Software Consulting

DevOps Projects

Corporate Training

“UpGrade Your Business with BEST BRAINS”

Docker Kubernetes

Rajesh G

Master Trainer & CTO,
Brain Upgrade Academy,
A division of Unisuraksha Tracking Systems Pvt Ltd

Docker

Rajesh G

CTO, Managing Partner
https://unigps.in

BrainUpgrade Academy

Training Objectives

At the end of training,

participants should be able to

- ❏ Know Docker & swim with them
- ❏ Bundle applications in Docker images
- ❏ Run Docker Containers

Table of Contents

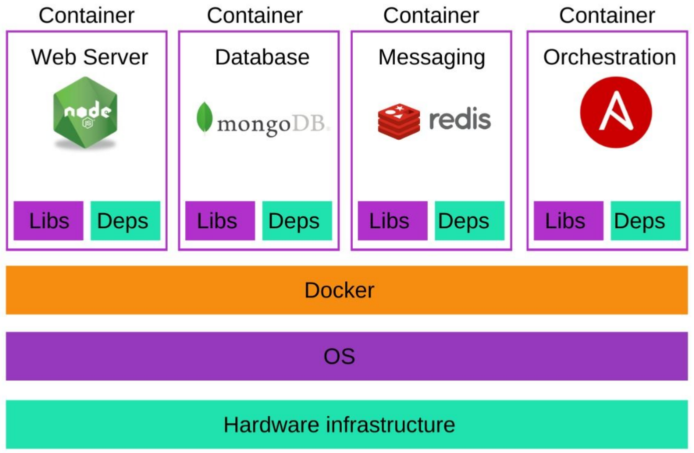
Module 1: Docker Concept & Terms

- What is container & Why?
- Container vs Virtual Machine
- Linux Containers & Docker
- Terminologies in Docker world
- Docker Architecture
- Lab Exercises

What is Container?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Containerization: Use of linux (/ windows) containers to deploy application is called containerization



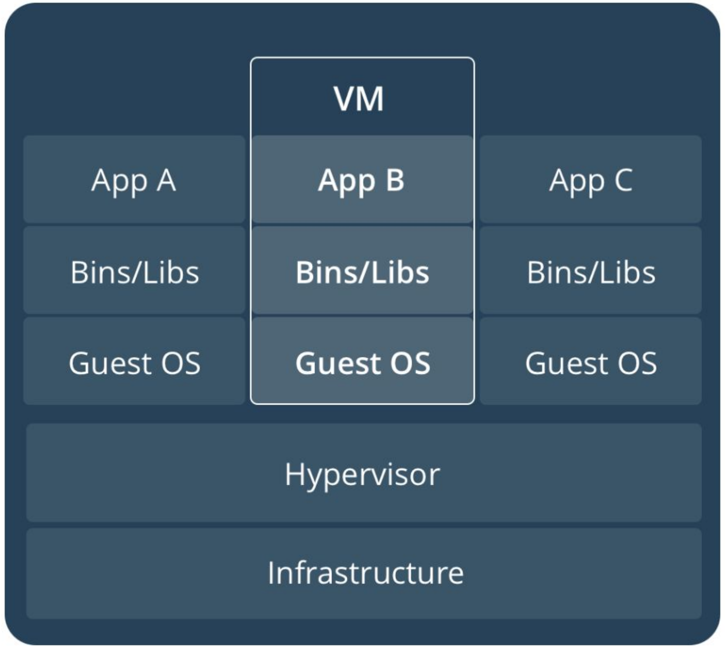
Why Containers?

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly
- Running more workload on the same hardware

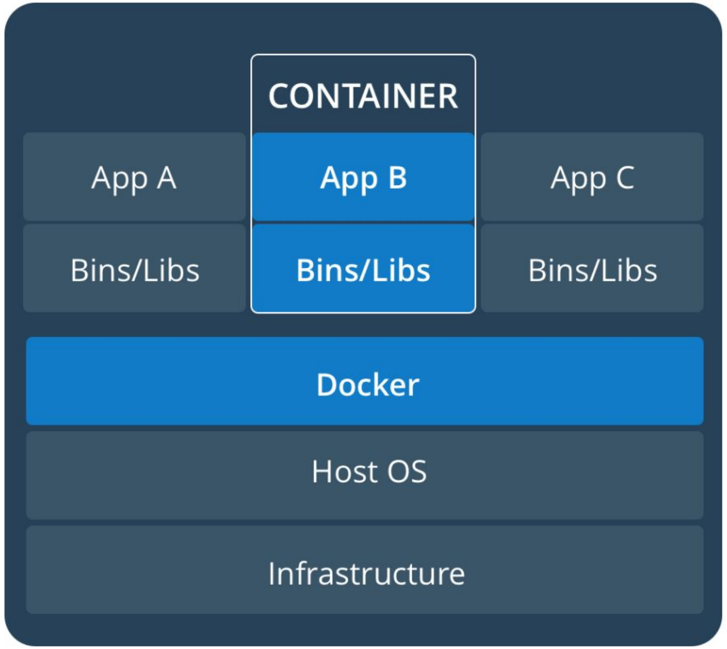
BrainUpgrade Academy

Virtual Machines and Containers

Virtual Machine diagram



Container diagram



Linux container & Docker

Linux Containers (LXC) (now windows too!)

OS level virtualization to provide isolation to a set of processes from rest of the system.

Features:

- Namespace: pid, net, ipc, mnt, uts
- Control Groups: cpu, memory, io, devices, network, freezer
- Union File System: aufs, btrfs, vfs, devicemapper
- Container format: libcontainer
- Security: AppArmor, Seccomp, Capabilities

Docker

Uses LXC to build, deploy & run apps with containers

Docker enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere.

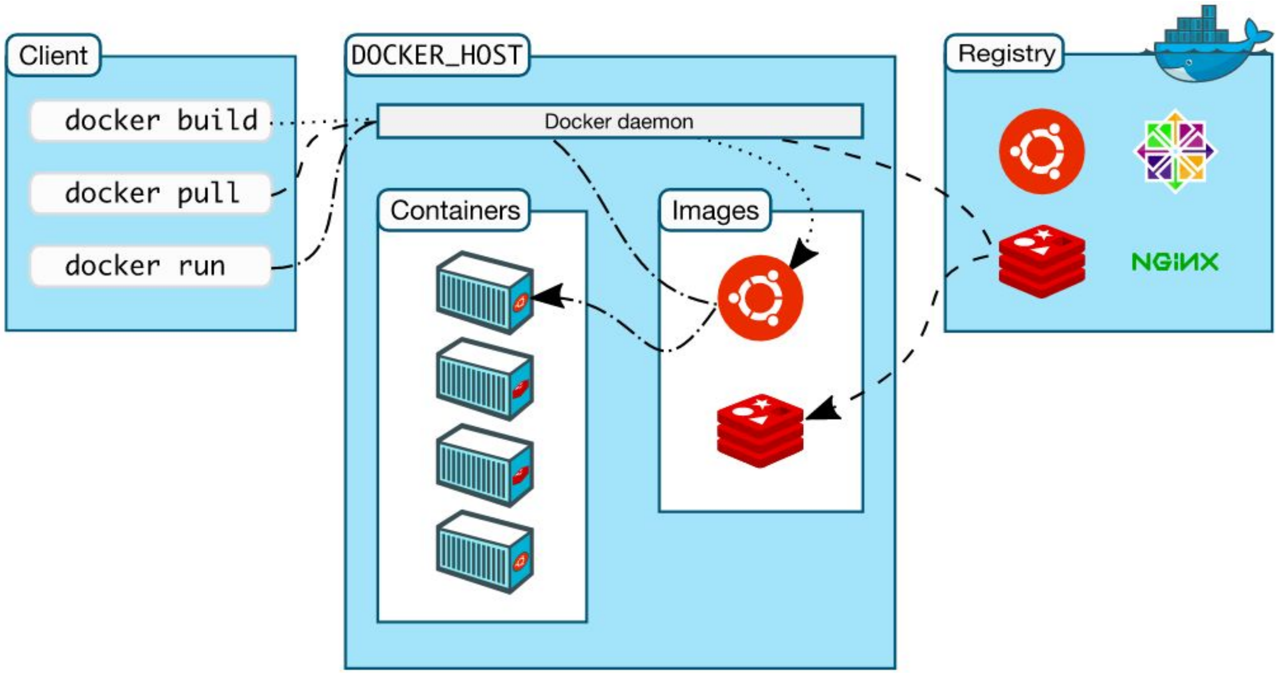
BrainUpgrade Academy

Terminologies

- **Image** - Executable package that includes everything needed to run an application – the code, a runtime, libraries, environment variables, and configuration files
- **Container** -
    - Runtime instance of an image–what the image becomes in memory when executed
- **Service** -
    - a container but service codifies the way image runs -replicas, port, name etc
- **Swarm** -
    - cluster of machines running docker containers
- **Stack** -
    - group of interrelated services that can be orchestrated and scaled together
- **Registry** -
    - storage and content delivery system, holding named Docker images, available in different tagged versions
- **Server Daemon** -
    - creates and manages docker objects - images, containers, network, volumes, swarm etc
- **Docker Client** -
    - CLI to communicate with server using Docker API
- **Docker REST API** -
    - Communication contract between docker component (servers & clients)
- **Network** -
    - Docker object holding the networking meta-data
- **Node** -
    - machine participating in Swarm
- **Volume** -
    - Storage of persistence data generated and managed by Docker containers

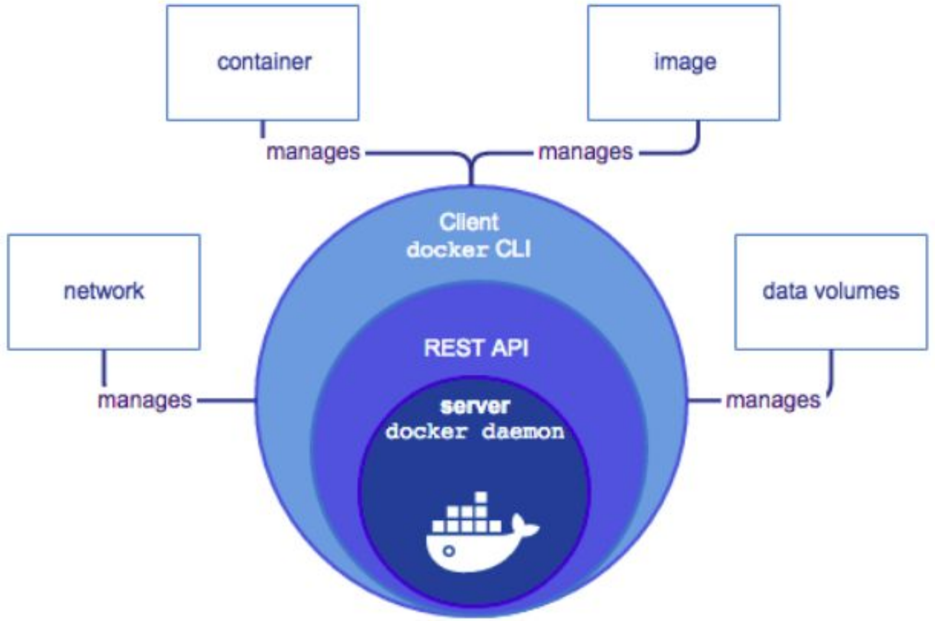
BrainUpgrade Academy

Docker Architecture

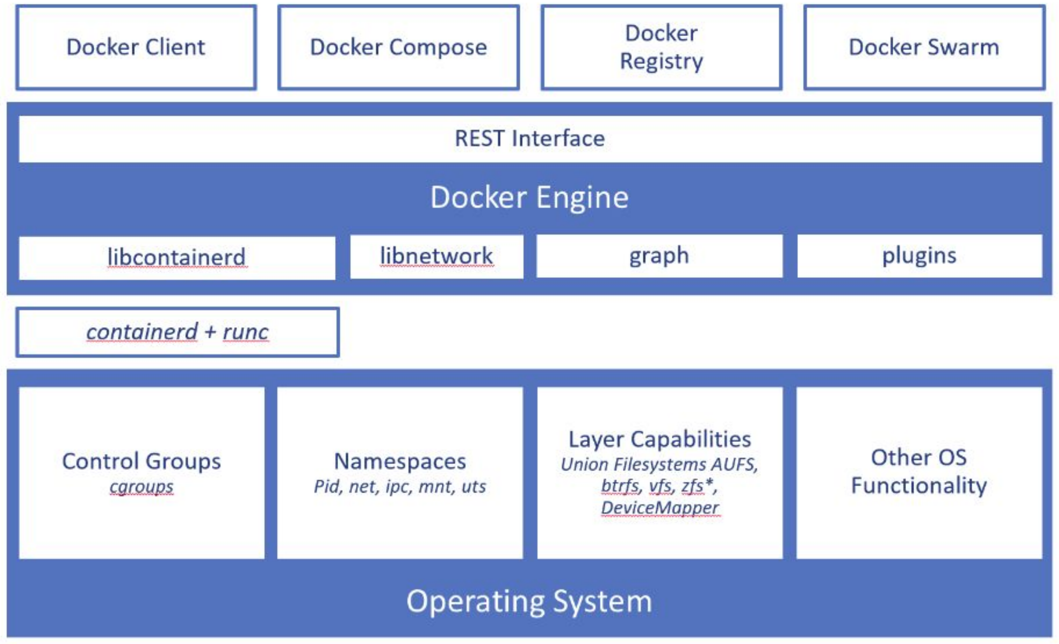


BrainUpgrade Academy

Docker Architecture

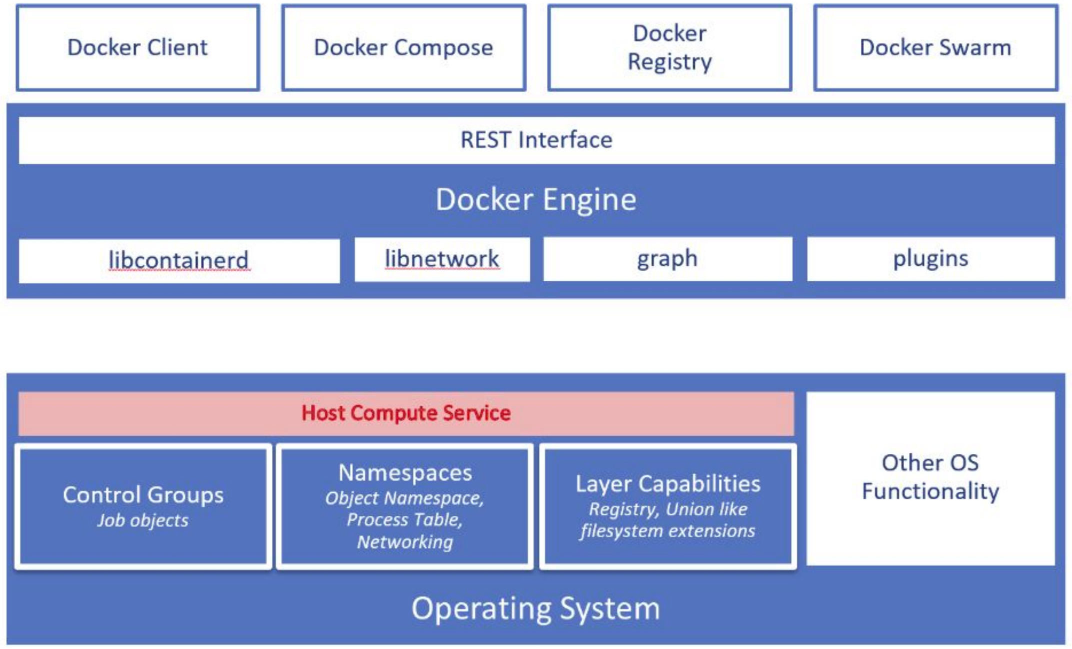


Docker Architecture - Linux



BrainUpgrade Academy

Docker Architecture - Windows



Ref: https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/containerd

Docker Setup (Ubuntu)

```
sudo apt-get update

sudo apt-get remove docker docker-engine docker.io

sudo apt install docker.io

sudo groupadd docker

sudo usermod -aG docker $USER

sudo systemctl start docker

sudo systemctl enable docker
```

Lab Exercises

Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments

BrainUpgrade Academy

Docker Containers

- Creating & Starting containers
- Running containers
- Docker Images
- Connecting containers
- Lab Exercises

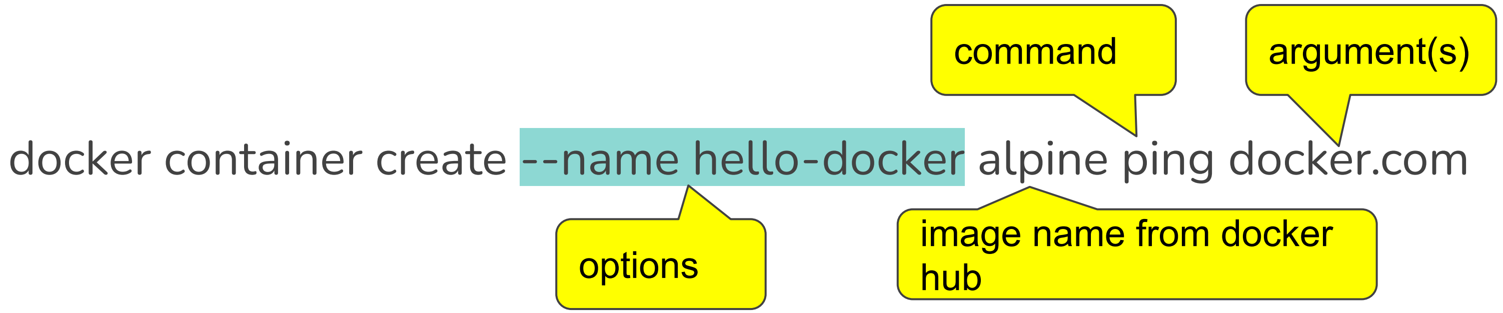
Creating containers

```
docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Options:

```
--name string name of the container
--cpus decimal number of CPUs
--label list set metadata on a container
--memory bytes memory limit
--network string connect container to a network (default “default”)
--publish list publish container’s port to the host
--rm remove container when it exits
-i interactive Keep STDIN open if not attached
-t allocates psuedo-TTY
```

Creating containers - Examples



```
docker container create --name hello-docker alpine ping docker.com
```

```
docker create --name busy -it busybox

docker container create --name alpine -it alpine sh

docker container create --name hello -p 80:80 tutum/hello-world
```

Starting containers

```
docker container start [OPTIONS] CONTAINER [CONTAINER...]
```

Options:

-i Attach container’s STDIN
-a Attach container’s STDOUT/STDERR and forward signals

Examples:

```
docker container start -ia busy
```

```
docker container start -ia alpine
```

```
docker container start hello
```

Running containers

docker container run [OPTIONS] IMAGE [COMMAND] [ARG...]

Options:

--name string name of the container
--cpus decimal number of CPUs
--label list set metadata on a container
--memory bytes memory limit
--network string connect container to a network (default “default”)
--publish list publish container’s port to the host
--rm remove container when it exits
-i interactive mode
-t allocates a pseudo-TTY

Running containers - Examples

```
docker container run -p 80:80 tutum/hello-world  (creates container with random name)

docker container run -p 80:80 nginx  (connects to tty, Ctrl+C to exit)

docker run -p 81:80 nginx   (shorthand command)

docker run --name ngx -p 80:80 -it nginx  (interactive terminal, Ctrl+PQ to leave it running)

docker attach ngx

docker run -d -p 80:80 --name nginx nginx  ( run in the background)

docker run -P --name nginx nginx  ( map exposed ports to random ports on the host)

docker run -d -p 8000-9000:80 nginx  (maps port 80 to any random port between 8000 to 9000 on host)

docker run --restart always -p 80:80 -it nginx
```

Running containers - Examples...

```
docker run -ti --rm r-base
```

```
docker run -it --rm -v /home/rajesh/git/twics-bu-20210419/docker/containers/hello-r/:/tmp r-base Rscript /tmp/main.R
```

```
docker run --name db -e MYSQL_ROOT_PASSWORD=docker -e MYSQL_DATABASE=docker -e MYSQL_USER=docker -e MYSQL_PASSWORD=docker -d mysql:5.6(provide environment params to the process)
```

```
docker run --link db:mysql -e spring.datasource.url=jdbc:mysql://mysql:3306/docker -p 8080:8080 -d    rajeshgheware/spring-db:1.0.0
```

```
docker run --log-opt max-size=20m --log-opt max-file=5 --link db:mysql -itd -p 8080:80 --name springdb --restart always  -v /tmp/docker/:/tmp/docker/ -e JAVA_OPTS='-Xms1g' -e java.security.egd=file:/dev/./urandom  -e spring.profiles.active=dev  -e spring.datasource.url=jdbc:mysql://mysql:3306/db  -e jasypt.encryptor.password=pwd -e security.oauth2.client.clientId=clientid -e security.oauth2.client.clientSecret=auth -e aws.accessKeyId=aa -e aws.secretKey=aa  -e server.port=80 rajeshgheware/spring-db:1.0.0
```

Docker Images

- **Image** - Executable package that includes everything needed to run an application – the code, a runtime, libraries, environment variables, and configuration files

- **docker images**
- **docker images nginx**
- **docker images java:8**
- **docker images --filter "dangling=true"** (untagged images)
- **docker rmi $(docker images -f "dangling=true" -q)**
- **docker search oracle (searches docker hub images having mention of oracle in it)**

Lab Exercises

Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments - Docker Container

Module 3: Provisioning Docker Images

- Introducing the Dockerfile
- Building images manually / Examples...
- Storing and retrieving Docker Images from Docker Hub
- Building images using Continuous Integration tools
- Inspecting a Dockerfile from DockerHub
- Lab Exercises

Introducing the Dockerfile

A Dockerfile is a text document that contains

- a set of instructions required to assemble the app (image) and/ run it

```
Usage:

docker build [OPTIONS] PATH | URL | -

Options:

      --add-host list           Add a custom host-to-IP mapping (host:ip)

      --compress                Compress the build context using gzip

      --cpu-quota int           Limit the CPU CFS (Completely Fair Scheduler) quota

  -f, --file string             Name of the Dockerfile (Default is 'PATH/Dockerfile')

      --force-rm                Always remove intermediate containers

      --label list              Set metadata for an image

  -m, --memory bytes            Memory limit

      --pull                    Always attempt to pull a newer version of the image

      --rm                      Remove intermediate containers after a successful build (default true)

  -t, --tag list                Name and optionally a tag in the 'name:tag' format
```

Introducing the Dockerfile

Example:

- docker build -f Dockerfile .

```
rajesh@rajesh-Gazelle:~/git/twics-bu-20210419/docker/images/simple$ cat Dockerfile
FROM alpine:latest

MAINTAINER info@brainupgrade.in
```

Introducing the Dockerfile

Example with tag:

- docker build -t myfirstimage .

Run the container using image name:

- docker run myfirstimage ping google.com

Introducing the Dockerfile

Few more variations:

- docker build -t myfirstimage -f Dockerfile .
- docker build -f /home/rajesh/git/twics-bu-20210419/docker/images/simple/Dockerfile-myfirstimage .
- docker build -t myfirstimage -f ./simple/Dockerfile ./simple/
- docker build -t myimage -t rajesh/myimage:1.0.0 -t localhost:5000/rajesh/myimage:1.0.0 .

Introducing the Dockerfile

- ENV - to set environment variables
- EXPOSE - to expose ports
- FROM - base image
- LABEL - to add metadata to image
- HEALTHCHECK - to check if container is running
- USER - to set user and group
- VOLUME - to specify mount point from external host
- WORKDIR - workdir to run any of the commands

Introducing the Dockerfile

- `ARG - variable used during build time`
- `CMD - to provide defaults to executing container`
- `RUN - to execute commands in new layer`
- `COPY - Copy file,dir or remote url to image`
- `ADD - Copy file,dir or remote url to image`
- `ENTRYPOINT - to configure container as executable`
- `MAINTAINER - the image maintainer`

RUN COPY ADD instructions create new layers in the image stack - refer layering section

Building Images (Alpine ping)

rajesh@rajesh-Gazelle:~/git/twics-bu-20210419/docker/images/simple-2$ cat Dockerfile

```
FROM alpine:latest

MAINTAINER info@brainupgrade.in

CMD ["ping","google.com"]
```

Build

- `docker build -t myalpine .`

Run

- `docker run myalpine`

Building Images (Ubuntu with utilities)

```
rajesh@rajesh-Gazelle:~/git/twics-bu-20210419/docker/images/simple-3$ cat Dockerfile
```

```
FROM ubuntu:latest

MAINTAINER info@brainupgrade.in

RUN apt-get update && apt-get install -y tree && apt-get install -y telnet && apt-get install -y curl
```

Build

- `docker build -t myubuntu .`

Run

- `docker run -it myubuntu`

Ref: https://hub.docker.com/_/ubuntu?tab=description

Building Images (Spring Boot)

rajesh@rajesh-Gazelle:~/git/rest-service$ cat Dockerfile

```
FROM openjdk:8-jre-alpine

MAINTAINER rajesh@unigps.in

COPY target/spring-db.jar app.jar

ENTRYPOINT ["/usr/bin/java", "-Djava.security.egd=file:/dev/./urandom", "-jar","app.jar"]
```

Build

- `Docker build -t rajeshgheware/spring-db:1.0.0 .`

Run

- `docker run --link db:mysql -e spring.datasource.url=jdbc:mysql://mysql:3306/docker -p 8080:8080 -d  rajeshgheware/spring-db:1.0.0`

Github Ref: https://github.com/rajeshgheware/rest-service

BrainUpgrade Academy

Building Images (Python)

rajesh@rajesh-Gazelle:~/git/twics-bu-20210419/docker/images/python$ cat Dockerfile

```
FROM python:2.7-slim
WORKDIR /app
ADD app.py /app
ADD requirements.txt /app
RUN pip install --trusted-host pypi.python.org -r requirements.txt
EXPOSE 80
ENV name world
CMD ["python","app.py"]
```

Build

- `docker build -t mypython .`

Run

- `docker run -p 80:80 mypython`

Dockerfile - Example (Apache)

```
FROM bitnami/minideb-extras:jessie-r23
LABEL maintainer "Bitnami <containers@bitnami.com>"

# Install required system packages and dependencies
RUN install_packages libapr1 libaprutil1 libc6 libexpat1 libffi6 libgmp10 libgnutls-deb0-28 libhogweed2 libldap-2.4-2 libnettle4
libp11-kit0 libpcre3 libsasl2-2 libssl1.0.0 libtasn1-6 libuuid1 zlib1g
RUN bitnami-pkg unpack apache-2.4.29-1 --checksum
42114e87aafb1d519ab33451b6836873bca125d78ce7423c5f7f1de4a7198596
RUN ln -sf /opt/bitnami/apache/htdocs /app

COPY rootfs /

ENV APACHE_HTTPS_PORT_NUMBER="443" \
    APACHE_HTTP_PORT_NUMBER="80" \
    BITNAMI_APP_NAME="apache" \
    BITNAMI_IMAGE_VERSION="2.4.29-r1" \
    PATH="/opt/bitnami/apache/bin:$PATH"

EXPOSE 80 443

WORKDIR /app
ENTRYPOINT ["/app-entrypoint.sh"]
CMD ["nami","start","--foreground","apache"]
```

Dockerfile - Example (Jenkins CI)

```
FROM jenkinsci/jenkins:latest
LABEL maintainer "r1co@post-box.cc"

USER root

# install docker cli
RUN mkdir -p /tmp/_install && cd /tmp/_install && wget https://get.docker.com/builds/Linux/x86_64/docker-latest.tgz  && tar -xvzf
docker-latest.tgz && cd docker && cp docker /usr/bin/docker && rm -rf  /tmp/_install
RUN chmod +x /usr/bin/docker
# add jenkins to docker group
RUN groupadd -g 999 docker
RUN usermod -a -G docker jenkins
# install docker-compose
RUN curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` >
/usr/local/bin/docker-compose
RUN chmod +x /usr/local/bin/docker-compose
USER jenkins
```

Dockerfile - Example (Multi stage)

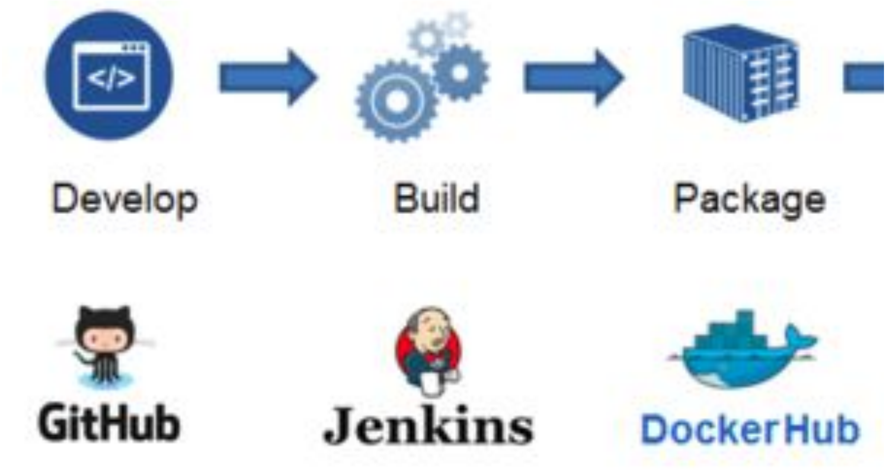
```
FROM golang:1.7.3 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go    .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

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

Docker Hub - store & retrieve

https://hub.docker.com (register and create login)

- docker tag alpine rajeshgheware/alpine:rajesh
- docker push rajeshgheware/alpine:rajesh
- docker pull rajeshgheware/alpine:rajesh

Build Image using CI / Jenkins



```
docker run --name jenkins -u 0 -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock -v $(which docker):$(which docker)
jenkins/jenkins:lts
```

Build Image - CI (Maven)

```
<profile>
        <id>docker</id>
        <build>
                <plugins>
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>dockerfile-maven-plugin</artifactId>
                                <version>1.3.6</version>
                                <executions>
                                        <execution>
                                                <id>default</id>
                                                <goals>
                                                        <goal>build</goal>
                                                        <goal>push</goal>
                                                </goals>
                                        </execution>
                                </executions>
                                <configuration>
                                        <repository>${docker.image.prefix}/${project.artifactId}</repository>
                                        <tag>${project.version}</tag>
                                        <buildArgs>
                                                <JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
                                        </buildArgs>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</profile>
```

BrainUpgrade Academy

Dockerfile References (Docker Hub)

- https://hub.docker.com/u/bitnami/
- https://hub.docker.com/_/ubuntu?tab=description
- https://github.com/docker-library/cassandra
- https://hub.docker.com/r/sebp/elk/~/dockerfile/

Lab Exercises

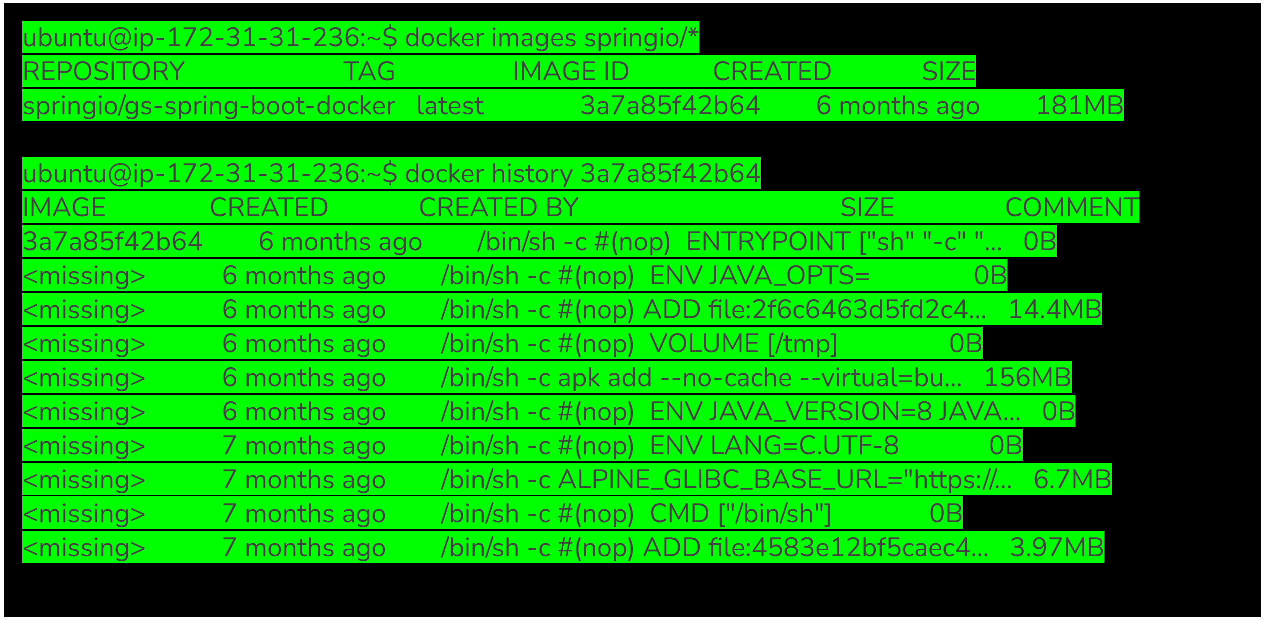
Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments

Module 4: Diving deeper - Dockerfile

- Dockerfile and Layers
- The Build cache
- The ENTRYPOINT Instruction
- The CMD Instruction Docker
- The ENV Instruction
- Volumes and the VOLUME Instruction
- Lab Exercises

Dockerfile & Layers

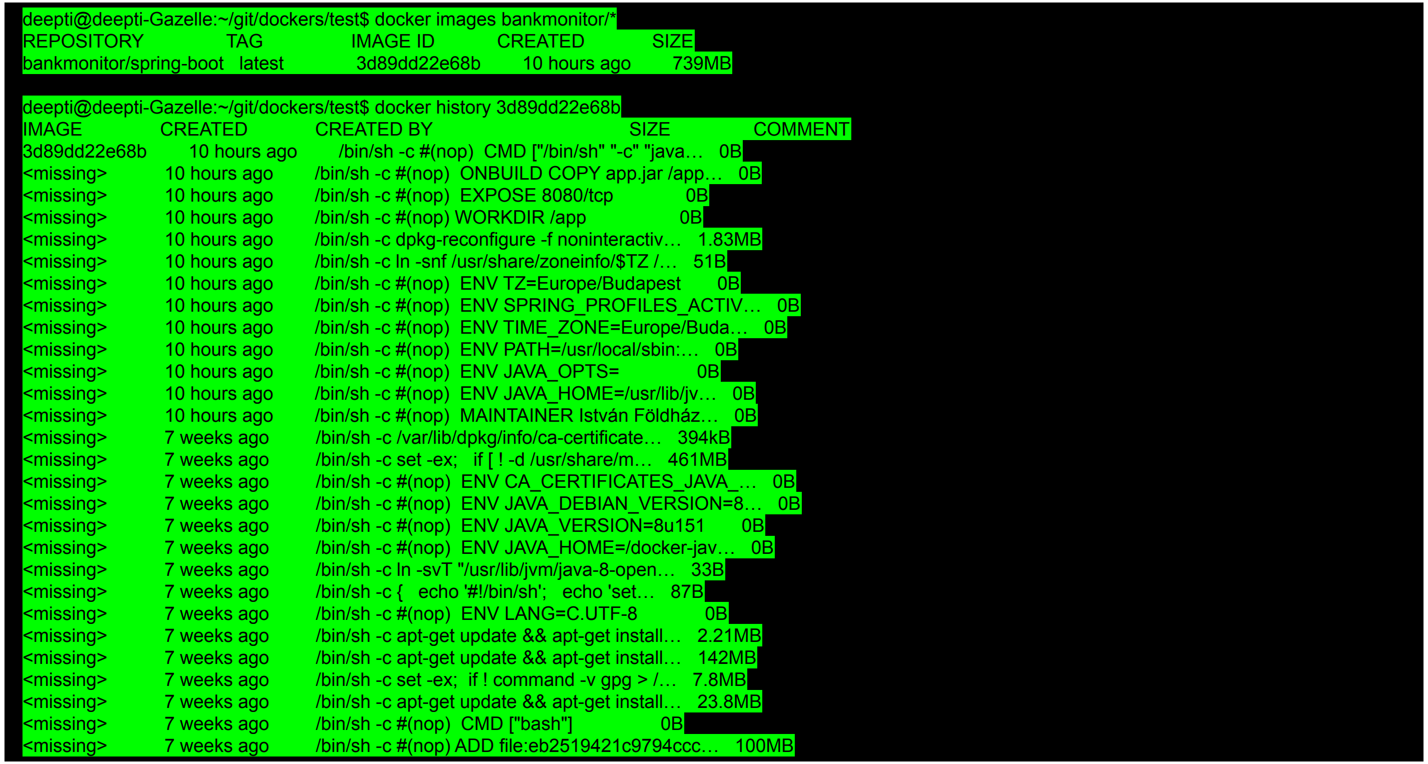


```
ubuntu@ip-172-31-31-236:~$ docker images springio/*
REPOSITORY                    TAG           IMAGE ID          CREATED           SIZE
springio/gs-spring-boot-docker   latest          3a7a85f42b64       6 months ago        181MB

ubuntu@ip-172-31-31-236:~$ docker history 3a7a85f42b64
IMAGE              CREATED            CREATED BY                                        SIZE              COMMENT
3a7a85f42b64        6 months ago        /bin/sh -c #(nop)  ENTRYPOINT ["sh" "-c" "...   0B
<missing>          6 months ago        /bin/sh -c #(nop)  ENV JAVA_OPTS=              0B
<missing>          6 months ago        /bin/sh -c #(nop) ADD file:2f6c6463d5fd2c4...   14.4MB
<missing>          6 months ago        /bin/sh -c #(nop)  VOLUME [/tmp]               0B
<missing>          6 months ago        /bin/sh -c apk add --no-cache --virtual=bu...   156MB
<missing>          6 months ago        /bin/sh -c #(nop)  ENV JAVA_VERSION=8 JAVA...   0B
<missing>          7 months ago        /bin/sh -c #(nop)  ENV LANG=C.UTF-8            0B
<missing>          7 months ago        /bin/sh -c ALPINE_GLIBC_BASE_URL="https://...   6.7MB
<missing>          7 months ago        /bin/sh -c #(nop)  CMD ["/bin/sh"]             0B
<missing>          7 months ago        /bin/sh -c #(nop) ADD file:4583e12bf5caec4...   3.97MB
```

Dockerfile & Layers

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE
ADD ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

Dockerfile & Layers



```
deepti@deepti-Gazelle:~/git/dockers/test$ docker images bankmonitor/*
REPOSITORY               TAG               IMAGE ID          CREATED           SIZE
bankmonitor/spring-boot   latest            3d89dd22e68b       10 hours ago       739MB

deepti@deepti-Gazelle:~/git/dockers/test$ docker history 3d89dd22e68b
IMAGE             CREATED             CREATED BY                                      SIZE              COMMENT
3d89dd22e68b       10 hours ago        /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "java…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ONBUILD COPY app.jar /app…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  EXPOSE 8080/tcp              0B
<missing>         10 hours ago        /bin/sh -c #(nop) WORKDIR /app                  0B
<missing>         10 hours ago        /bin/sh -c dpkg-reconfigure -f noninteractiv…   1.83MB
<missing>         10 hours ago        /bin/sh -c ln -snf /usr/share/zoneinfo/$TZ /…   51B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV TZ=Europe/Budapest       0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV SPRING_PROFILES_ACTIV…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV TIME_ZONE=Europe/Buda…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV PATH=/usr/local/sbin:…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV JAVA_OPTS=               0B
<missing>         10 hours ago        /bin/sh -c #(nop)  ENV JAVA_HOME=/usr/lib/jv…   0B
<missing>         10 hours ago        /bin/sh -c #(nop)  MAINTAINER István Földház…   0B
<missing>         7 weeks ago         /bin/sh -c /var/lib/dpkg/info/ca-certificate…   394kB
<missing>         7 weeks ago         /bin/sh -c set -ex;   if [ ! -d /usr/share/m…   461MB
<missing>         7 weeks ago         /bin/sh -c #(nop)  ENV CA_CERTIFICATES_JAVA_…   0B
<missing>         7 weeks ago         /bin/sh -c #(nop)  ENV JAVA_DEBIAN_VERSION=8…   0B
<missing>         7 weeks ago         /bin/sh -c #(nop)  ENV JAVA_VERSION=8u151       0B
<missing>         7 weeks ago         /bin/sh -c #(nop)  ENV JAVA_HOME=/docker-jav…   0B
<missing>         7 weeks ago         /bin/sh -c ln -svT "/usr/lib/jvm/java-8-open…   33B
<missing>         7 weeks ago         /bin/sh -c {   echo '#!/bin/sh';   echo 'set…   87B
<missing>         7 weeks ago         /bin/sh -c #(nop)  ENV LANG=C.UTF-8             0B
<missing>         7 weeks ago         /bin/sh -c apt-get update && apt-get install…   2.21MB
<missing>         7 weeks ago         /bin/sh -c apt-get update && apt-get install…   142MB
<missing>         7 weeks ago         /bin/sh -c set -ex;  if ! command -v gpg > /…   7.8MB
<missing>         7 weeks ago         /bin/sh -c apt-get update && apt-get install…   23.8MB
<missing>         7 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>         7 weeks ago         /bin/sh -c #(nop) ADD file:eb2519421c9794ccc…   100MB
```

Dockerfile & Layers

```
FROM openjdk:8-jdk
MAINTAINER István Földházi <istvan.foldhazi@gmail.com>

ENV JAVA_HOME               /usr/lib/jvm/java-8-openjdk-amd64
ENV JAVA_OPTS               ""
ENV PATH                    $PATH:$JAVA_HOME/bin

ENV TIME_ZONE               Europe/Budapest
ENV SPRING_PROFILES_ACTIVE test

ENV TZ=$TIME_ZONE
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
RUN dpkg-reconfigure -f noninteractive tzdata

WORKDIR /app

EXPOSE 8080

COPY app.war /app/app.war

CMD ["/bin/sh", "-c", "java $JAVA_OPTS -jar /app/app.war --spring.profiles.active=$SPRING_PROFILES_ACTIVE"]
```

```
/bin/sh -c set -ex;   if [ ! -d /usr/share/man/man1 ]; then   mkdir -p /usr/share/man/man1;  fi;   apt-get update;  apt-get install -y   openjdk-8-jdk="$JAVA_DEBIAN_VERSION"   ca-certificates-java="$CA_CERTIFICATES_JAVA_VERSION"  ;  rm -rf
/var/lib/apt/lists/*;   [ "$(readlink -f "$JAVA_HOME")" = "$(docker-java-home)" ];   update-alternatives --get-selections | awk -v home="$(readlink -f "$JAVA_HOME")" 'index($3, home) == 1 { $2 = "manual"; print | "update-alternatives --set-selections" }';
update-alternatives --query java | grep -q 'Status: manual'   461MB
```

Build Cache

Why Layers & Cache?

- To identify similar portions of content by componentizing image
- To avoid downloading similar content thus reduce network traffic
- To build images faster by reusing parts which were created earlier

BrainUpgrade Academy

The ENTRYPOINT instruction

To configure a container that will run as an executable

Two forms:

- `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* form, preferred)
- `ENTRYPOINT command param1 param2` (*shell* form)

Notes:

- Container run arguments will be appended to the above
- Override using docker run --entrypoint flag
- Last ENTRYPOINT will have effect
- CMD / Container run arguments will make executable NOT receive UNIX signal like SIGTERM (when run in shell form)
- Shell form ignores CMD / docker run arguments

Examples:

- `ENTRYPOINT ["top", "-b"]`
- `ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]`
- `ENTRYPOINT [ "sh", "-c", "echo $HOME" ]`
- `ENTRYPOINT exec top -b`

The CMD instruction

To provide defaults for an executing container

Three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

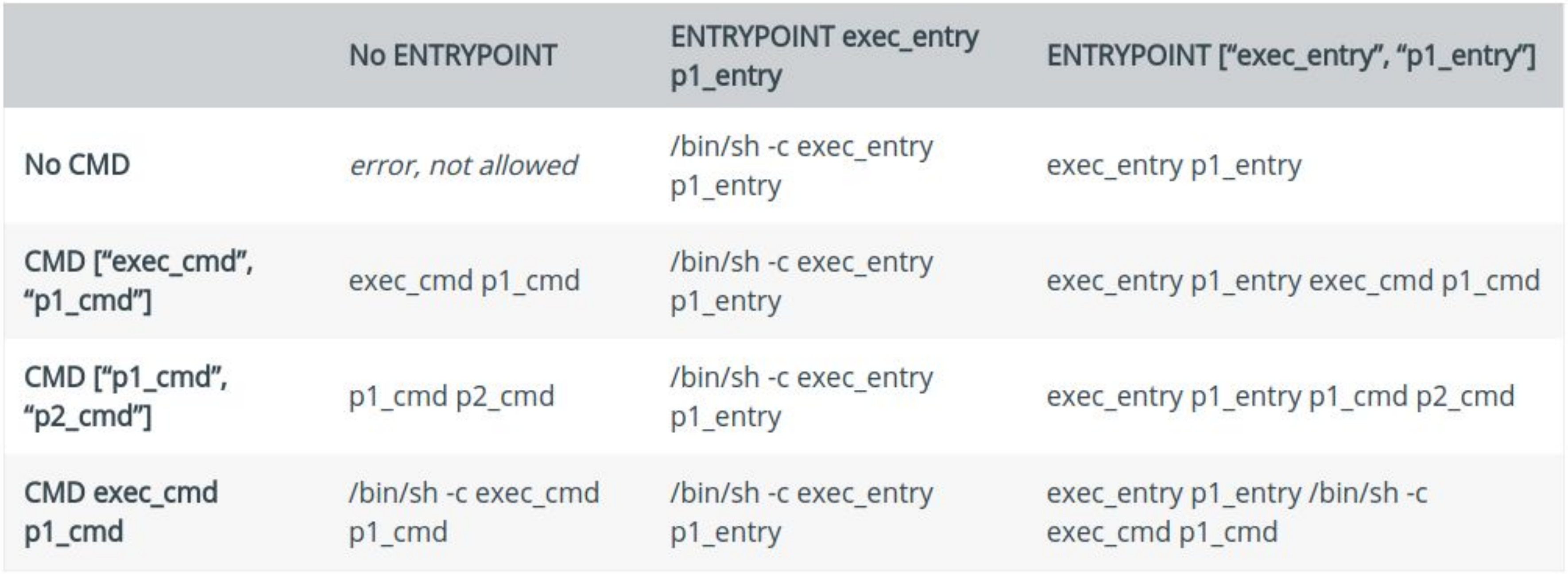
Notes:

- Only the last CMD taken into account per Dockerfile
- If executable not specified, then ENTRYPOINT must
- Differs from RUN as RUN is executed at container build time and results committed to image
- No shell is used for non-shell form so do not use env variable in non-shell form
- Container run arguments override CMD arguments

Examples:

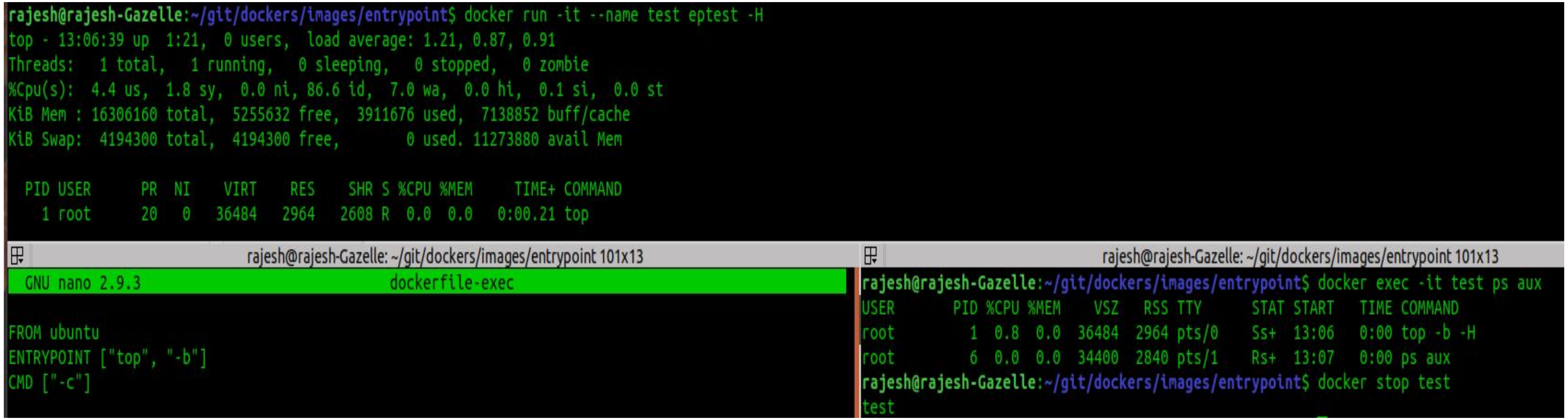
- CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
- CMD ["rails", "server"]
- CMD npm start
- CMD ["mvn", "clean", "install", "-D skip.unit.tests=true"]
- CMD /usr/sbin/sshd -D
- CMD ["bash", "-c", "(while true; do echo '.'; sleep 60; done) & tox"]
- CMD ["java", "Main"]
- CMD [`"sh", "-c", "echo $HOME"`]

ENTRYPOINT & CMD



	No ENTRYPOINT	ENTRYPOINT exec_entry p1_entry	ENTRYPOINT [“exec_entry”, “p1_entry”]
No CMD	*error, not allowed*	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry
CMD [“exec_cmd”, “p1_cmd”]	exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry exec_cmd p1_cmd
CMD [“p1_cmd”, “p2_cmd”]	p1_cmd p2_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry p1_cmd p2_cmd
CMD exec_cmd p1_cmd	/bin/sh -c exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry /bin/sh -c exec_cmd p1_cmd

exec - Example



```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker run -it --name test eptest -H
top - 13:06:39 up  1:21,  0 users,  load average: 1.21, 0.87, 0.91
Threads:   1 total,   1 running,   0 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.4 us,  1.8 sy,  0.0 ni, 86.6 id,  7.0 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 16306160 total,  5255632 free,  3911676 used,  7138852 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used. 11273880 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0   36484   2964   2608 R  0.0  0.0   0:00.21 top
```

rajesh@rajesh-Gazelle: ~/git/dockers/images/entrypoint 101x13

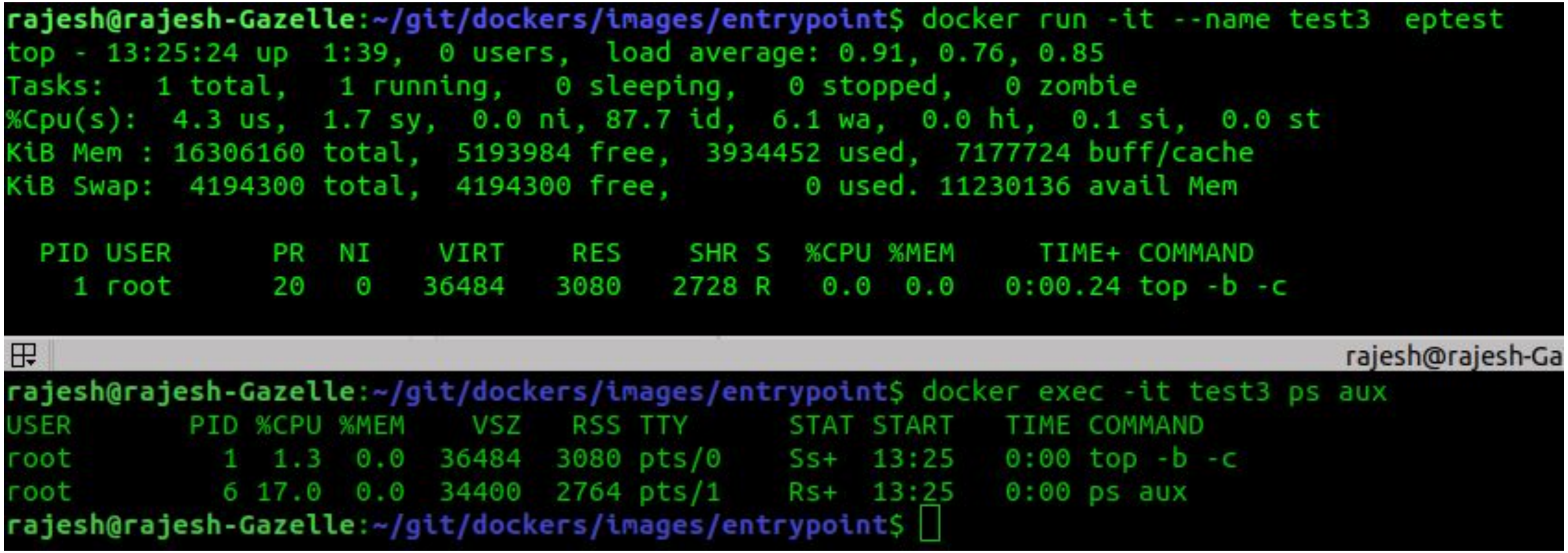
```
  GNU nano 2.9.3                    dockerfile-exec

FROM ubuntu
ENTRYPOINT ["top", "-b"]
CMD ["-c"]
```

rajesh@rajesh-Gazelle: ~/git/dockers/images/entrypoint 101x13

```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker exec -it test ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.8  0.0  36484  2964 pts/0    Ss+  13:06   0:00 top -b -H
root         6  0.0  0.0  34400  2840 pts/1    Rs+  13:07   0:00 ps aux
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker stop test
test
```

Container run arguments suppress CMD arguments

Exec - Example



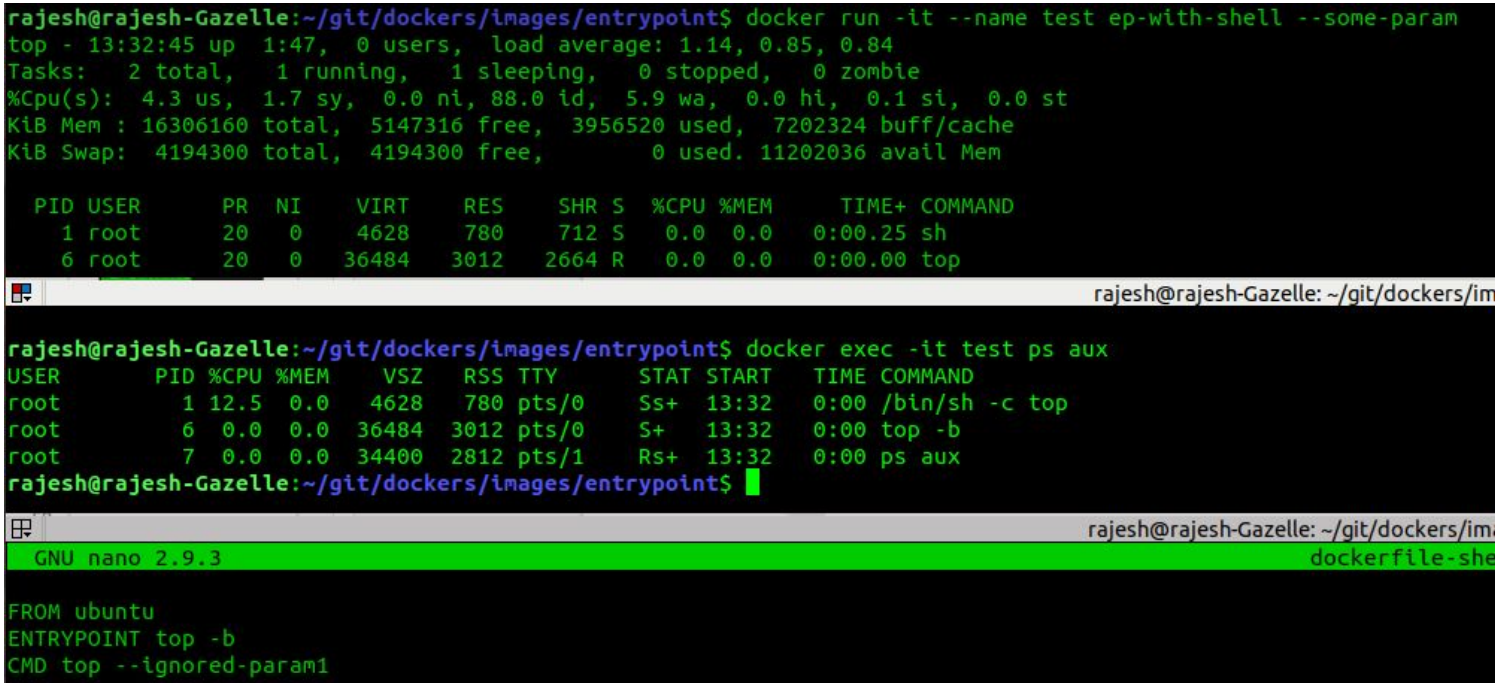
```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker run -it --name test3  eptest
top - 13:25:24 up  1:39,  0 users,  load average: 0.91, 0.76, 0.85
Tasks:   1 total,   1 running,   0 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.3 us,  1.7 sy,  0.0 ni, 87.7 id,  6.1 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 16306160 total,  5193984 free,  3934452 used,  7177724 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used. 11230136 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0   36484   3080   2728 R   0.0  0.0   0:00.24 top -b -c
```

rajesh@rajesh-Ga

```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker exec -it test3 ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  1.3  0.0  36484  3080 pts/0    Ss+  13:25   0:00 top -b -c
root         6 17.0  0.0  34400  2764 pts/1    Rs+  13:25   0:00 ps aux
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ 
```

CMD arguments appended to the ENTRYPOINT when no argument to docker run

Shell - Example



```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker run -it --name test ep-with-shell --some-param
top - 13:32:45 up  1:47,  0 users,  load average: 1.14, 0.85, 0.84
Tasks:   2 total,   1 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.3 us,  1.7 sy,  0.0 ni, 88.0 id,  5.9 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 16306160 total,  5147316 free,  3956520 used,  7202324 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used. 11202036 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0    4628    780    712 S   0.0  0.0   0:00.25 sh
    6 root      20   0   36484   3012   2664 R   0.0  0.0   0:00.00 top
```

rajesh@rajesh-Gazelle: ~/git/dockers/im

```
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$ docker exec -it test ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1 12.5  0.0   4628   780 pts/0    Ss+  13:32   0:00 /bin/sh -c top
root         6  0.0  0.0  36484  3012 pts/0    S+   13:32   0:00 top -b
root         7  0.0  0.0  34400  2812 pts/1    Rs+  13:32   0:00 ps aux
rajesh@rajesh-Gazelle:~/git/dockers/images/entrypoint$
```

rajesh@rajesh-Gazelle: ~/git/dockers/im

```
  GNU nano 2.9.3                                                                          dockerfile-she

FROM ubuntu
ENTRYPOINT top -b
CMD top --ignored-param1
```

When in shell form then CMD as well as docker run arguments

The ENV instruction

To set environment variable <key> to the <value>

Two forms:

- `ENV key value`
- `ENV key=value`

Notes:

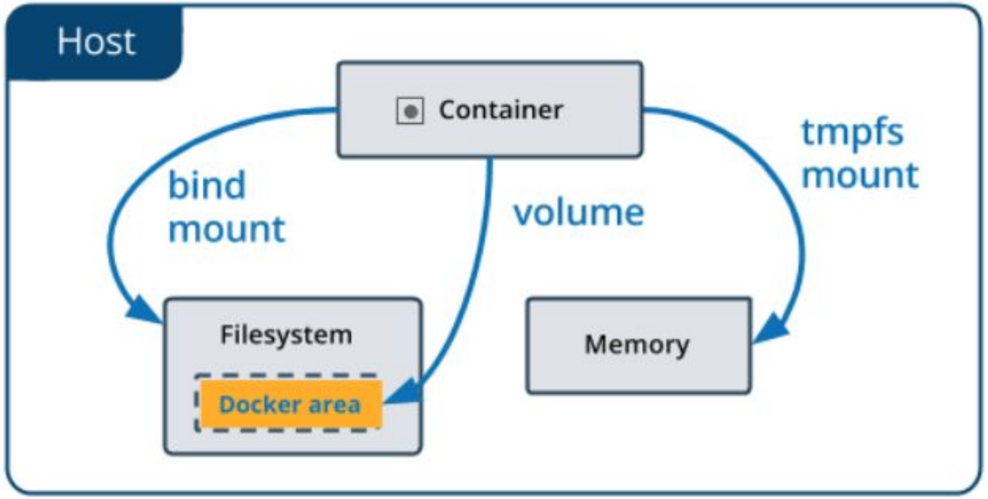
- Override using docker run --env flag
- Extremely useful in planning & executing deployments

Examples:

- `ENV myName=rajesh g`
- `ENV org unigps`
- `ENV CN IN`
- `ENV environment dev uat`
- `ENV myName=”rajesh g” org=unigps CN=IN`
- `ENV REST_ARCHIVE=rust-1.21.0-x86_64-unknown-linux-gnu.tar.gz`
- `ENV REST_DOWNLOAD_URL=https://static.rust-lang.org/dist/$RUST_ARCHIVE`
- `ENV PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/.cargo/bin"`
- `JENKINS_HOME="/data/jenkins"`

The VOLUME - Data Persistence

Storage of persistence data generated by managed by Docker containers



Commands:

- `docker volume create my-vol`
- `docker volume ls`
- `docker volume inspect my-vol`
- `docker volume rm my-vol`

VOLUME - Examples

Examples (volume): Persist data in a container’s writeable layer

- `docker run -d --name devtest --mount source=app,target=/app nginx:latest`
- `docker service create -d --replicas 4 --mount source=app,target=/app nginx:latest`

Examples (bind volume): a file or directory on the *host machine* is mounted into a container. Performant but not-reliable

- `docker run -d -it --name devtest --mount type= bind,source="$(pwd)",target=/app nginx:latest`
- `docker run -d -it --name devtest --mount type=bind,source="$(pwd)",target=/app,readonly  nginx:latest`

Examples (tmfs volume): For temporary sensitive data to be kept only in memory

- `docker run -d -it --name tmptest --mount type=tmpfs,destination=/app nginx:latest`

VOLUME - preferred way

- Volumes are easier to back up or migrate than bind mounts.
- You can manage volumes using Docker CLI commands or the Docker API.
- Volumes work on both Linux and Windows containers.
- Volumes can be more safely shared among multiple containers.
- Volume drivers allow you to store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
- A new volume's contents can be pre-populated by a container.

Lab Exercises

Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments

Module 5: Working with Registry

- Overview
- Creating a Public repo on Docker Hub
- Using our Public repo on Docker Hub
- Using a Private Registry
- Docker Enterprise
- Lab Exercises

Overview - Registry

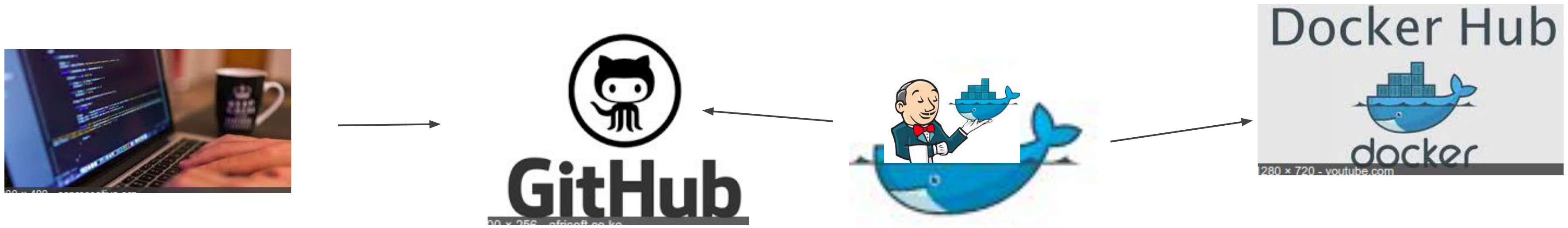
Registry

Stateless, highly scalable server side application that stores and lets you distribute Docker images.

When to use

- tightly control where your images are being stored
- fully own your images distribution pipeline
- integrate image storage and distribution tightly into your in-house development workflow

Dockerizing dev workflow



```
docker run --name jenkins -u 0 -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock -v $(which docker):$(which docker)
jenkins/jenkins:lts
```

Notes:

Add docker pipeline jenkins plugin to work

Test project: https://github.com/brainupgrade-in/nodejsappdocker.git

Add jenkins credential having ID **docker-hub-credentials** for docker hub push access

Registry Server

- With no docker volume (uses default volume for container)
  - `docker run -d -p 5000:5000 --name registry registry:2`
  - `docker push localhost:5000/rajesh/alpine:test`
  - `Docker pull localhost:5000/rajesh/alpine:test`
- With docker volume
  - `docker volume create docker_registry`
  - `docker run -d -p 5000:5000 -v docker_registry:/var/lib/registry --name registry registry:2`
  - `docker container stop registry && docker container rm -v registry`
- With Volume Mount on Host
  - `docker run -d -p 5000:5000 -v /root/docker_registry:/var/lib/registry --name registry registry:2`

Mount host FS

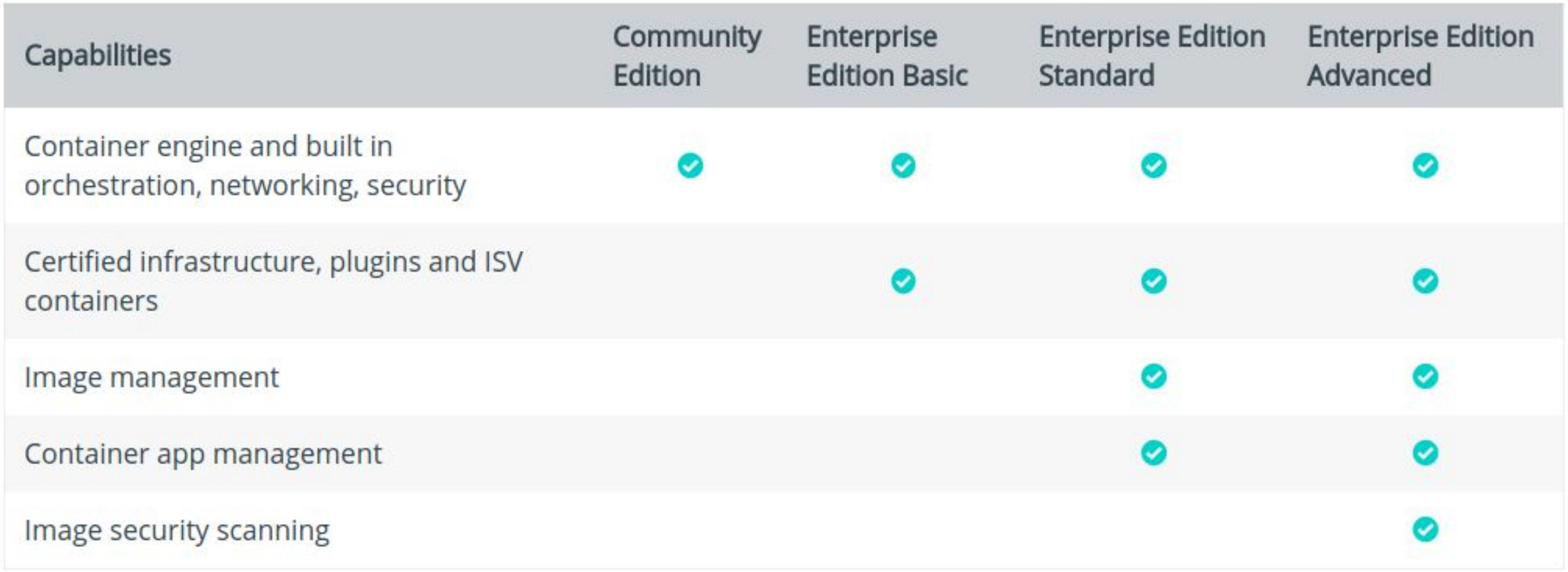
Case One

```
docker container run -ti -v /tmp:/data alpine sh
```

Case Two (faster development with debugging)

```
docker container run -d -p 8080:80--mount type=bind,source="$(pwd)",target=/usr/share/nginx/html  nginx:latest
```

Docker Enterprise



Capabilities	Community Edition	Enterprise Edition Basic	Enterprise Edition Standard	Enterprise Edition Advanced
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Certified infrastructure, plugins and ISV containers		✓	✓	✓
Image management			✓	✓
Container app management			✓	✓
Image security scanning				✓

Lab Exercises

Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments

BrainUpgrade Academy

Module 6: Docker Networking

- Overview
- The docker0 Bridge
- User Defined Network
- Exposing Ports
- Viewing Exposed Ports
- Linking Containers
- Lab Exercises

Overview - Networking

Defines how containers communicate with external world, amongst cluster members etc

Two types of networks:

- Default
- Custom Defined

Default:

- Bridge - docker0 (docker created default network) Configurable
- Host - container on host network stack Not configurable
- None - container specific network stack (no network interface) Not configurable

Custom Defined Network: User specific network rules using underlying iptables

Notes:

- Change container network(s) on the fly
- First non internal network is the main external connectivity interface

The docker0 bridge

- Containers default network is docker0
- Container inter-connectivity using IP addresses (no name resolution)
- For name resolution, legacy --link feature available for limited period
- Change default bridge to none using --network flag or daemon.json server config

User Defined Network

To control which containers can communicate with each other

Automatic DNS resolution of container names to IP addresses (DNS 127.0.0.11)

Create unlimited networks

Types

- Bridge Network
- Overlay Network
- MACVLAN Network

User Defined Network - bridge

bridge

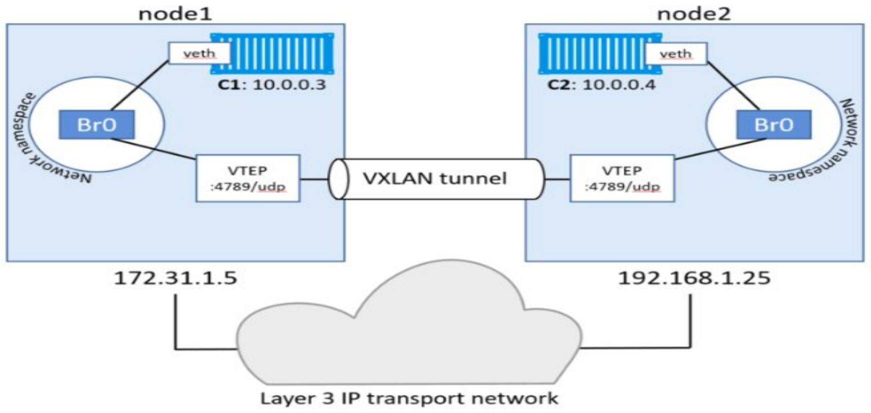
- Most common type of network in Docker world
- Good for small network

docker_gwbridge

- Docker created network for communication among swarm nodes
- Provides external connectivity when none of the networks provide

Overlay Network

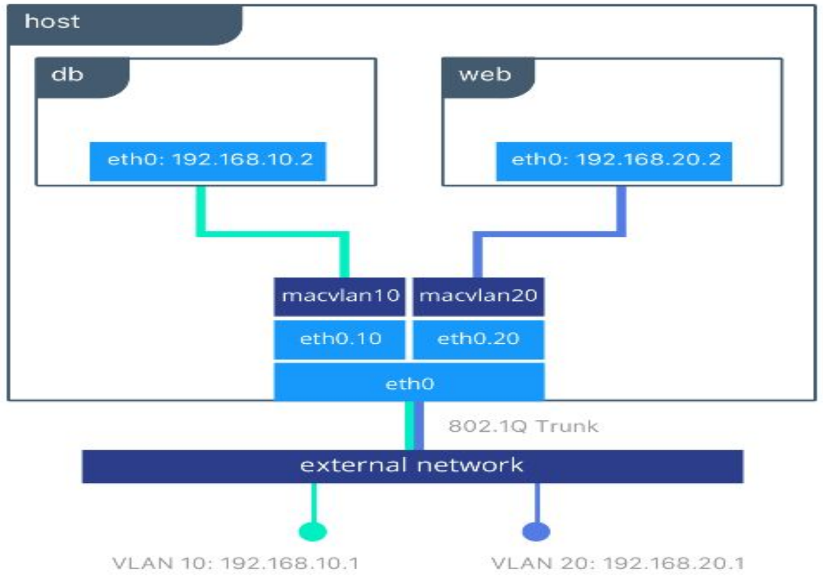
- Scope is swarm mode
- Provided to service tasks in swarm cluster
- Only for swarm nodes and not for standalone containers else require key-value store (Zookeeper, Consul etc)
- Uses NAT and port mapping (iptables)



BrainUpgrade Academy

MACVLAN Network

- Provides better control over IPv4 and IPv6 addressing
- Extremely lightweight & highly performant
- Attached to Docker Host directly
- Stricter dependency between localhost and external network
- Does not use linux bridge or port mapping
- Scope is outside swarm



Test Setup - 1

Test Setup:

Create custom network n1

- docker network create n1

Create two busybox containers attached to n1

- docker run -itd --name c1 --network n1 busybox
- docker run -itd --name c2 --network n1 busybox

Tests

- Log into c1 and ping c2 (should succeed)
    - docker exec -it c1 sh
    - ping c2
- Log into c2 and ping c1 (should succeed)
    - docker exec -it c2 sh
    - ping c1

Test Setup - 2

Prerequisites: Test Setup -1

Test Setup:
Remove network from both containers c1 & c2

- docker network disconnect n1 c1
- docker network disconnect n1 c2

Tests:

- Login into c1 and ping c2 (should fail)
  - docker exec -it c1 sh
  - ping c2
- Login into c1 and ping google.com (should fail)
  - docker exec -it c1 sh
  - ping google.com
- Run ifconfig on c1 to see interfaces (should see only loopback interface)
  - docker exec -it c1 sh
  - ifconfig
- Do the same on c2 (results should be similar)

Test Setup - 3

Test Setup:

- Create four networks n1, n2, n3, n4
  - docker network n1
  - docker network n2
  - docker network n3
  - docker network n4
- Create four containers c1 (n1), c2 (n2), c3 (n3), c4 (n4) associated with denoted network
  - docker run -itd --name c1 --network n1 busybox
  - docker run -itd --name c2 --network n2 busybox
  - docker run -itd --name c3 --network n3 busybox
  - docker run -itd --name c4 --network n4 busybox
- Create n23 network and connect c2 and c3 with it
- docker network n23
- docker network connect n23 c2
- docker network connect n23 c3

Tests:

- Login into c2 and ping c3 (should succeed)
  - docker exec -it c2 sh
  - ping c3
- Login into c3 and ping c4 (should fail)
  - docker exec -it c3 sh
  - ping c4

Test Setup - 4

Test Setup:

- Create container c5 with host network
  docker run -itd --name c5 --network host busybox

Tests:

- Run ifconfig on c5 as well as docker host (networks listed should be same)
  - docker exec -it c5 sh
  - ifconfig
- Disconnect c5 from host (operation should fail)
  - docker network disconnect host c5

Lab Exercises

Please refer google classwork: https://classroom.google.com/w/MzE2MjM4Njg1NDM1/t/all

And do all the lab work as per the instructions noted in classwork assignments

Misc - Dockerization steps

- docker run --name demo-mysql -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=demo -e MYSQL_USER=demo_user -e MYSQL_PASSWORD=demo_pass -d mysql:5.6

- docker run -p 8080:8080 -e spring.profiles.active=prod -e spring.datasource.url=jdbc:mysql://mysql:3306/demo -e spring.datasource.username=demo_user -e spring.datasource.password=demo_pass --link demo-mysql:mysql --name spa -itd -v logs:/logs rajeshgheware/spa-sboot-docker:1.3.0

- docker run -p 5601:5601 -p 9200:9200 -p 5044:5044   -e ES_HEAP_SIZE="2g" -e LS_HEAP_SIZE="1g" --name elk -v /tmp/elastic_search:/var/lib/elasticsearch/nodes -v /tmp/elastic_search/logs:/logs -itd sebp/elk (requires to set sudo sysctl -w vm.max_map_count=262144)

Misc - Logstash config for java

```
root@0c415fec6fb4:/etc/logstash/conf.d# cat logstash-spring.conf
input {
  stdin {}
  file {
    path =>  [ "/logs/spa-boot-docker/server-rolling.log" ]
  }
}
filter {
   multiline {
      pattern => "^(%{TIMESTAMP_ISO8601})"
      negate => true
      what => "previous"
   }
   grok {
      # Do multiline matching with (?m) as the above mutliline filter may add newlines to the log messages.
      match => [ "message", "(?m)^%{TIMESTAMP_ISO8601:logtime}%{SPACE}%{LOGLEVEL:loglevel}
%{SPACE}%{NUMBER:pid}%{SPACE}%{SYSLOG5424SD:threadname}%{SPACE}---%{SPACE}%{JAVACLASSSHORT:classname}%{SPACE}:%{SPA
CE}%{GREEDYDATA:logmessage}" ]
   }
}
output {
  elasticsearch { host => "localhost" }
```

Restart logstash agent:

BrainUpgrade Academy

Misc - K8S - Docker

	Docker	Kubernetes
Scheduling Unit	Container	Pod
Scaling	Service	ReplicaSet
Rolling Updates	Service	Deployment
Load Balancer, DNS	Service	Service
Cluster Manager	Swarm	Deployment

Thank You for your active participation!

Please join gheWARE cluster

(community of brainlets sharing brainware to help upgrade each other)

rajesh@unigps.in

9880195215

https://www.linkedin.com/in/rajesh-gheware/