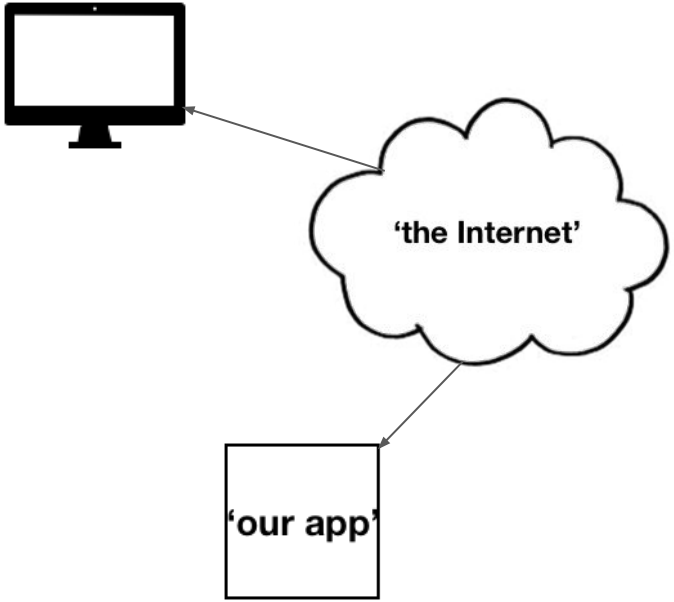
Containerizing (Dockerizing) your application

Maddie Patrichi: ioana-madalina.patrichi@oracle.com
Andrea Rosa: andrea.rosa@oracle.com
@reclaro (twitter)

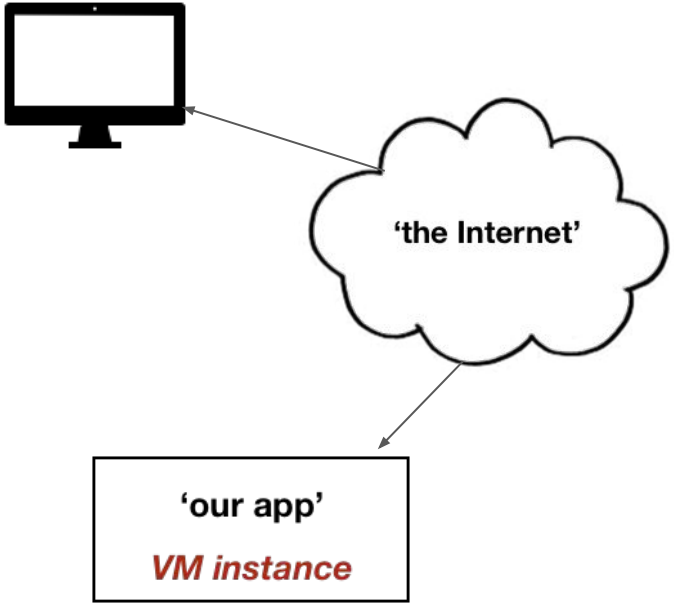
The Goal

Let’s move our application into a Docker container.



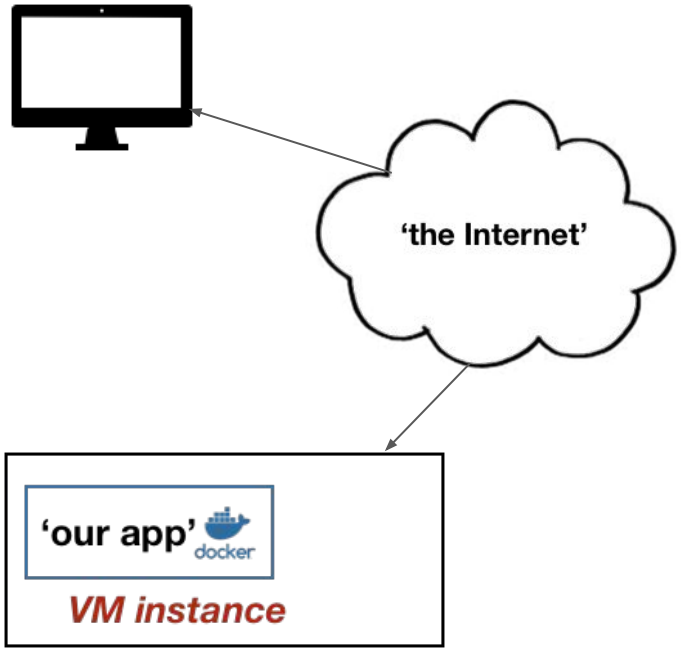
The Goal

Last lecture, we had our app deployed in a VM in OCI.



The Goal

To make our app more flexible, we'll containerize it within a deployed VM.



Tutorial Overview

1. Containers
2. Docker overview
3. Develop with Docker
4. Use multi-staged builds
5. Manage application data
6. Using Docker in the real world
7. Container orchestration with Docker

Ideal lifecycle of delivering an application

Existing Application

Convert to a **Container** with Docker EE

Modern Infrastructure
Built on premise, in the cloud, or as part of a hybrid environment

Modern Methodologies
Integrate to CI/CD and other automation systems

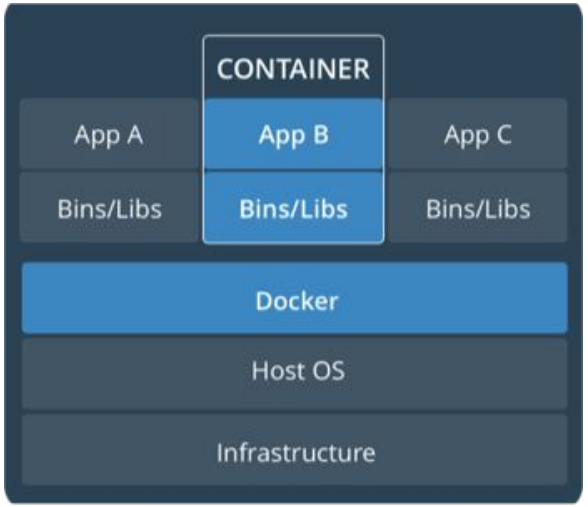
Modern Microservices
Add new services or start peeling off services from monolith code base

Containers

- Application, not machine centric view of the world
- Containers as a common format for software delivery
- Immutable unit of delivery
- Decouple applications from infrastructure
- Cloud and OS portability

Containers vs VMs

Container layout



VM layout

In depth analysis: https://www.youtube.com/watch?v=L1ie8negCjc

Containers vs VMs

Important distinction:

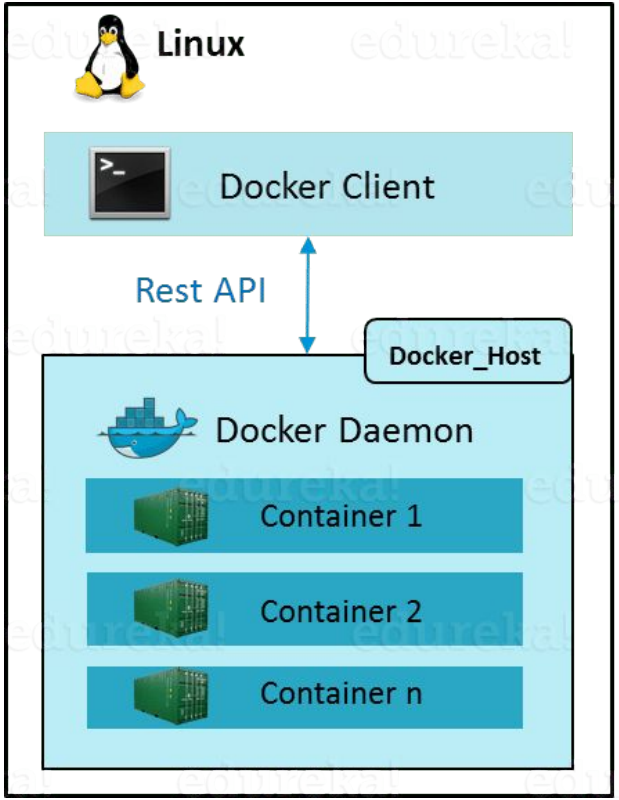
OS = kernel + filesystem/libraries

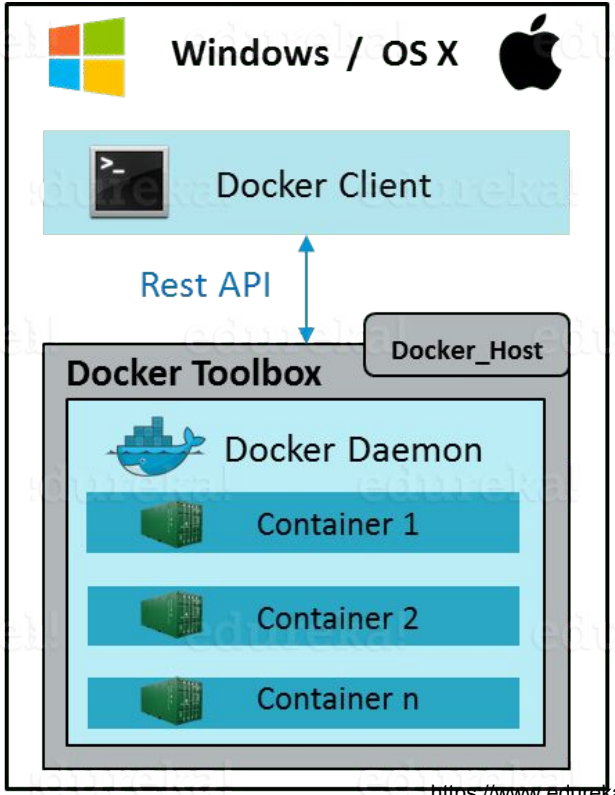
Image = filesystem/libraries

Containers run on the **same** kernel, but on **different/same** images.

In depth analysis: https://www.youtube.com/watch?v=L1ie8negCjc

Docker Components





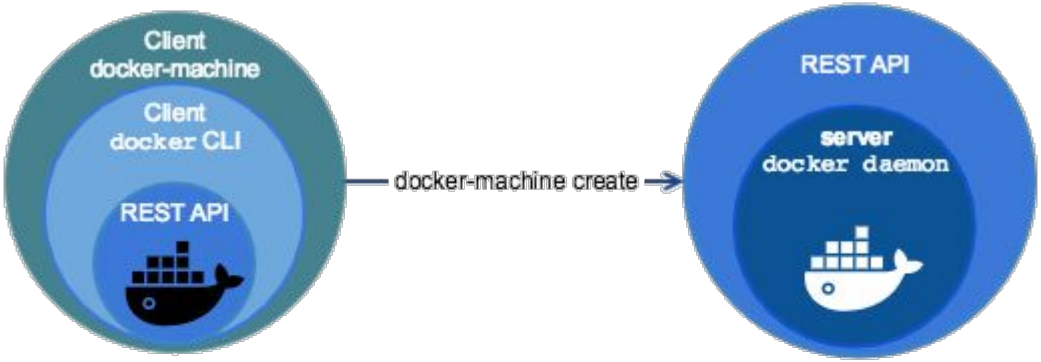
edureka!

Docker Toolbox includes:

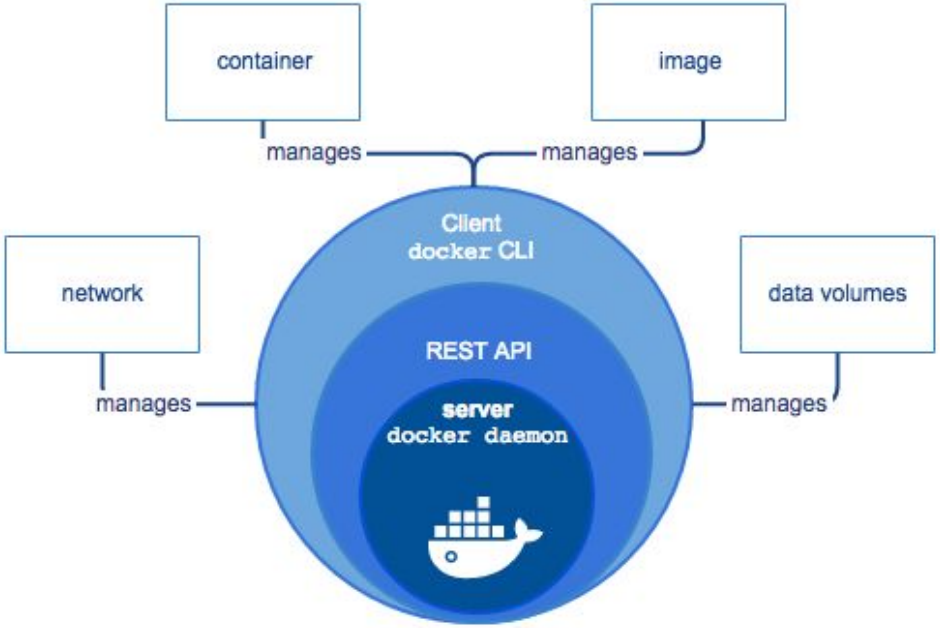
- Docker Client
- Compose(Mac only)
- Kitematic
- Machine and
- VirtualBox

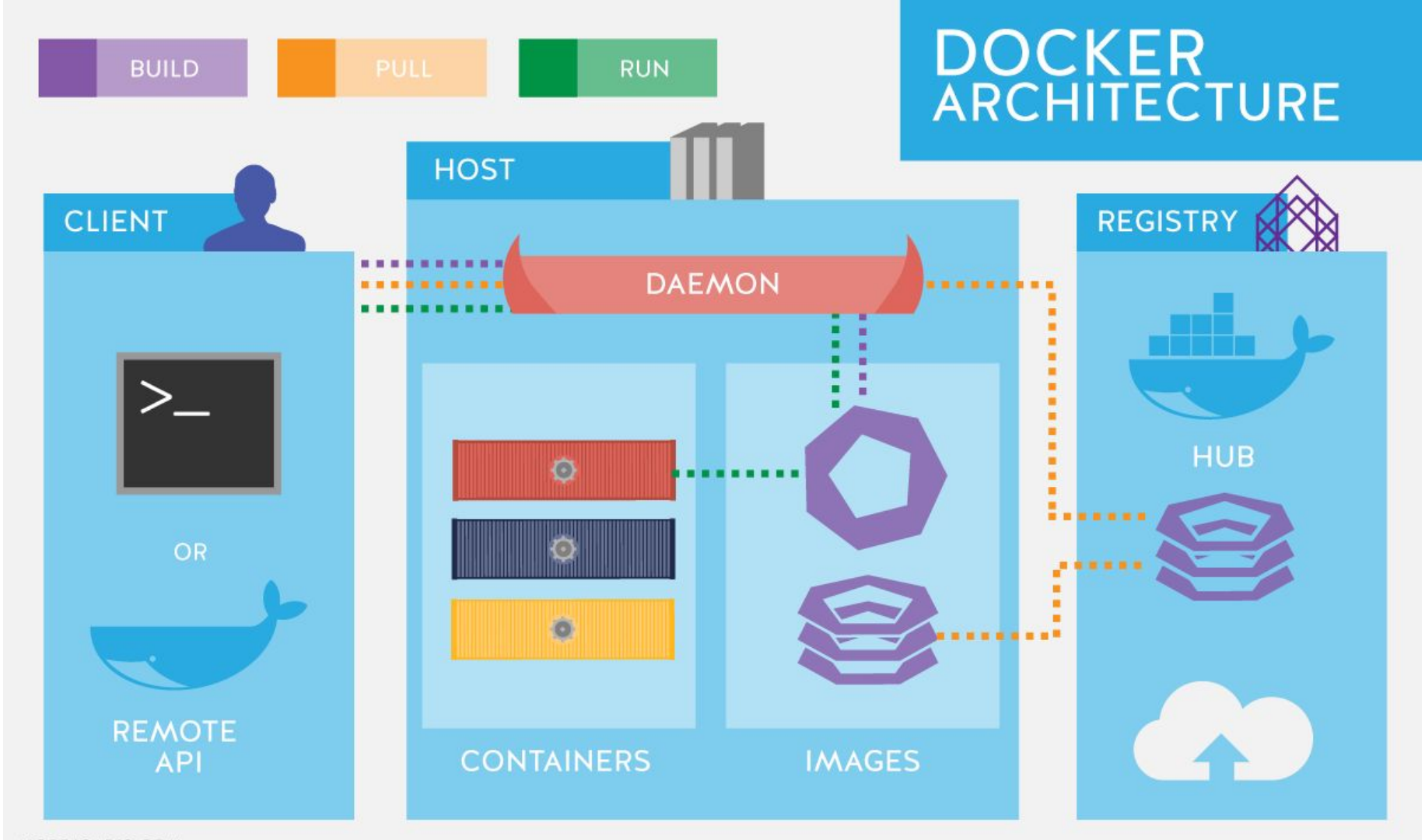
https://www.edureka.co/blog/what-is-docker-container

Docker Components



Docker Engine




DOCKER ARCHITECTURE
BUILD
PULL
RUN
CLIENT
>_
OR
REMOTE API
HOST
DAEMON
CONTAINERS
IMAGES
REGISTRY
HUB

Develop with Docker

Follow the install instructions for your Operating System:

https://docs.docker.com/install/

Add your user to the docker group (make life easier)

Verify version: docker --version

Run hello-world container

```
 $ docker run hello-world
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://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

What’s Dockerfile

A way to build a new Docker image

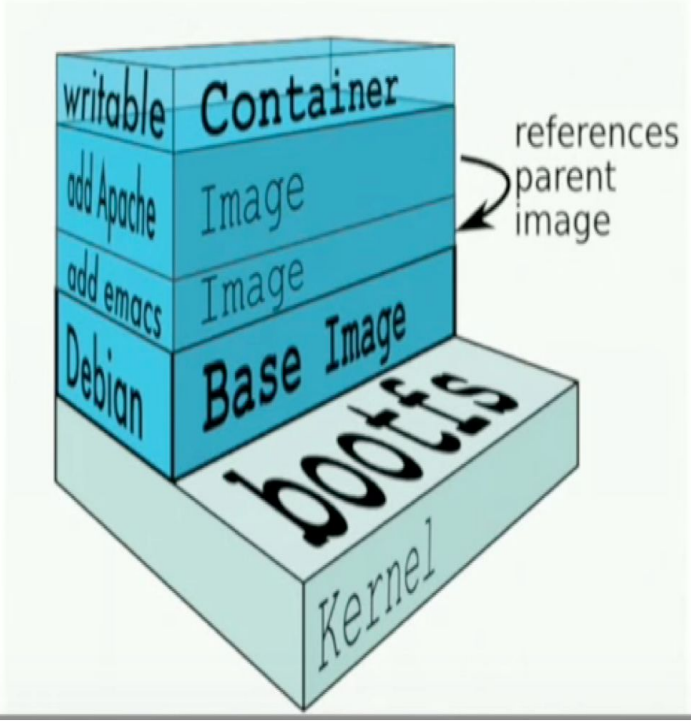
Starting from a base image we define a set of instructions to build a new image.

Each instructions generates a new docker layer

Simplest docker file:

```
FROM alpine
```

Image layers



Images

ID:	ca1f5f48ef43
Parent:	91bac885982d
Name:	my_image:1.0

ID:	91bac885982d
Parent:	3df5aff384fc
Name:	""

ID:	3df5aff384fc
Parent:	a719479f5894
Name:	""

ID:	a719479f5894
Parent:	""
Name:	""

Layers

.../aufs/diff

- ca1f5f48ef43 ...
- 91bac885982d ...
- 3df5aff384fc ...
- a719479f5894 ...

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>


COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>


COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Base image

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>

COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Set working directory inside the container

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>

COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Copy a file from your local machine inside the container

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>

COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Run a command inside the container, for example to install some library

Dockerfile (cont.)

```
FROM openjdk:8

WORKDIR <directory name>


COPY <file_source_from_context> <destination_on_the_container>

RUN <some_command_to_run_to_modify_your_container>

ENTRYPOINT ["<some_command>”]
```

Define which command will be executed when we the container starts

Docker build command

docker build -t <name_of_the-generated_image> <docker_build_context>

```
$ docker build -t uob .
Sending build context to Docker daemon  5.632kB
Step 1/1 : FROM alpine
 ---> 196d12cf6ab1
Successfully built 196d12cf6ab1
Successfully tagged uob:latest
```

Docker build command

docker build -t <name_of_the-generated_image> <docker_build_context>

```
$ docker build -t uob .
Sending build context to Docker daemon  5.632kB
Step 1/1 : FROM alpine
 ---> 196d12cf6ab1
Successfully built 196d12cf6ab1
Successfully tagged uob:latest
```

Docker build command

docker build -t <name_of_the-generated_image> <docker_build_context>

```
$ docker build -t uob .
Sending build context to Docker daemon  5.632kB
Step 1/1 : FROM alpine
 ---> 196d12cf6ab1
Successfully built 196d12cf6ab1
Successfully tagged uob:latest
```

Where is my image?

The images build are kept local, watch out at your disk space!

Get the list of available images:

```
$ docker images

REPOSITORY          TAG         IMAGE ID        CREATED         SIZE
alpine              8           9a4r52cf6ab1    6 weeks ago     14.41MB
uob                 latest      196d12cf6ab1    5 weeks ago     4.41MB
```

Note the image ID is the last ID returned by the build command, is the ID of the last layer

Run my container!

Run a container

```
docker run uob
```

Run a container in detached mode

```
docker run -d uob
```

Run with an allocated pseudo TTY in interactive mode

```
docker run -it uob
```

...for more options: https://docs.docker.com/engine/reference/commandline/run/

LIVE CODING

Dockerize our uob app

```
FROM openjdk:8

WORKDIR /app
RUN apt-get update; apt-get install zip

# cloning the repo
RUN git clone https://github.com/MadalinaPatrichi/uob-cloud-computing.git


# build the app via gradlew
RUN cd /app/uob-cloud-computing/app && ./gradlew build --full-stacktrace


# run the app
ENTRYPOINT ["java", "-jar", "/app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar"]
```

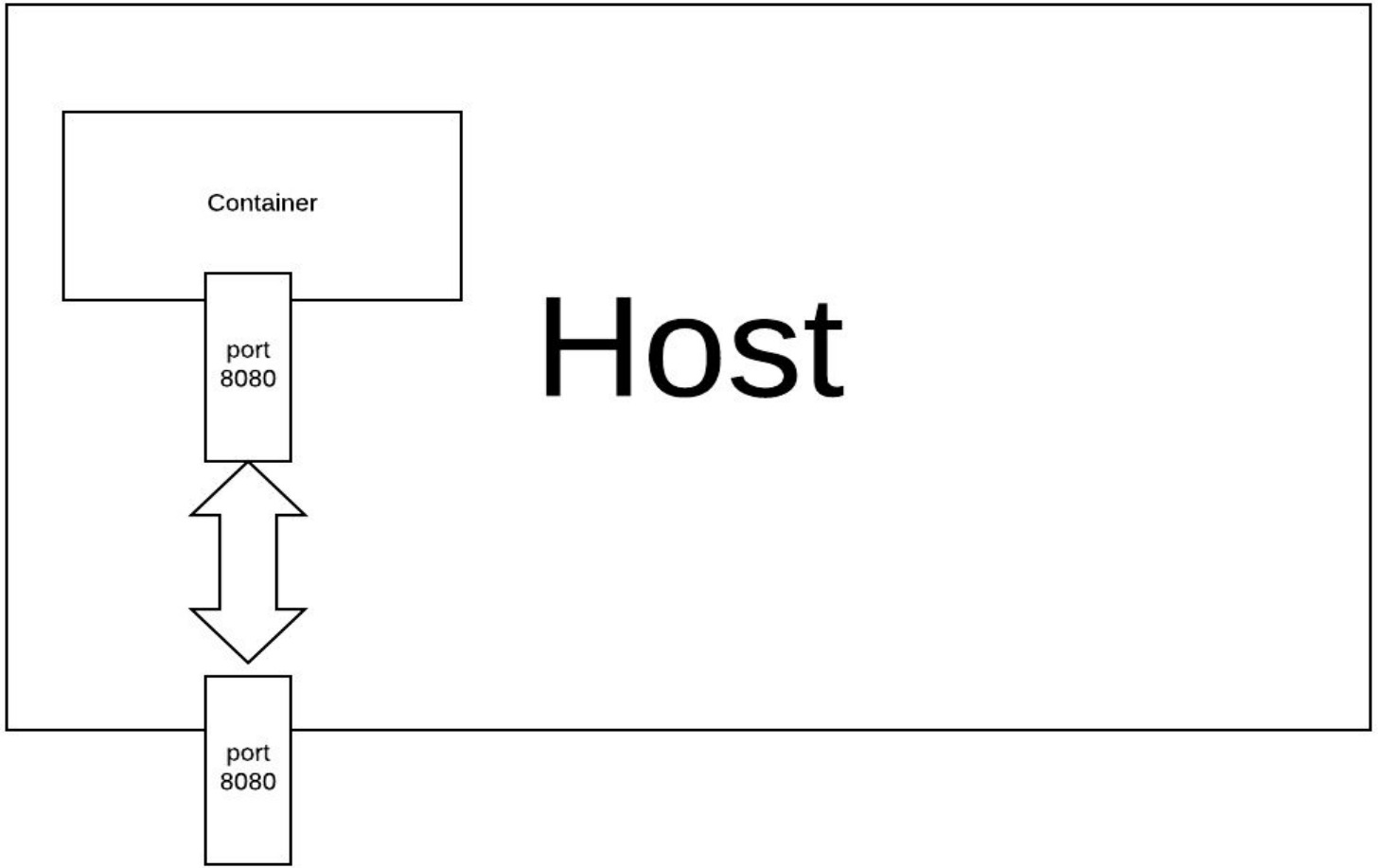
Dockerize our uob app

```
$ docker build -t uob-full .

$ docker  run -it uob-full
```

LIVE CODING

Mapping a port



Export the port and map it locally

$ docker run -it -p8080:8080 uob-full

The local port 8080 is mapped to the container port 8080.

Curl http://localhost:8080/api/todos

Publish image on a docker repository: Oracle registry

1. Tag the image, the tag needs to include the repository URL
2. Docker login <registry url>
3. Docker push <image_name>

MENU

ORACLE Cloud Infrastructure

Search

us-ashburn-1

Networking

DATABASE

Bare Metal, VM, and Exadata

Autonomous Data Warehouse

Autonomous Transaction Processing

SOLUTIONS, PLATFORM AND EDGE

Email Delivery

Edge Services

Developer Services

GOVERNANCE AND ADMINISTRATION

Identity

Security

Governance

Users

Groups

Dynamic Groups

Policies

Compartments

nstances *in* chapter3 *Compartment*

Create Instance

Sort by: Created Date (Desc)

Displaying 1 Instances Page 1

RUNNING

instance-20180914-1447

OCID: ...yhwsmq Show Copy

Shape: VM.Standard2.1

Region: iad

Availability Domain: qAGf:US-ASHBURN-AD-3

Fault Domain: FAULT-DOMAIN-3

Created: Fri, 14 Sep 2018 13:49:21 GMT

Maintenance Reboot: -

Displaying 1 Instances Page 1

MENU ORACLE Cloud Infrastructure Search us-ashburn-1

Create/Reset Password | Unblock | Delete | Apply Tag(s)

ACTIVE

User Information | Tags

OCID: ...lvwcbq Show Copy **Status:** Active

Created: Wed, 12 Sep 2018 20:44:20 GMT

Resources

API Keys (1)

Auth Tokens (1) ← 1

SMTP Credentials (0)

Customer Secret Keys (0)

Groups (1)

Auth Tokens

Displaying 1 Auth Tokens

Generate Token ← 2

AT

OCID: ...a4r3ja Show Copy **Description:** ocir_token **Created:** Fri, 19 Oct 2018 21:04:18 GMT

MENU

ORACLE Cloud Infrastructure

Search

us-ashburn-1

Create/Reset Password | Unblock | Delete | Apply Tag(s)

ACTIVE

Generate Token

help cancel

DESCRIPTION

my-token-description

Generate Token

Resources

- API Keys (1)
- **Auth Tokens (1)**
- SMTP Credentials (0)
- Customer Secret Keys (0)
- Groups (1)

Displaying 1 Auth Tokens

Generate Token

AT

OCID: ...a4r3ja Show Copy **Description:** ocir_token **Created:** Fri, 19 Oct 2018 21:04:18 GMT

Terms of Use and Privacy Cookie Preferences

MENU ORACLE Cloud Infrastructure Search us-ashburn-1

Create/Reset Password | Unblock | Delete | Apply Tag(s)

ACTIVE

Generate Token

help close

GENERATED TOKEN

)W)KdDZg;bNWd+wKtl0<

Copy this token for your records. It will not be shown again.

Copy

Close

Resources

- API Keys (1)
- **Auth Tokens (2)**
- SMTP Credentials (0)
- Customer Secret Keys (0)
- Groups (1)

Displaying 2 Auth Tokens

	OCID	Description	Created
AT	...sovnsq Show Copy	my-token-description	Mon, 22 Oct 2018 14:31:25 GMT
AT	...a4r3ja Show Copy	ocir_token	Fri, 19 Oct 2018 21:04:18 GMT

Push to Oracle registry a.k.a. OCIR

Example:

$ docker tag 196d12cf6ab1 iad.ocir.io/uobtestaccount1/uob-full

$ docker login iad.ocir.io

Username: uobtestaccount1/uob_test_account_1@fastmail.com

Password:

$ docker push iad.ocir.io/uobtestaccount1/uob-full

LIVE CODING

Exec into a container

$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5a4580f0380c	uob-full	"java -jar /app/uob-…"	3 seconds ago	Up 2 seconds	0.0.0.0:8080->8080/tcp	dazzling_chaplygin

$ docker exec -it dazzling_chaplygin /bin/bash

Note you can define a name for your container when you run it with the **--name** option

Exec into a container

```
root@a33aef351b07:/app# ps -aux
USER    PID %CPU %MEM    VSZ   RSS TTY   STAT START   TIME COMMAND
root      1  0.3  5.5  436940 843572 ?     Ssl  11:09   0:32 java -jar /app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar
root    130  0.2  0.0  19952  3628 pts/0  Ss   14:05   0:00 /bin/bash
root    136  0.0  0.0  38388  3160 pts/0  R+   14:05   0:00 ps -aux
```

Docker multi stage

The image build is very big, it is based on the openjdk + we add our layers for configuring out application, the resulting image is 1.34GB

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
uob-full	latest	7e8b2d9d208b	18 hours ago	1.34GB
openjdk	8	81f83aac57d6	6 weeks ago	624MB

Can we do a better JOB?

Docker multi stage

We only need to add in our image the jar file and having a base image with the java virtual machine.

Step 1 Build the jar

Step 2 Put the jar into a slim image with the JVM

Docker multi stage

```
FROM openjdk:8 AS builder

WORKDIR /app

RUN apt-get update; apt-get install zip

# cloning the repo

RUN git clone https://github.com/MadalinaPatrichi/uob-cloud-computing.git

RUN cd /app/uob-cloud-computing/app && ./gradlew build --full-stacktrace

FROM openjdk:alpine

WORKDIR /app

COPY --from=builder /app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar .

ENTRYPOINT ["java", "-jar", "uob-todo-app-0.1.0.jar"]
```

Docker multi stage

```
FROM openjdk:8 AS builder

WORKDIR /app

RUN apt-get update; apt-get install zip

# cloning the repo

RUN git clone https://github.com/MadalinaPatrichi/uob-cloud-computing.git

RUN cd /app/uob-cloud-computing/app && ./gradlew build --full-stacktrace

FROM openjdk:alpine

WORKDIR /app

COPY --from=builder /app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar .

ENTRYPOINT ["java", "-jar", "uob-todo-app-0.1.0.jar"]
```

Docker multi stage

```
FROM openjdk:8 AS builder

WORKDIR /app

RUN apt-get update; apt-get install zip

# cloning the repo

RUN git clone https://github.com/MadalinaPatrichi/uob-cloud-computing.git

RUN cd /app/uob-cloud-computing/app && ./gradlew build --full-stacktrace
```

```
FROM openjdk:alpine

WORKDIR /app

COPY --from=builder /app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar .

ENTRYPOINT ["java", "-jar", "uob-todo-app-0.1.0.jar"]
```

Docker multi stage

```
FROM openjdk:8 AS builder

WORKDIR /app

RUN apt-get update; apt-get install zip

# cloning the repo

RUN git clone https://github.com/MadalinaPatrichi/uob-cloud-computing.git

RUN cd /app/uob-cloud-computing/app && ./gradlew build --full-stacktrace
```

```
FROM openjdk:alpine

WORKDIR /app

COPY --from=builder /app/uob-cloud-computing/app/build/libs/uob-todo-app-0.1.0.jar .

ENTRYPOINT ["java", "-jar", "uob-todo-app-0.1.0.jar"]
```

Check the image size

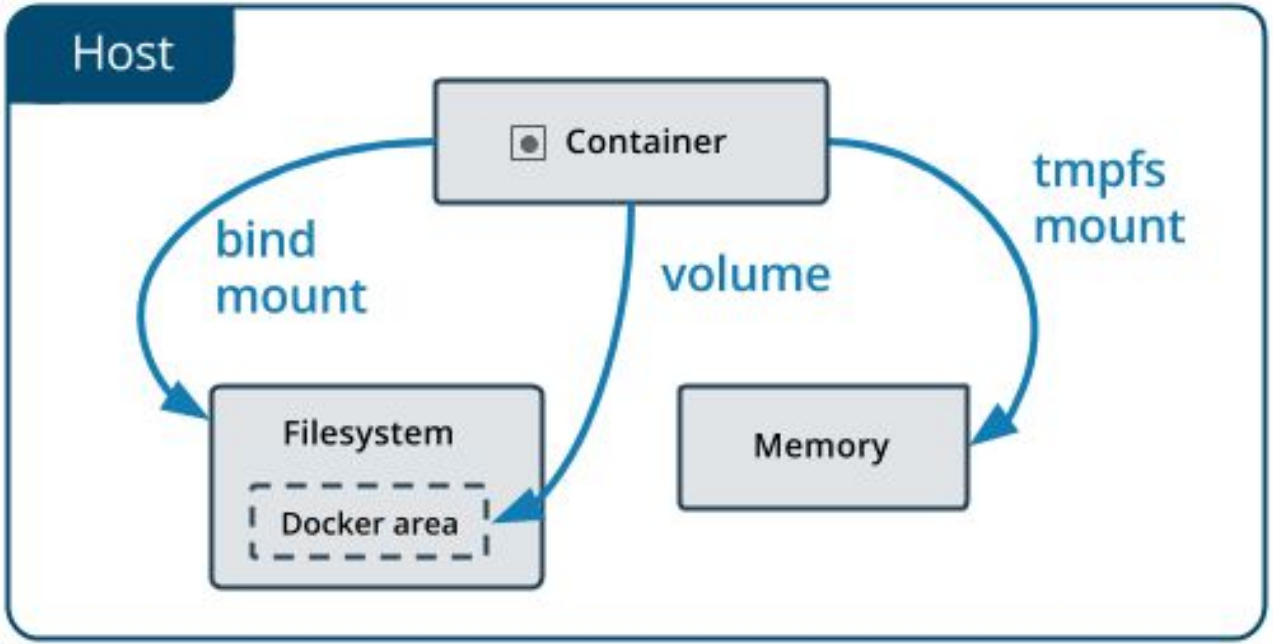
$ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
uob-multi	latest	ad4e6a57f7aa	21 hours ago	**147MB**
uob-cloud	1	7e8b2d9d208b	3 days ago	**1.34GB**

Happy days!

Manage application data

- Volumes
- Bind mounts
- tmpfs mounts



Manage application data - Volumes

- Part of the host filesystem managed by Docker: `/var/lib/docker/volumes/`
- Create a volume:

  $ docker volume create

  C80df714dc7e2c4b19c7a92b45d72c57aa121b90a0fd5d5c6bfaf63054d630e1

- Volumes can be mounted into multiple containers simultaneously

Manage application data - Volumes

- Check the volumes existing created on the container:

  $docker volume list

  DRIVER VOLUME NAME

  local c80df714dc7e2c4b19c7a92b45d72c57aa121b90a0fd5d5c6bfaf63054d630e1

- Remove volumes:

  $docker volume rm c80df714dc7e2c4b19c7a92b45d72c57aa121b90a0fd5d5c6bfaf63054d630e1

Manage application data - Volumes

When to use volumes:

- Use data across multiple containers

- Store data generated by the container on a remote host, rather than locally
  - Benefits: ability to backup data, defer storage responsibilities to the cloud provider, etc.

- Back-up/restore/migrate data from one Docker host to another:
  - Stop containers from using the volume
  - Back-up the volume’s directory

Manage application data - Bind mounts

When to use bind mounts:

- Share configuration files from the host machine to containers
  - Internally used by Docker to provide DNS resolution (mounts /etc/resolv.conf from the host machine into each container)

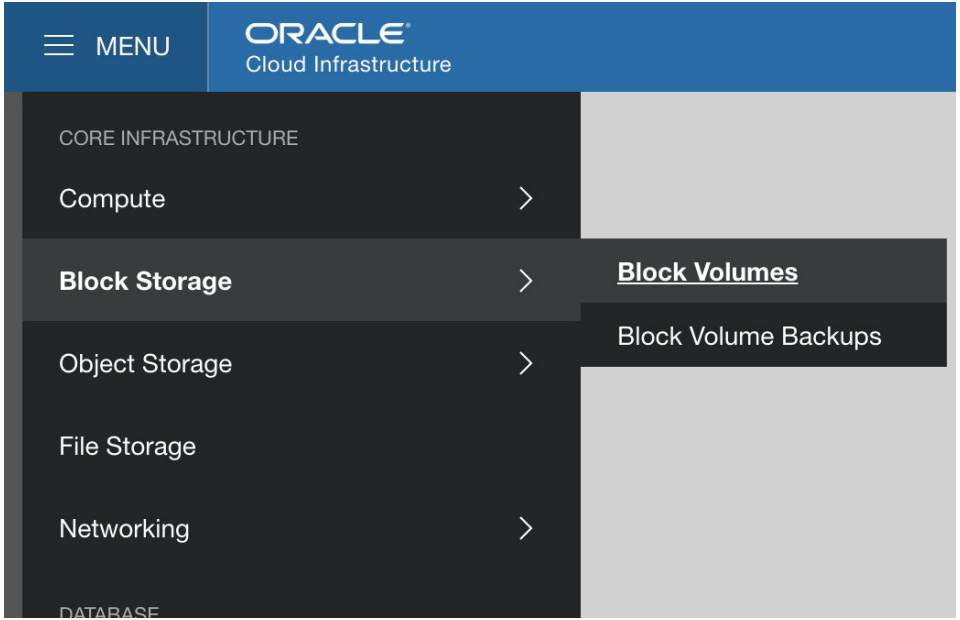
- Share source code or build artifacts between a development environment on the Docker host and a container

Manage application data - tmpfs

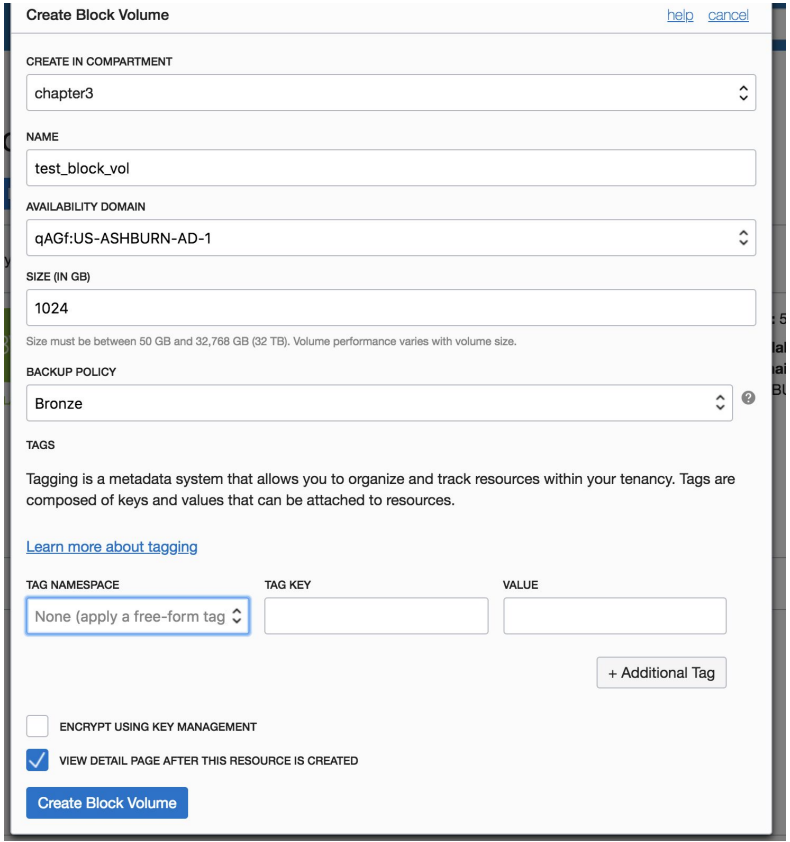
When to use tmpfs:

- Data not wanted to be persisted on the host machine or within the container
- App neds to write a large volume of non-persistent state data
- Enhance the performance of the container
- Security
  - Container created in read-only mode

Adding a Block Volume

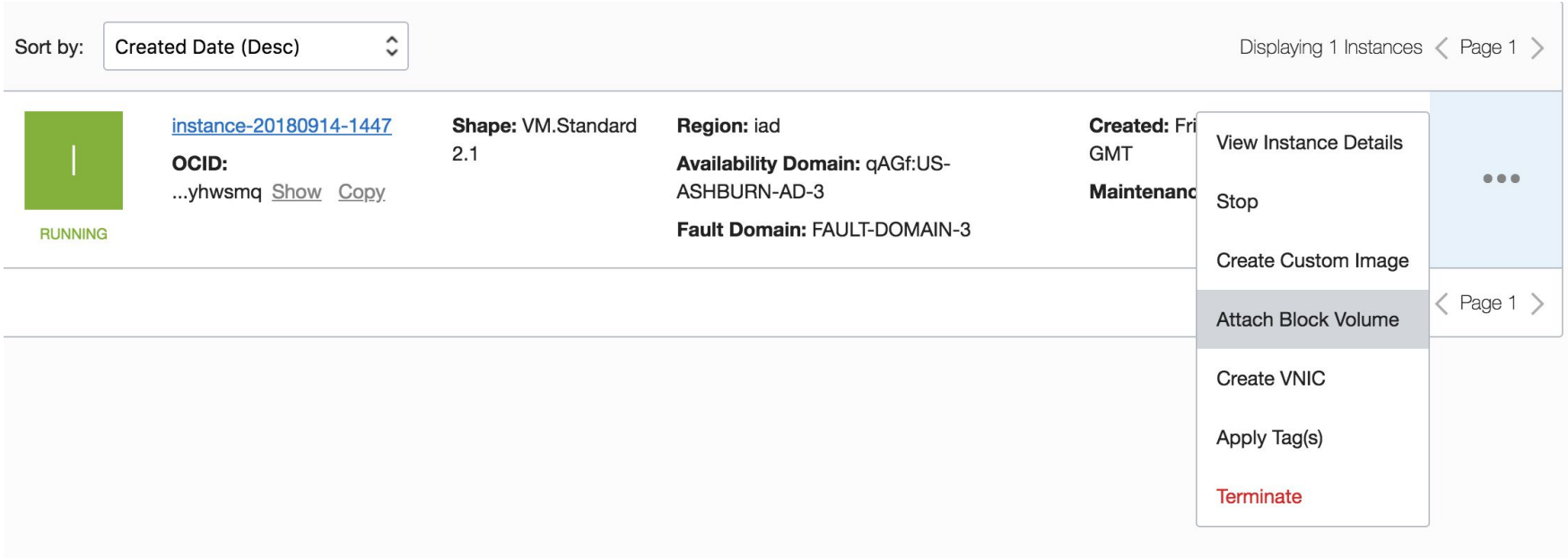


Adding a Block Volume

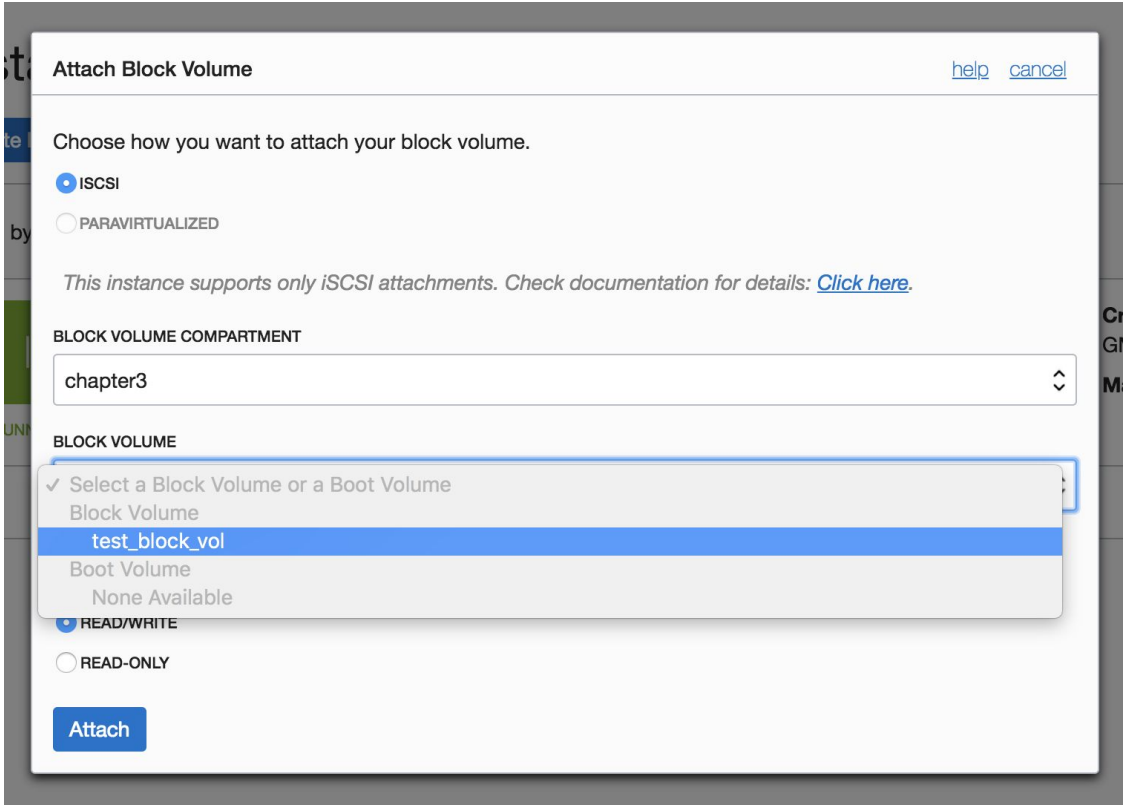


Note: Make sure that your Block Volume is in the same Availability Domain as your target instance

Adding a Block Volume



Adding a Block Volume



Adding a Block Volume

MENU | ORACLE Cloud Infrastructure | Search | us-ashburn-1

Storage » Block Volumes » Block Volume Details

BV

AVAILABLE

test_block_vol

iSCSI Commands & Information | Detach from Instance | Resize | Delete Block Volume | Apply Tag(s)

Block Volume Information | Tags

OCID: ...vyhi6a Show Copy

Size: 1.0 TB

Availability Domain: qAGf:US-ASHBURN-AD-3

Attachment Access: Read/Write

Hydrated: true

Encryption Key: None

Attached Instance: instance-20180914-1447

Date Attached: Tue, 23 Oct 2018 14:21:18 GMT

Protocol: iscsi

Created: Fri, 19 Oct 2018 12:56:27 GMT

Backup Policy: None Assign

Resources

Backups (0)

Clones (0)

Backups

No Backups

Create Backup

There are no Backups for this Block Volume.

Create Backup

Adding a Block Volume

Volume Details

test

iSCSI C

Block V

OCID:

Size:

Availab

Attach

Hydrat

Encryp

Bac

Creat

iSCSI Commands & Information

help close

Use OS tools to edit your /etc/fstab volume to have the _netdev and nofail options from the OS. Failure to run commands will cause instance boot failure.

ATTACH COMMANDS

```
sudo iscsiadm -m node -o new -T iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-819
sudo iscsiadm -m node -o update -T iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-
sudo iscsiadm -m node -T iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-8194-27e13
```

Copy

DETACH COMMANDS

```
sudo iscsiadm -m node -T iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-8194-27e13
sudo iscsiadm -m node -o delete -T iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-
```

Copy

IP ADDRESS AND PORT

169.254.2.3:3260

Copy

VOLUME IQN

iqn.2015-12.com.oracleiaas:3b568db7-89c5-46d4-8194-27e13582d7e3

Copy

30914-1447

14:21:18 GM

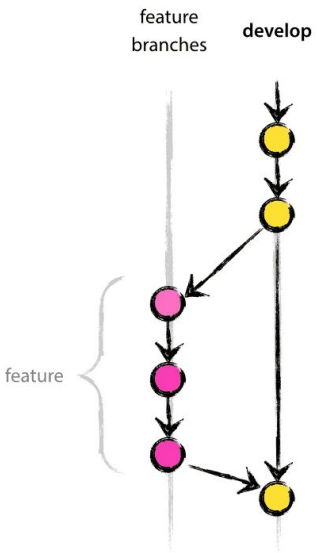
7 GMT

Mounting a Block Volume and attach to container

LIVE CODE

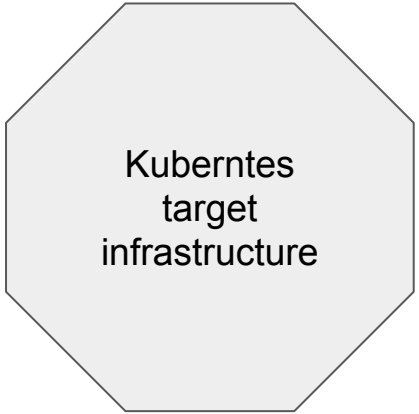
Practical use case for using Docker:

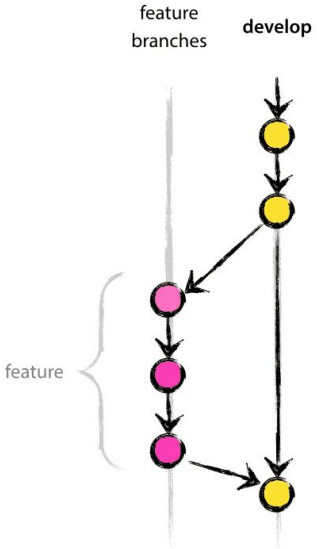
Testing Oracle Cloud Controller Manager (CCM)



Practical use case for using Docker:

Testing Oracle Cloud Controller Manager (CCM)

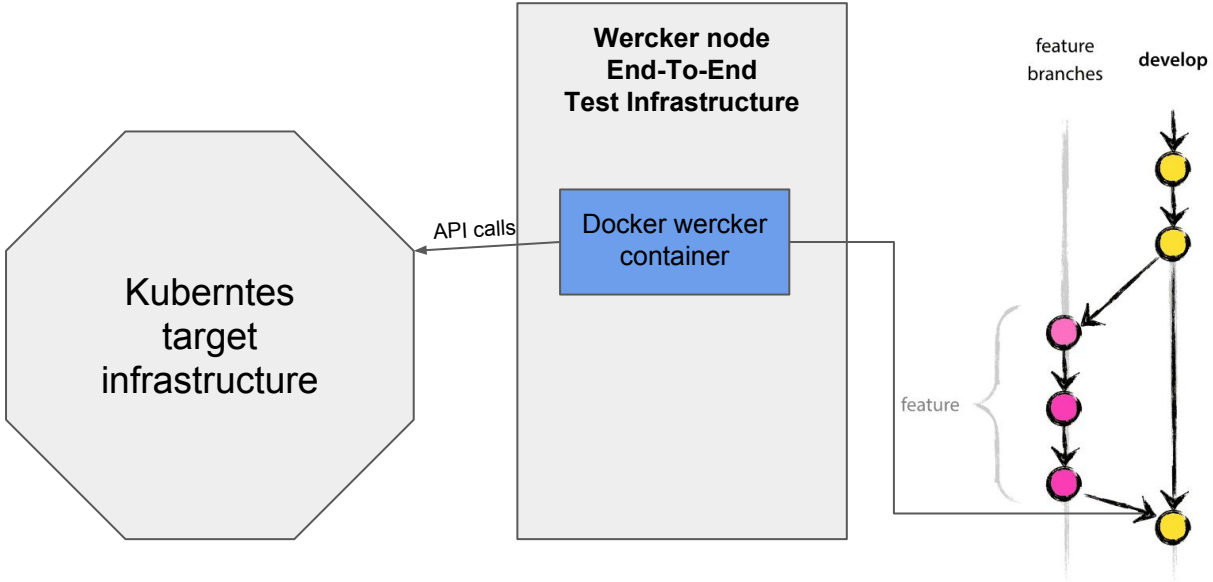




https://github.com/oracle/oci-cloud-controller-manager/blob/master/ci-docker-images/Dockerfile

Practical use case for using Docker:

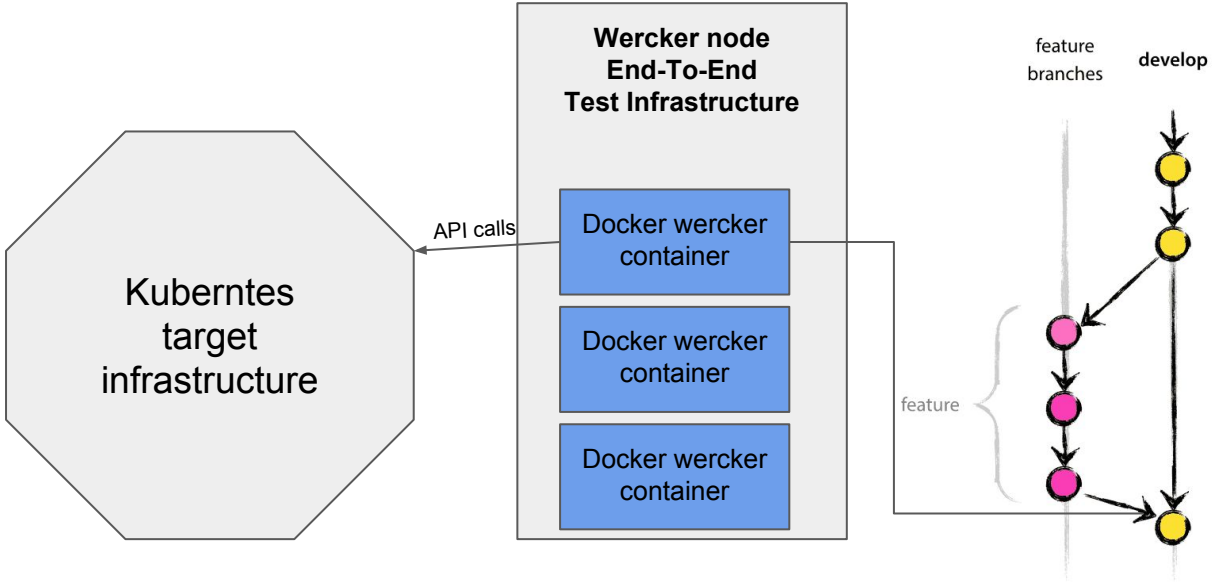
Testing Oracle Cloud Controller Manager (CCM)



https://github.com/oracle/oci-cloud-controller-manager/blob/master/ci-docker-images/Dockerfile

Practical use case for using Docker:

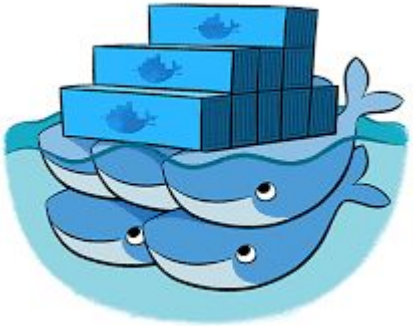
Testing Oracle Cloud Controller Manager (CCM)

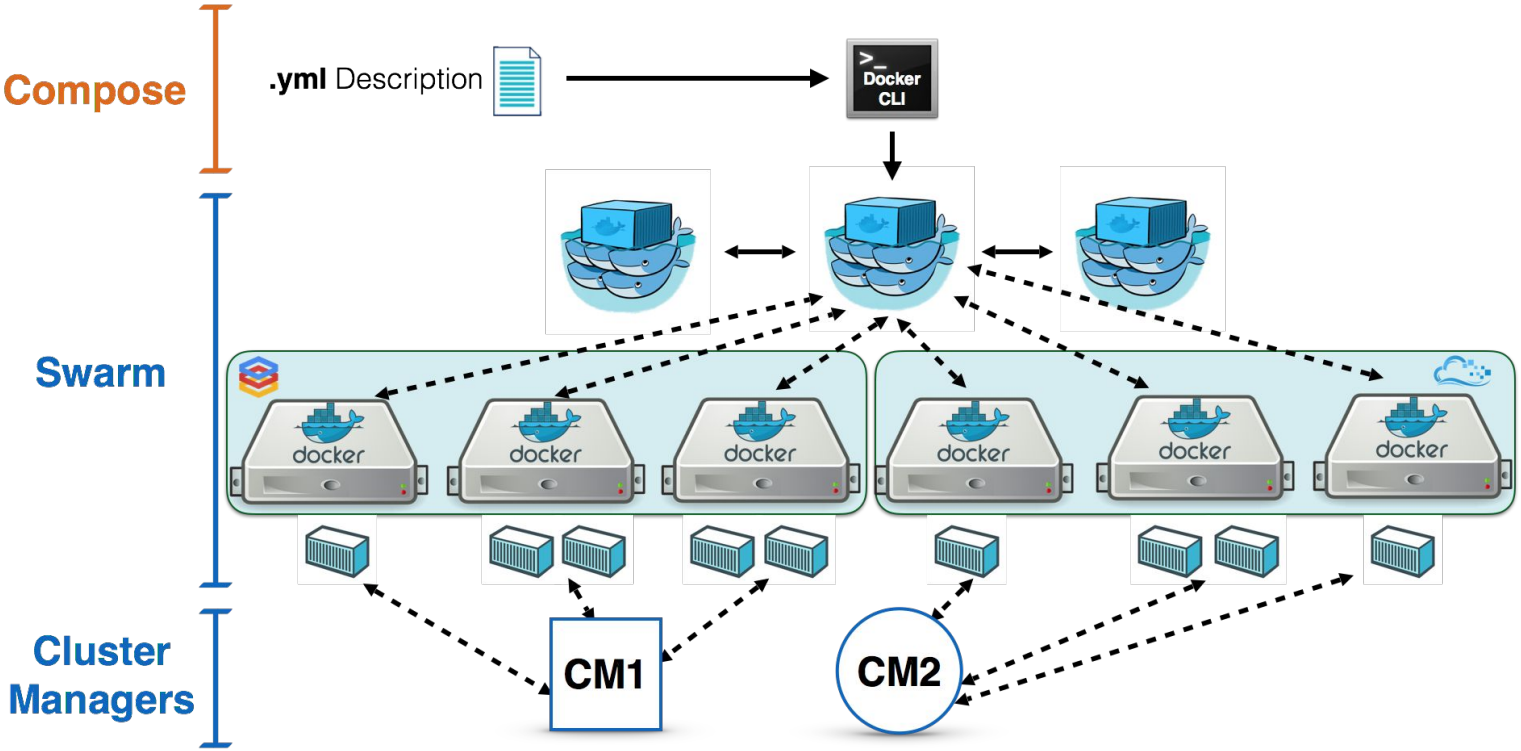


https://github.com/oracle/oci-cloud-controller-manager/blob/master/ci-docker-images/Dockerfile

But wait…..there’s more

Container orchestration with Docker Swarm




Compose
.yml Description
Docker CLI
Swarm
docker
docker
docker
docker
docker
docker
Cluster Managers
CM1
CM2

https://tinyurl.com/uob-cloud-unit-slack