

Docker Containers

CMIS 545 - Cloud Computing Architecture

Rafael Marino, Eduardo Cassinelli

McGill University

November, 2020

- 1 Just Enough Microservices
- 2 Virtual Machines vs Containers
- 3 Docker
- 4 Docker Primitives
- 5 Docker Demo

Section 1

Just Enough Microservices

What are Microservices?

- Microservices are a good starting point for containers
- Microservices are self-contained, independent, autonomous, loosely coupled services that work together
- No standard definition, they are usually built around a business domain
- Designed specifically to allow independent deployability of each service

How do Microservices relate to Containers?

- Containers facilitate the modularized development and deployment of microservices
- For this level of independence and flexibility to exist, a clear definition between service boundaries must also exist
- Having services tied to specific machines would be inefficient and risky
- Using one container per service guarantees independence

Section 2

Virtual Machines vs Containers

Virtual Machines

- Virtualization allows the creation of an abstraction layer (Hypervisor) on top of the host OS to divide and virtualize physical resources
- Resources from the host machine are carved out into multiple Virtual Machines
- Each VM runs an independent, guest OS

Containers

- Containerization is based on the creation of isolated partitions or zones directly from within the host OS, no virtualization software required
- Partitions only have access to their own content, and to specific shared OS Kernel components
- A container is a standard unit of software that can be deployed on any other machine, virtual or otherwise, that runs a compatible OS; independently of any other containers

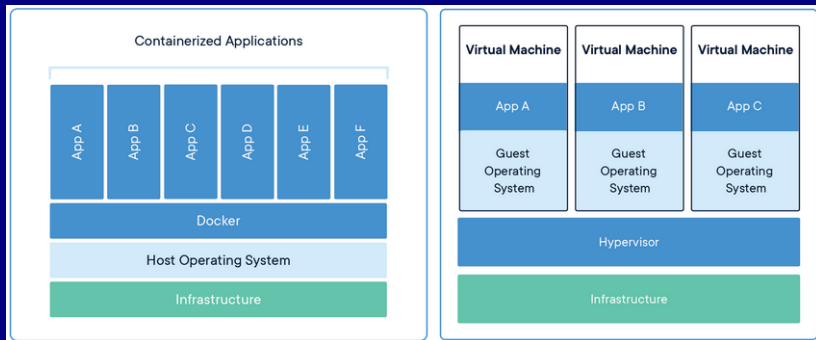


Figure 1: Container vs VM Architecture

Section 3

Docker

Definition

Docker is a Platform xxx

- Docker can package code from an application, including all its dependencies, into a container.

Section 4

Docker Primitives

Just Enough Microservices

○○○

Virtual Machines vs Containers

○○○○

Docker

○○

Docker Primitives

○●○○○

Docker Demo

○○○○○○○○

Docker Engine

Docker Image

- We can think of a Docker Image as a stopped Docker Container
- Each element within an image represents an image layer. Layers are then stacked on top of each other and ready to run.
-

Docker Container

- Containers are the central unit on top of which all Docker is built, and they are better examined practically.

Docker Compose

Section 5

Docker Demo

Graylog App

- Graylog is an open source log management solution for capturing, storing, and analyzing machine data
- It needs two dependencies:
 - MongoDB: An open-source, “general purpose, document-based, distributed database”
 - Elasticsearch: An open-source, “powerful analytics engine to explore data easily”

Docker Run

~\$ docker container run <options> <image>:<tag> <app>

- <options>: refers to the container command flags
- <image>: refers to the original name or id of the image
- <tag>: refers to the specific version of that image

~\$ docker container run -d --name ubuntu ubuntu:latest

Graylog Setup

```
eduardo@eduardo-L380:~$ docker container run --name mongo -d mongo:3
```

```
eduardo@eduardo-L380:~$ docker run --name elasticsearch \  
-e "http.host=0.0.0.0" \  
-e "ES_JAVA_OPTS=-Xms512m -Xmx512m" \  
-d docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.10
```

```
eduardo@eduardo-L380:~$ docker run --name graylog --link mongo --link  
elasticsearch \  
-p 9000:9000 -p 12201:12201 -p 1514:1514 -p 5555:5555\  
-e GRAYLOG_HTTP_EXTERNAL_URI="http://127.0.0.1:9000/" \  
-d graylog/graylog:3.3
```

Figure 2: Graylog Setup Commands

Note

In Ubuntu 20.04 LTS stock, installing graylog requires adjusting default virtual memory settings using: *sudo sysctl -w vm.max_map_count=262144*

Running Containers

```
eduardo@eduardo-L380:~$ docker container ls
```

CONTAINER ID	IMAGE	CREATED	STATUS	NAMES
1de8801d699f	graylog/graylog:3.3	4 seconds ago	Up 3 seconds (health: starting)	graylog
76b0e75fb7be	./elasticsearch/	22 seconds ago	Up 21 seconds	elasticsearch
229f3aebfe56	mongo:3	27 seconds ago	Up 26 seconds	mongo

Figure 3: Running Containers' List

Testing Graylog

```
eduardo@eduardo-L380: $ echo 'Testing log message for CMIS545 Cloud  
Computing Architecture' | nc localhost 5555
```

Figure 4: Echo Command

The screenshot displays the Graylog web interface. At the top, there's a header 'All Messages' with a hamburger menu icon on the left and a search icon on the right. Below the header, a table lists messages. The first message is selected, showing its details in a sidebar on the right. The message content is 'Testing log message for CMIS545 Cloud Computing Architecture'. The source is '172.17.0.1'. The timestamp is '2020-11-16 16:39:19 +00:00'. The message ID is '46318500-282a-11eb-a630-0242ac11000'. The sidebar also shows the message was received by 'd49f27d0 / 8759155980c7', stored in index 'graylog_0', and routed to the 'All messages' stream. The message details section shows the message content, source IP, and timestamp.

timestamp	source
2020-11-16 16:39:19 +00:00	172.17.0.1

Testing log message for CMIS545 Cloud Computing Architecture

✉ 46318500-282a-11eb-a630-0242ac11000 [Permalink](#) [Copy ID](#) [Show surrounding messages](#) [Test against stream](#)

Timestamp
2020-11-16 16:39:19.560

Received by
Text Input on [d49f27d0 / 8759155980c7](#)

Stored in index
[graylog_0](#)

Routed into streams

- [All messages](#)

message
Testing log message for CMIS545 Cloud Computing Architecture

source
172.17.0.1

timestamp
2020-11-16 16:39:19 +00:00

Figure 5: Graylog Dashboard

Stopping and Removing Containers

```
eduardo@eduardo-L380:~$ docker container stop mongo
```

```
eduardo@eduardo-L380:~$ docker container ls -a
```

CONTAINER ID	IMAGE	CREATED	STATUS	NAMES
1de8801d699f	graylog/graylog:3.3	2 minutes ago	Up 2 minutes (healthy)	graylog
76b0e75fb7be	./elasticsearch/	53 seconds ago	Up 52 seconds	elasticsearch
229f3aebfe56	mongo:3	2 minutes ago	Exited(0) 5 seconds ago	mongo

Figure 6: Stopping Single Container

```
eduardo@eduardo-L380:~$ docker container stop mongo elasticsearch graylog
```

```
eduardo@eduardo-L380:~$ docker container rm mongo elasticsearch graylog
```

Figure 7: Stopping and Removing all Containers

Further Container Commands[1]

Command	Description
<i>docker container prune</i>	Remove all stopped containers
<i>docker container start</i>	Start one or more stopped containers
<i>docker container diff</i>	Inspect file or directory changes
<i>docker container exec</i>	Run a command in a running container
<i>docker container export</i>	Export a container's filesystem as a tar
<i>docker container inspect</i>	Display detailed information
<i>docker container kill</i>	Kill one or more running containers
<i>docker container logs</i>	Fetch the logs of a container

[1]

Documentation: https://docs.docker.com/engine/reference/commandline/container_run/