:
vim Dockerfile

```
FROM nginx
COPY /tmp/docker-host/html /usr/share/nginx/html
```

```
docker image build -t my-nginx-image-1 .
```

```
docker container run --name my-nginx-container-2 -P -d my-nginx-image-1
```

```
docker container ps (note the ports for my-nginx-container-2)
```

```
curl http://localhost:<port that's mapped to 80 of my-nginx-container-2 >
Or browse http://<ip or docker host name>:<port that's mapped to 80 of my-nginx-container-2 >
```

Selenium grid

Run:

```
docker run -d -p 4444:4444 --name selenium-hub selenium/hub
```

go to http://<ip or computer name>:4444/grid/console

See:

```
docker logs $(docker ps |grep selenium/hub|awk '{print $1}')
docker ps –a

docker run -d --link selenium-hub:hub selenium/node-chrome
docker run -d --link selenium-hub:hub selenium/node-firefox
```

go to http://<ip or computer name>:4446/grid/console

See:

```
docker logs $(docker ps |grep selenium/hub|awk '{print $1}')
```

Selenium grid – cont'

docker-compose.yml

```
version: "3"
services:
  selenium-hub:
    image: selenium/hub
    container_name: selenium-hub
    ports:
      - "4444:4444"
  chrome:
    image: selenium/node-chrome
    depends_on:
      - selenium-hub
    environment:
      - HUB_PORT_4444_TCP_ADDR=selenium-hub
      - HUB_PORT_4444_TCP_PORT=4444
  firefox:
    image: selenium/node-firefox
    depends_on:
      - selenium-hub
    environment:
      - HUB_PORT_4444_TCP_ADDR=selenium-hub
      - HUB_PORT_4444_TCP_PORT=4444
```

Run:

docker-compose scale chrome=5 up -d

docker-compose scale chrome=5

See: docker ps, browser


ANY
QUESTIONS
?

Docker networks

try:

```
docker network ls
docker run -itd --name=networktest ubuntu #new containers join default bridge network
docker network inspect bridge
docker ps

docker network create -d bridge od_bridge   #-d sets the network type
docker run -d --net=od_bridge --name db training/postgres
docker run -d --net=od_bridge --name web training/webapp python app.py

docker container exec db ping web
```

Multistage CI/CD


```
FROM golang:1.7.3 AS builder
WORKDIR /go/src/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/href-counter/app .
CMD ["./app"]
```

```
# ---- Base Node ----
FROM alpine:3.5 AS base
# install node
RUN apk add --no-cache nodejs-current tini
# set working directory
WORKDIR /root/chat
# Set tini as entrypoint
ENTRYPOINT ["/sbin/tini", "--"]
# copy project file
COPY package.json .
 #
# ---- Dependencies ----
FROM base AS dependencies
# install node packages
RUN npm set progress=false && npm config set depth 0
RUN npm install --only=production
# copy production node_modules aside
RUN cp -R node_modules prod_node_modules
# install ALL node_modules, including 'devDependencies'
RUN npm install
# ---- Test ----
# run linters, setup and tests
FROM dependencies AS test
COPY . .
RUN  npm run lint && npm run setup && npm run test
 #
# ---- Release ----
FROM base AS release
# copy production node_modules
COPY --from=dependencies /root/chat/prod_node_modules ./node_modules
# copy app sources
COPY . .
# expose port and define CMD
EXPOSE 5000
CMD npm run start
```

Clean slate

```
docker stop $(sudo docker ps -a -q) # sends SIGTERM,then SIGKILL
docker rm $(sudo docker ps -a -q) # removes from docker ps -a
docker rm -f #just sends SIGKILL then removes state

docker ps -a
```

Exercise

DevGeekWeek 2020
INSPIRED BY INNOVATION

If you'd like an extra review of the basics:

https://github.com/nashpaz123/Please-Contain-Yourself/tree/master/2-Long_Lived_Containers

If you're comfortable with docker, go for the app:

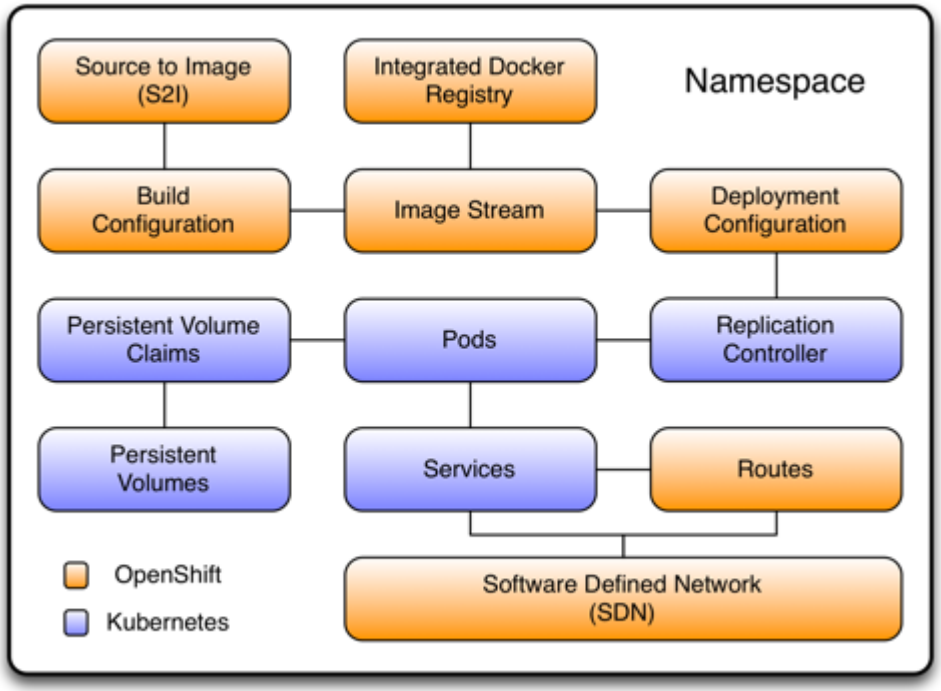
https://github.com/nashpaz123/Please-Contain-Yourself/tree/master/3-Bundle_Your_App_Into_An_Image

If you're done with plain old docker, try docker-compose:

https://github.com/nashpaz123/Please-Contain-Yourself/tree/master/6-Docker_Compose_For_Multi-Container_Apps

What will they think of next?

- GUI (Portainer)
- multiple, scale (Docker-compose)
- Orchestration (Kubernetes)
- PaaS (Openshift)




```
$ vim  docker-compose.yml
```

add following content.

```
version: '3'
services:
  db:
     image: mysql
     container_name: mysql_db
     restart: always
     environment:
        - MYSQL_ROOT_PASSWORD="secret"
  web:
    image: apache
    build: ./webapp
    depends_on:
       - db
    container_name: apache_web
    restart: always
    ports:
      - "8080:80"
```

DevGeekWeek 2020
INSPIRED BY INNOVATION

Nahshon (Nash) Paz
nash@matrix.co.il

JOHN BRYCE
תלמד הייטק. זה עובד!
a matrix company