Docker

Objective

After finishing this chapter students will have overview of Docker architecture. Should be able to build and run a Docker Image.

Agenda

- What is Docker
- Why Docker
- Containerization
- Containers Vs VM’s
- Docker Architecture
- Concepts
- Dockerfile
- Lab

What is Docker?

What is Docker?

- Docker is an open source platform which makes application development, packaging and running easier by help of containers.
- Docker is like a VM(Virtual machine). But instead of creating a whole operating system it uses linux kernel. The application developer then just have to package and ship the application with rest of the libraries and dependencies.

Why Docker ?

This Application works on my system but not on QA environment?

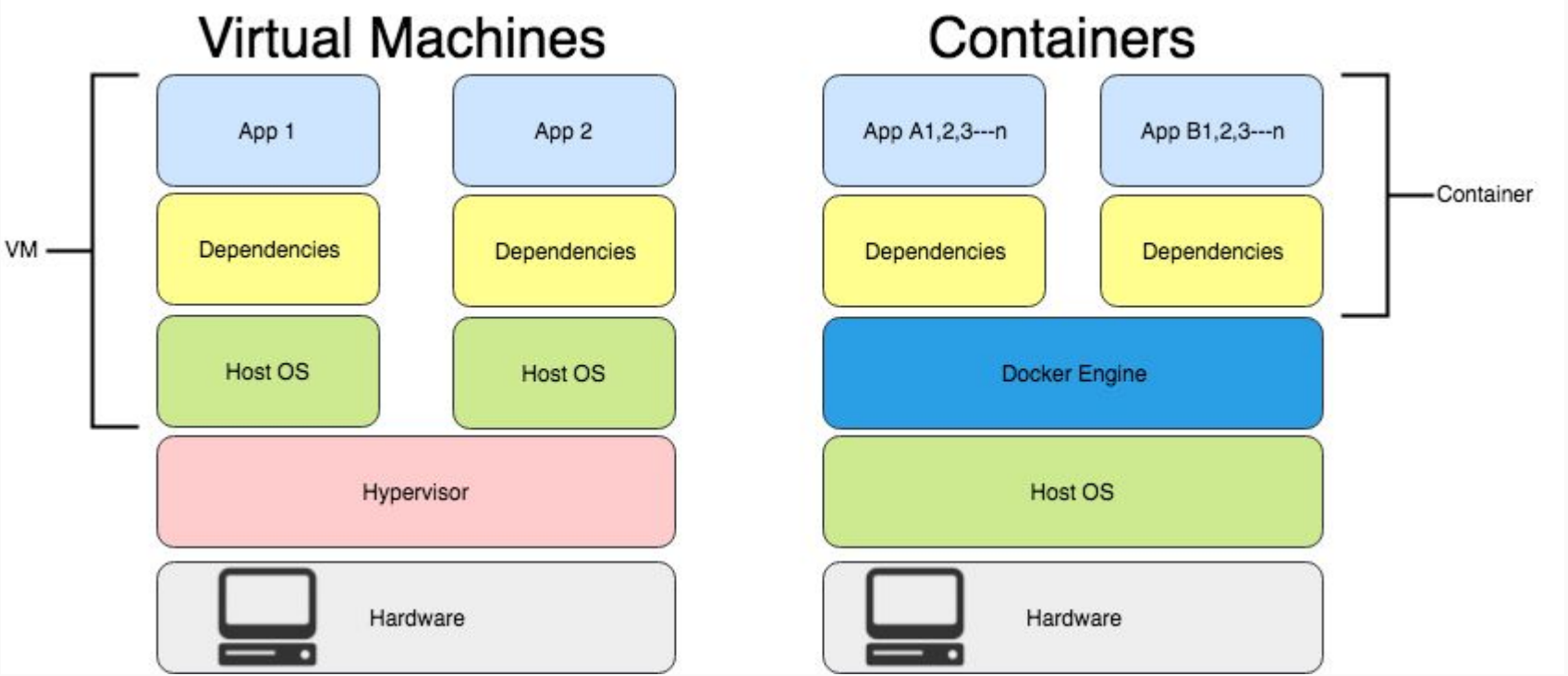
Containerization

Containerization

- Lightweight OS-level virtualization.
- Enables app developers to deploy and run distributed applications without launching an entire virtual machine (VM) for each app.
- Multiple isolated applications can run on a single host and access the same OS kernel.

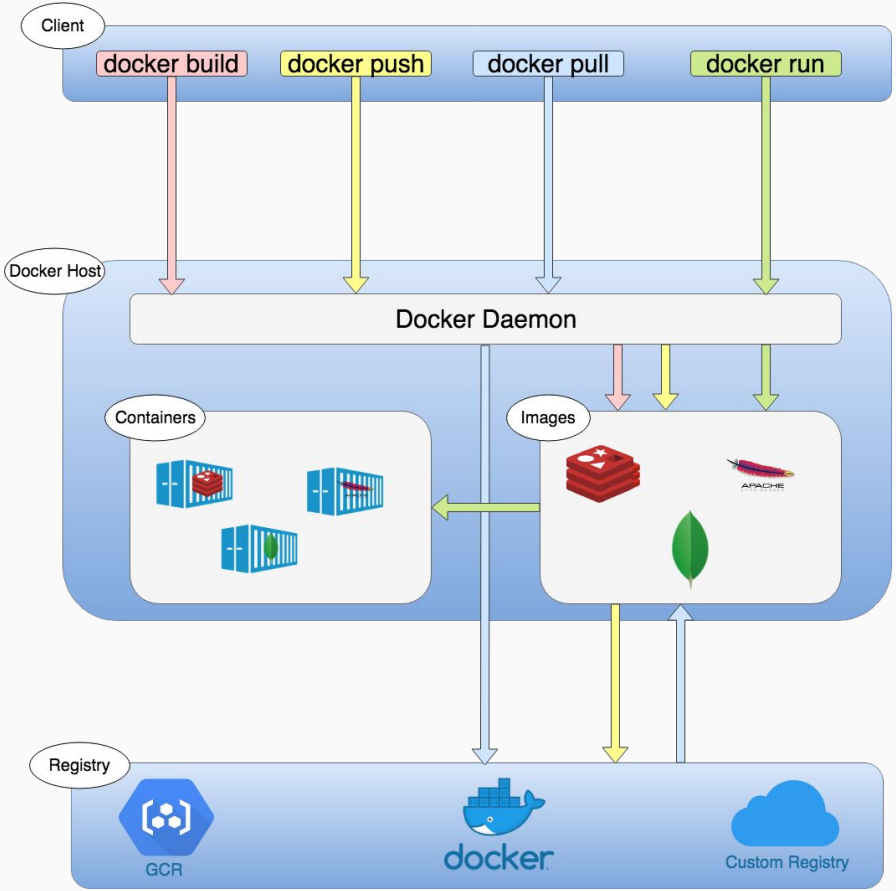
Containers Vs VMs

Containers Vs VMs



Docker Architecture

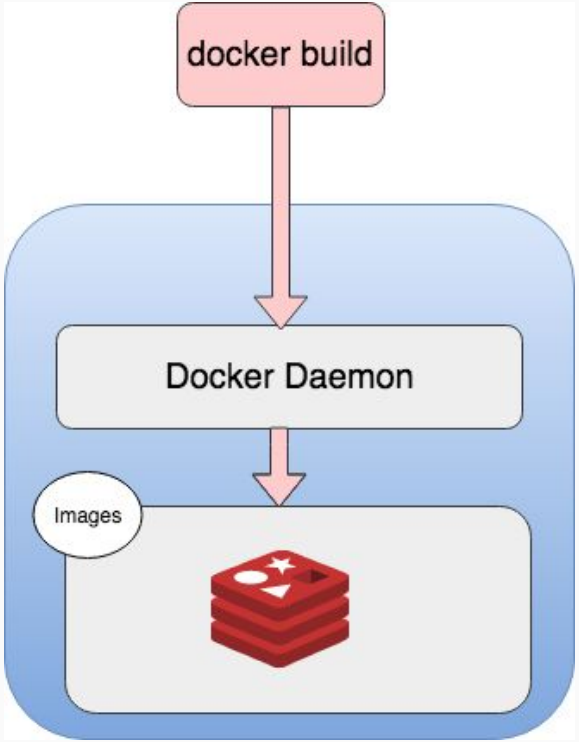
Docker Architechture



Concepts

- Images
- Containers
- Docker Daemon
- Docker Client
- Registry

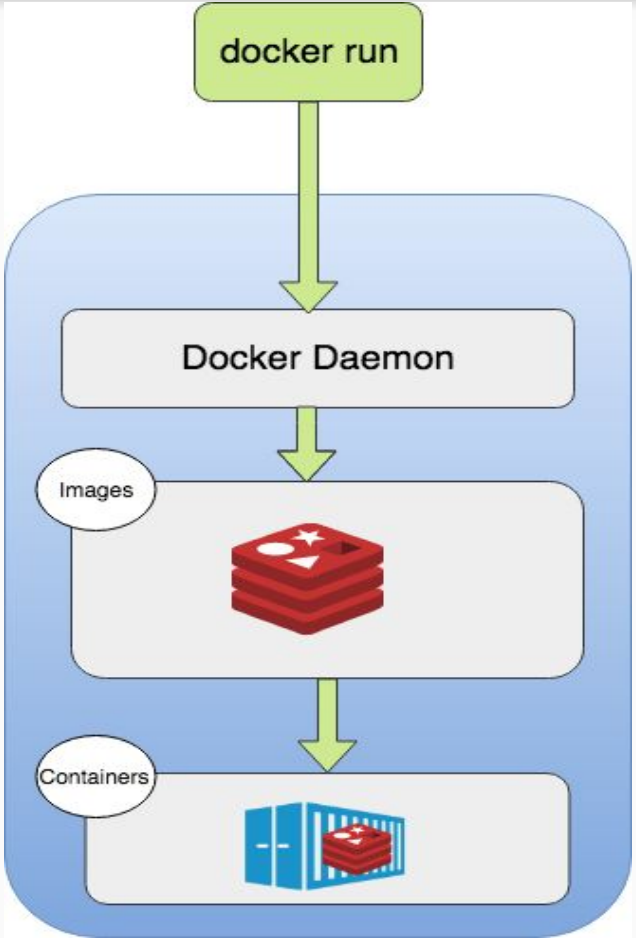
Concepts: Images



Concepts: Images

1. **Base images** are images that have no parent image, usually images with an OS like ubuntu, busybox or debian.
2. **Child images** are images that build on base images and add additional functionality.
3. **Official images** are images that are officially maintained and supported by the folks at Docker. These are typically one word long.
4. **User images** are images created and shared by users like you and me. They build on base images and add additional functionality. Typically, these are formatted as user/image-name.

Concepts: Containers



Concepts: Docker Daemon

Docker is a Client-Server architecture. Docker daemon is a background running service which is used by client to build, run and distribute docker containers.

Concepts: Docker Client

Docker client is a CLI which is used to interact to docker daemon.

Concepts: Docker Registry

Docker registry is where all the images are stored. These can be Docker Hub, GCR, ECS or Custom Registry. Developers and other corporations publish their images so that others can pull these images if they need it.

Dockerfile

Dockerfile

1. Its a text file which defines an image environment, dependencies and set of instruction when a container is created.
2. The name of the file is “Dockerfile”

```
# A basic apache server. To use either add or bind mount content under /var/www
FROM ubuntu:12.04

RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Dockerfile: Commands

- FROM
- RUN
- CMD
- LABEL
- EXPOSE
- ENV
- ADD
- COPY
- ENTRYPOINT
- VOLUME
- USER
- WORKDIR
- ARG
- ONBUILD
- STOPSIGNAL
- HEALTHCHECK
- SHELL

Dockerfile: Commands

FROM : Sets the base image for rest of the instructions in the Dockerfile
USAGE: FROM <image name>
EXAMPLE: FROM ubuntu

RUN : It takes a command as an argument. It is used to build the image.
USAGE: RUN <command>
EXAMPLE: RUN apt-get install -y apache2

CMD : It takes a command as argument but unlike RUN this command is executed when the container is instantiated.
USAGE: CMD <command>
EXAMPLE: CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]

Dockerfile: Commands

ENV : Sets the environment variables. These are key value pairs which can be used by Containers, Applications and Scripts.
USAGE: ENV <key> <value>
EXAMPLE: ENV APACHE_RUN_USER www-data

EXPOSE : If your application uses a specific PORT access you can use this command to achieve that.
USAGE: EXPOSE <port>
EXAMPLE: EXPOSE 8080

USER : Sets the user which need to run a container.
USAGE: USER <UID>
EXAMPLE: USER 152

Dockerfile: Commands

ADD : Copies files from source file system or URL to container file system.
USAGE: ADD <source> <destination>
EXAMPLE: ADD /src/app /home/src/app

LABEL : Adds metadata to image.
USAGE: LABEL <key>=<value>
EXAMPLE: LABEL “version”=”1.0”

WORKDIR : Sets the path where CMD would execute its command.
USAGE: WORKDIR <path>
EXAMPLE: WORKDIR /bin

Dockerfile: Commands

ENTRYPOINT : Default command which needs to run after container is instantiated. If you have entrypoint set then CMD can be used to pass just the args to entrypoint command.
USAGE: ENTRYPOINT <cmd>
EXAMPLE: ENTRYPOINT echo

Lab

Download and install Docker from below links:

- Docker for Windows
- Docker For MAC
- Other OS

Run Docker

- Run the installed Docker application.
- You will see the Whale icon on the status bar.

Docker Version

- Open terminal on mac.
- Run command “docker version”.

```
$docker version
Client:
 Version:      17.06.1-ce
 API version:  1.30
 Go version:   go1.8.3
 Git commit:   874a737
 Built:        Thu Aug 17 22:53:38 2017
 OS/Arch:      darwin/amd64

Server:
 Version:      17.06.1-ce
 API version:  1.30 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   874a737
 Built:        Thu Aug 17 22:54:55 2017
 OS/Arch:      linux/amd64
 Experimental: true
```

Test Docker Installation

- Run command “docker run hello-world”.

```
$docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:97ce6fa4b6cdc0790cda65fe7290b74cfebd9fa0c9b8c38e979330d547d22ce1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

$
```

First Image Dockerfile

- Create a separate folder for our first image.
- Create a file in your favorite text editor.
- Name it Dockerfile.
- Add below lines to the Dockerfile and save.

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y nginx
ENTRYPOINT [“/usr/sbin/nginx”,”-g”,”daemon off;”]
EXPOSE 8080
```

First Image: build

Run Command “docker build -t first-nginx-image:latest .”

```
$docker build -t first-nginx-image:latest .
Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM ubuntu
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
 ---> f975c5035748
Step 2/5 : RUN apt-get update
 ---> Running in ca5689bc447f

Step 3/5 : RUN apt-get install -y nginx
 ---> Running in 5e4e44e91ae2
Reading package lists...
Building dependency tree...
Reading state information...

Step 4/5 : ENTRYPOINT [“/usr/sbin/nginx”,”-g”,”daemon off;”]
 ---> Running in 6cb785069a66
 ---> 3257ae8c77e1
Removing intermediate container 6cb785069a66
Step 5/5 : EXPOSE 80
 ---> Running in d9c7fce2aace
 ---> e125077aedc5
Removing intermediate container d9c7fce2aace
Successfully built e125077aedc5
Successfully tagged first-nginx-image:latest
```

First Image: build

- Run Command “docker images”
- You should be able to see the image you just created.

```
$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
first-nginx-image       latest              e125077aedc5        20 minutes ago      209MB
```

First Image: Run

- Run Command “docker run -d -p 80:80 --name nginx-webserver first-nginx-image”
- Go to web browser and go to http://localhost:80
- You should see default page of nginx.

First Image: Push

- Run Command "**$ docker login --username=hub-user**" in ur terminal.
- List all the images using command "**$ docker images**"

```
$docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
first-nginx-image       latest              68421a785a78        4 days ago          209MB
```

- Copy the image id we will use it to tag
- Run command "**$ docker tag 68421a785a78 username/nginx:version1.0**"
- To push to repository on docker hub run command "**$ docker push username/nginx**"

Congratulations on Building, Running and Pushing your first Docker Image

Exercise

Create a docker image of spring boot application and tag it with a version and push it to docker hub.