

WHAT IS A CONTAINER?

Antonis Kalipetis
@akalipetis

ABOUT ANTONIS

Docker Captain and Docker Certified Associate

Python lover and developer

Senior Software Engineer, e-food.gr

ABOUT 2HOG

We teach the lessons we have learnt the hard way in production.

Consulting, training and contracting services on containers, APIs and infrastructure

WHY SHOULD I CARE ABOUT
CONTAINERS?

WHY SHOULD I CARE ABOUT CONTAINERS?

A way to package and distribute applications

A way to manage compute resources

A way to ship software

CONTAINERS VS VMS

VIRTUAL MACHINES VS. CONTAINERS?

They should co-exist. We should run N Containers in M Virtual Machines ($N > M$).

Imagine a Virtual Machine as a multi-floor building and a Container as a rented flat.

Virtual Machines provide deep isolation, so they are heavy and not versatile

Containers are fast and lightweight

CONTAINERS ARE A SET OF
KERNEL TOOLS AND
FEATURES THAT JAIL AND
LIMIT A PROCESS BASED ON
OUR NEEDS.

WHAT IS A CONTAINER? (IN A BIT MORE DETAILS)

It's a **process**

Isolated in it's own world, using **namespaces**

With limited resources, using **cgroups**

NAMESPACES

A namespace wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the namespace, but are invisible to other processes. One use of namespaces is to implement containers.

The Linux man-pages project:

<http://man7.org/linux/man-pages/man7/namespaces.7.html>

CGROUPS

cgroups (abbreviated from control groups) is a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes.

Wikipedia:

<https://en.wikipedia.org/wiki/Cgroups>

LET'S PLAY A BIT WITH CONTAINERS

```
docker run -it alpine sh
```

```
whoami
```

```
uname -a
```

```
top
```

```
cat /etc/os-release
```

```
exit
```

WHAT DID JUST HAPPEN?

The Alpine image was pulled

A new container (aka a process) was started using that image

The process was isolated in its own namespace

A TTY was opened for us, so we could run commands

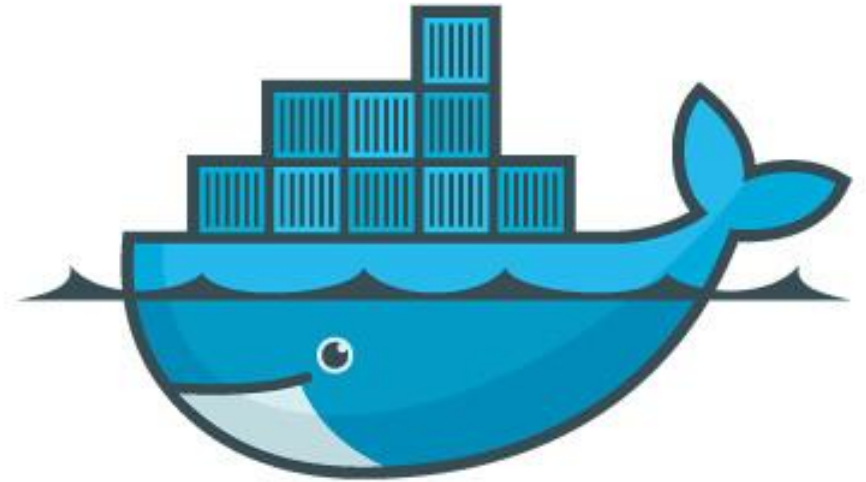
LET'S SEE THE DIFFERENCES

A container can have a different operating system than the host machine

A container cannot have a different kernel than the host machine – they all share the same kernel after all!

DOCKER ON TOP OF CONTAINERS

Copy on Write file system
Software Defined Networking
Storage management
Built-in Orchestration



docker

DOCKER MADE IT EASY FOR
YOU AND ME TO USE
CONTAINERS

THE MAGIC OF COW FILE SYSTEMS

Image layers can be reused, reducing disk space and download time

Every Dockerfile command creates a new layer

Layers can be cached, reducing build times if the files have not changed

Containers can start blazing fast, because they just create a writable layer and don't need to copy files

THE CONTAINER BEST-PRACTICE LIST

Containers should be considered ephemeral

The container should be single-purposed program (e.g. avoid using supervisord)

The image should be lightweight and slim

Configuration should be made by the environment, with sane defaults

Orchestration should be carried out by an external tool

QUESTIONS?

THANKS!

Antonis Kalipetis

@akalipetis