Docker introduction

January 2019

Índice

an NTT DATA Company

Introduction to containers

ALTEMISTA

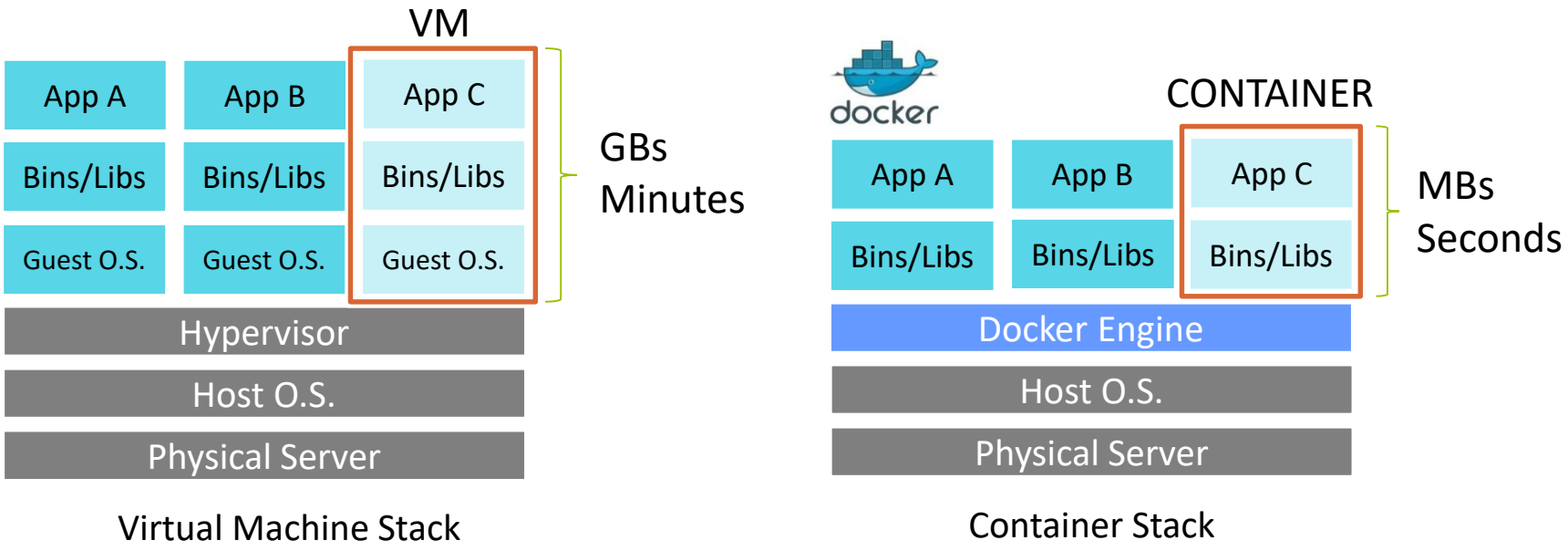
everis

an NTT DATA Company

¿What are containers?

- Self-contained execution environments with their own CPU, memory, I/O, FileSystem and network interfaces that result in isolated processes independent of the host operating system.
- Share the machine's OS system kernel .
- Isolate software so they always run in the same environment. There are no longer differences between environments.
- Similar to virtual machines but without the overhead of the guest operating system.
- They are lighter and start faster. Multiple containers per machine.
- **They are portable.**

Virtual Machines vs Containers



an NTT DATA Company

Containers and Virtual Machines

The containers are NOT virtual machines but…

- **Both** technologies **share characteristics**
- **Both** are designed to **provide an environment** in which to run an application
- **Both** can **move between hosts**

Houses	Virtual Machines	It has its own infrastructure - plumbing, heating, electricity, etc.-
Flats	Docker Containers	They are built around a shared infrastructure.
Buildings	Server running Docker daemon or also called Docker host	Shared plumbing, heating, electricity, etc to each particular floor. The floors are offered in several different sizes.

Introduction to containers

an NTT DATA Company

Containers and Virtual Machines

- **VM stores application code and data with states.** Everything that is usually saved on a physical server is saved.
- **Microservice architecture :** Many small services (each a Docker container) comprise an application.
- **Applications** in containers can be **deconstructed** into much **smaller components**..

Frequently asked questions in Sysadmins

- **How can I back up a container?**
- **What is my patch management strategy for running containers?**
- **Where is the application server running?**

an NTT DATA Company

Why containers?

- The software industry has changed
- Before:
    - Monolithic applications
    - Long development cycles
    - One or two environments as much
    - Vertical scaling every X time
- Now:
    - Uncoupled services
    - Iterative improvements and giving value as soon as possible
    - Multiple environments
    - Horizontal scaling on demand

docker

ALTEMISTA

everis

an NTT DATA Company

The deployment of applications has become very complex

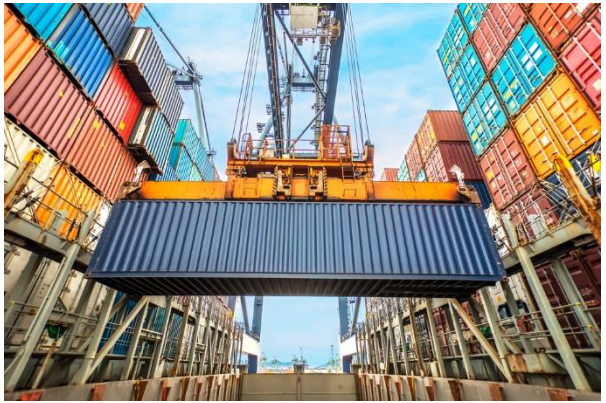
- Different stacks at the same time :
  - Languajes
  - Frameworks
  - Databases
- Diferent objectives :
  - Local environments for development
  - Testing, QA, Staging ...
  - Production: on premise, cloud, hybrid

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

Introduction to containers

an NTT DATA Company

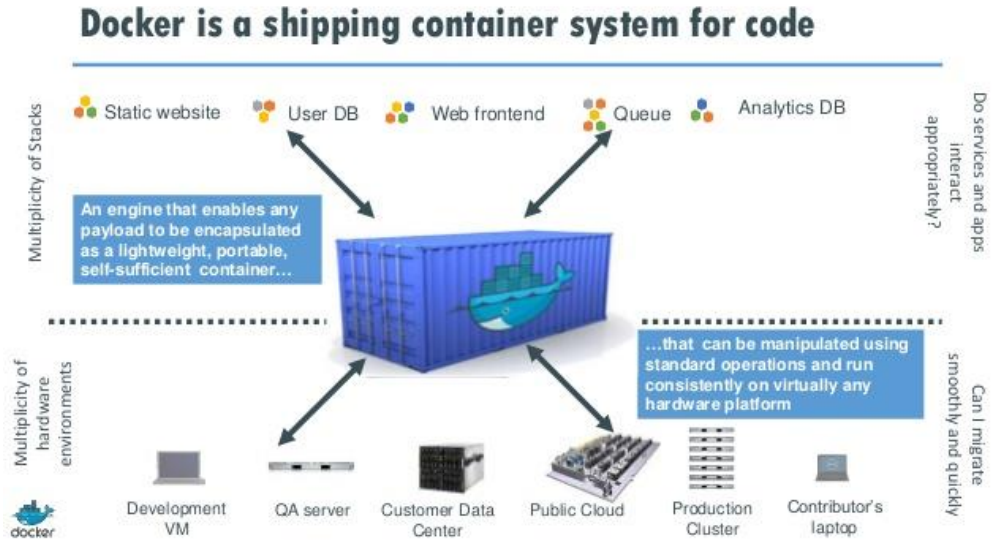
The containers in real life. The inspiration.

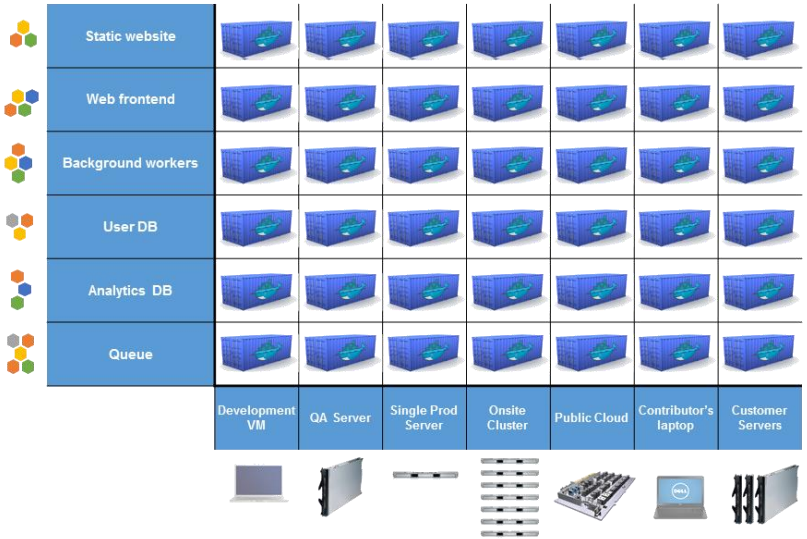


Introduction to containers

an NTT DATA Company

Removing the infernal matrix





"Dev-to-prod reduced from 9 months to 15 minutes (ING)"

an NTT DATA Company

Consecuences

- Same environment for applications
    - We escape from the hell of dependencies
    - We will never say again "In my machine it works“
    - Dev and Ops happy ☺
- Development environments, integration, QA, etc in minutes!
- CI / CD reliable in a simple way.
- Containers as construction devices.
- Decouple the system from the logic of the application.

an NTT DATA Company

What is Docker?

Docker is an open platform for developing, shipping, and running applications using container virtualization technologies.

Docker provides tooling and a platform to manage the lifecycle of your containers :

- A very rich CLI that simplifies the use of containers
- Rest API
- Separation of clear entities: images, containers, networks, volumes, etc.
- Optimizations under the hood
- Philosophy "Infrastructure as code"
- It facilitates a large community

ALTEMISTA

everis

an NTT DATA Company

Versions

Capacities	Community Edition	Enterprise Edition Basic	Enterprise Edition Standard	Enterprise Edition Advanced
Container engine and built-in orchestration, networks, security ...	✓	✓	✓	✓
Infrastructure, plugins and containers certified by Docker		✓	✓	✓
Image management			✓	✓
Container management application			✓	✓
Image security scanner				✓

More information about the different versions : https://www.docker.com/get-docker

an NTT DATA Company

Update wheels

Stable channel	Edge Channel
• Channel to have a platform of confidence. • Stable versions • Allows you to select to send statistics and other data to Docker. • Version once per quarter.	• Channel to get functionalities faster. • It can be unstable and with bugs. • Published once a month.

ALTEMISTA

everis

an NTT DATA Company

Compatibility with containers

Platforms	Linux	MacOS **	Windows (Technology Hyper-V)			
			< Windows 10	Windows 10	Windows Server (EE)	
					WS 2016	> WS 2016
Linux	✓	✓	✓	✓	✗	✓
Windows	✗	✗	✓ **	✓	✓	✓

** Docker ToolBox

an NTT DATA Company

Compatibility with containers

- **Native Docker on Linux** :
  - Ubuntu, CentOS, Debian, Fedora, Oracle Linux, RHEL, SLES…
- **Native Docker on Windows**:
  - Desde Windows Server 2016
- **Docker For Windows**
  - From Windows 10 Pro or Enterprise 64-bit onwards
  - Use virtualization Hyper-V
  - Allows building applications in Linux and Windows Server environments
- **Docker For Mac**
  - From Mac OSX Yosemite 10.10.3 onwards
  - Uses MacOS Hypervisor framework
- **Docker Toolbox**
  - For earlier versions of Windows and Mac OSX
  - VirtualBox + Docker CLI + Machine + Compose
  - Boot2Docker VM (~30MB)

Success Center

- **Official documentation** https://docs.docker.com
- **Certifications** https://success.docker.com/certification
- **Training** https://training.docker.com/instructor-led-training
- **Support** https://www.docker.com/docker-community

an NTT DATA Company

Docker platform

Docker as a platform consists of multiple tools :

- **Docker Engine:** They are the devil of docker and client in charge of executing the containers.
- **Docker Machine:** It allows to install and manage Docker Engine in different types of host: VM, Clouds, etc.
- **Docker Compose:** Allows you to define and execute multi-container Docker applications.
- **Docker Swarm:** The Docker native container orchestrator. Allows managing a cluster of Docker Engines multihost.
- **Docker Hub:** Public record of Docker image repositories.
- **Docker Registry:** Docker private image registration.

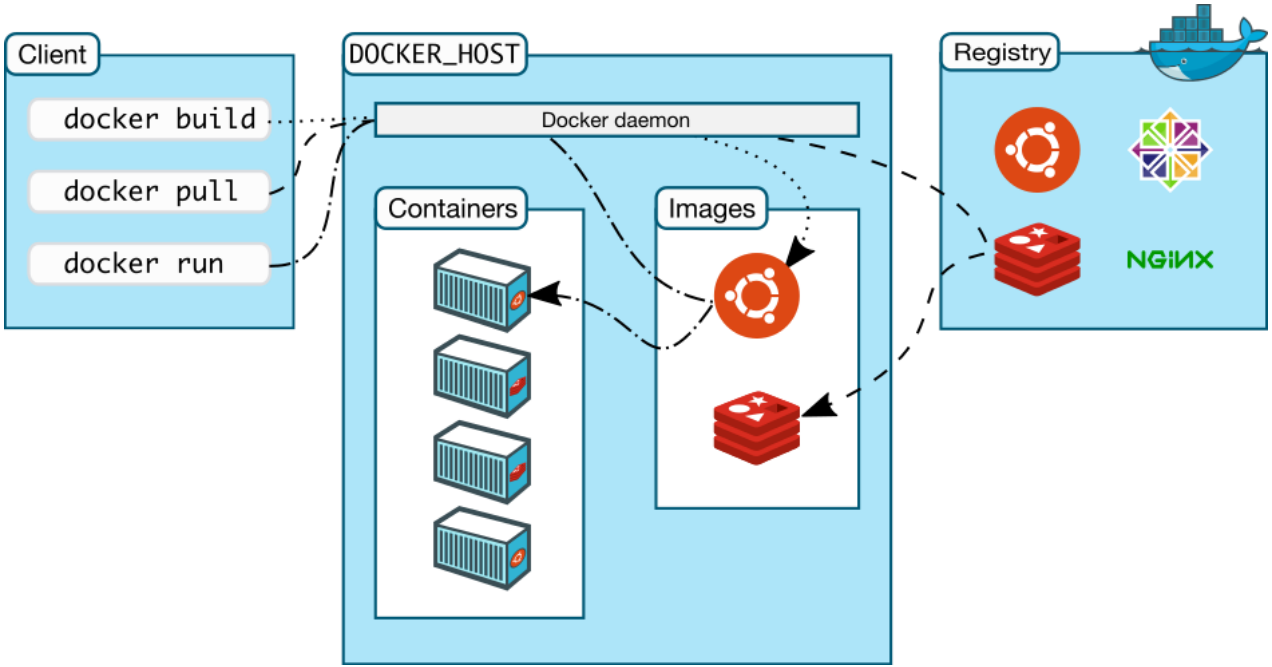
an NTT DATA Company

Docker Engine

It is a client-server application that contains these components :

- **A daemon process (dockerd)** that acts as a server on the host. It is responsible for creating and managing Docker objects such as images, containers, networks, volumes, etc. Can also communicate with other daemons to administer services (Swarm).
- **A REST API** that specifies the interfaces that programs can use to talk to the daemon and tell it what to do.
- **A command interface (CLI) that acts as a client (docker).** Use the REST API to communicate with the daemon. The client can connect to the daemon on the same host and with daemons on remote hosts through UNIX sockets or network interfaces.

Docker Engine



an NTT DATA Company

Docker Internals

- It is written in **Go** and takes advantage of important features of the Linux kernel (also of Windows) to provide its functionality:
- **Namespaces:** Provide an isolation layer for each container creating a set of namespaces that limit access to the outside :
    - **Pid:** Isolation process
    - **Net:** Isolation in the network interfaces
    - **Ipc:** Isolation in the communication between processes
    - **Mnt:** Isolation in the filesystem and mounting points
    - **Uts:** Kernel isolation and version identifiers**.**
- **Cgroups:** Limit access to hardware resources to a container. Ex: available memory
- **UnionFS:** File system that operates with layers, making it very light and fast.
- **Container format :** Combine all this in the libcontainer wrapper.

an NTT DATA Company

Docker objects

When you use Docker, you create and use images, containers, networks, volumes, plugins and other objects :

- **Images**
  - Read-only template with instructions for creating a Docker container.
  - Usually use another image base, adding additional customization.
  - You can create your own images or use created by others from a registry.
  - Each image has one or more read-only layers with the steps that have been taken.
  - Thanks to this the images are light (small and fast) compared with other virtualization systems.
- **Containers**
  - It is an executable instance of an image.
  - You can connect it to one or more networks, associate a storage or create a new image from the current state of the container.
  - It is isolated from other containers. The processes do not have visibility outside.
  - Its state is volatile and disappears when its execution ends.

Docker: Containers

ALTEMISTA

everis

an NTT DATA Company

Hello Container

```
$ docker-machine create -d virtualbox \
--engine-env HTTP_PROXY={{proxy-url} \
--engine-env HTTPS_PROXY={{proxy-url}} \
docker-training
```

```
$ eval $(docker-machine env docker-training --no-proxy)
```

```
$ docker run hello-world
```

```
$ docker run –it ubuntu bash
root@4a02a8f358f1:/# apt-get update && apt-get install -y
figlet lolcat fortune cowsay
root@4a02a8f358f1:/# export PATH=$PATH:/usr/games
root@4a02a8f358f1:/# figlet hola contenedor | lolcat
root@4a02a8f358f1:/# fortune | cowsay | lolcat
```

```
$ docker versión
Client:
 Version:      17.03.1-ce
 API version:  1.27
 Go version:   go1.7.5
 Git commit:   c6d412e
 Built:        Tue Mar 28 00:40:02 2017
 OS/Arch:      windows/amd64

Server:
 Version:      17.04.0-ce
 API version:  1.28 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   4845c56
 Built:        Wed Apr  5 18:45:47 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

Docker: Containers

ALTEMISTA

everis

an NTT DATA Company

Managing containers

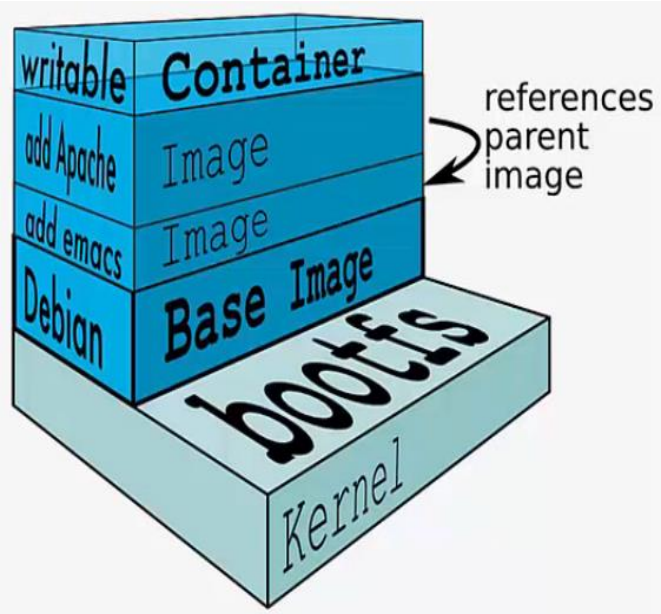
```
$ docker run -d jdeiviz/clock
076303d677b6996073034a56d944cc52b575ab6801e8fb30144cdbd86ccc0711
```

```
$ docker ps
$ docker ps -lq
$ docker logs 076
$ docker logs --tail 3 076
$ docker logs --tail 1 --follow $(docker ps -lq)
$ docker stop 076 (10s)
$ docker kill $(docker ps -q)
$ docker ps -a
$ docker run -ti --name clock jdeiviz/clock #Then detach with ^P^Q
$ docker attach clock
$ docker stop clock
$ docker start clock
```

Docker: Images

What is an image?

- It is a collection of files + metadata
- Conforms the filesystem of the container and is read only
- They consist of stacked layers
- Each layer adds, changes and deletes files
- The images share the layers to optimize the use of the disk, the transfer time and the use of memory (copy on write)
- Examples:
    - Ubuntu
    - JRE
    - Tomcat
    - Dependencies
    - Jar with the application
    - Configuration



Docker: Images

an NTT DATA Company

How are they created?

- The first images can be created in two ways:
  - From an empty image called scratch
  - With the docker import command you can import a tarball
- The other images can be created in two ways :
  - The docker **commit command** creates a new image with the status of the container
  - With the docker **build command** that interprets a Dockerfile file with the commands to execute. Allows repeatable compilation sequences.

Docker: Images

ALTEMISTA

everis

an NTT DATA Company

Namespace

- There are three namespaces for the images :
    - Officials: ubuntu, busybox, alpine, redis, nginx
    - User or organization: jdeiviz / clock, everis / steps
    - Hosted in a private registry: registry.example.com:5000/project/image
- In addition to the name it contains a label at the end. Ex: ubuntu:latest, ubuntu:17.04, ubuntu:precise
    - They usually define versions or variants of the image
    - If not indicated, it is used by default: latest
    - Latest is updated often
    - BE CAREFUL: You can update the image with the same label.

Docker: Images

ALTEMISTA

everis

an NTT DATA Company

Managing Images

```
$ docker images
$ docker search redis
$ docker pull debian:jessie
$ docker login
$ docker tag jdeiviz/clock jdeiviz/clock:1.0
$ docker push jdeiviz/clock jdeiviz/clock:1.0
$ docker rmi jdeiviz/clock:1.0
$ docker pull jdeiviz/clock:1.0
$ docker history jdeiviz/clock:1.0
$ docker image prune
```

Docker: Images

ALTEMISTA

everis

an NTT DATA Company

Our first image

```
$ docker run –it ubuntu bash
root@4a02a8f358f1:/# apt-get update && apt-get install -y figlet
root@4a02a8f358f1:/# exit
$ docker diff $(docker ps -lq)
$ docker commit $(docker ps -lq)
$ docker tag <id> figlet
$ docker run -it figlet
```

Docker: Images - Dockerfile

ALTEMISTA

everis

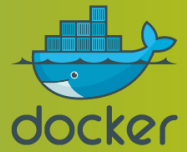
an NTT DATA Company

Automate the construction with Dockerfile

- Dockerfile is a recipe to build an image
- It contains a series of instructions that tell Docker how to build it
- The docker build command interprets Dockerfile and builds the image
- Each step of the execution creates a new image
- Only build if there are changes as it has a cache system

Most important commands :

- **FROM:** Indicates the base image from which we started.
- **RUN**: Executes commands and creates a new layer. It must be non-interactive.
- **COPY and ADD**: Add files from the host to the image.
- **ENV**: Add environment variables.
- **WORKDIR**: Modify the working directory from this line.
- **USER**: Modifies the user to execute commands from this line.
- **VOLUME**: Mount a directory of the team in a directory of the image
- **EXPOSE**: Exposes the indicated ports to be accessible from outside the container
- **CMD and ENTRYPOINT**: Indicates the command that will be executed when the container is started.



CMD and ENTRYPOINT

- **CMD**
    - Defines a default command when none is specified
    - It can be overwritten from the execution of the container
    - Only the last defined is valid
- **ENTRYPOINT**
    - Defines the base command and its parameters for the container
    - You can add parameters in the execution of the container
    - Only the last defined is valid
- **JUNTOS**
    - ENTRYPOINT defines the base command
    - CMD defines the default parameters
    - You can overwrite the parameters in the container execution

Docker: Images - Dockerfile

ALTEMISTA

everis

an NTT DATA Company

Our first image with Dockerfile

```
FROM ubuntu
RUN apt-get update && \
          apt-get install figlet
ENTRYPOINT ["figlet", "-f", "script"]
CMD ["hello world"]
```

```
$ docker build –t figlet .
$ docker run figlet
$ docker run figlet hola mundo
$ docker run –it –entrypoint bash figlet
```

Docker Networks

an NTT DATA Company

Introduction

- There are three initial networks :
    - Bridge: It is the default network, it provides communication between containers, host and internet
    - None: The containers associated with this network do not have access to the outside
    - Host: Add the container to the host's network stack. For performance
- Can be seen with docker network ls
- Containers expose ports with the parameter -p host:container
- To find the ip of a container *docker inspect --format '{{ .NetworkSettings.IPAddress }}' <container_id>*
- *docker port <container_id> <internal port>*

Docker Networks

an NTT DATA Company

Container networks

You can create custom networks :

- These networks provide insulation between containers
- A container can have more than one network
- Containers can only communicate if they are from the same network
- The docker daemon provides a DNS server that allows you to connect using the name of the container
- docker network create --driver bridge mi_red
- docker run –network my_network
- Currently there are two drivers, bridge and overlay. There are others via plugin like weable.

Docker Networks

an NTT DATA Company

Examples

```
$ docker run --name static-site -d -P dockersamples/static-site
$ docker port static-site
```

```
$ docker network create mem
$ docker run --name redis-server --network mem -d redis
$ docker run --network mem -it redis redis-cli -h redis-server -p 6379
```

Docker Volumes

ALTEMISTA

everis

an NTT DATA Company

What are they?

- It is a special directory within one or more containers that allows data sharing
- Can be shared and reused between containers
- Persist after stopping or erasing the container
- A host directory can be shared to be visible from the container
- Changes made to the volume are not included when updating the image
- They can be declared in two ways :
    - VOLUME in Dockerfile
    - Flag –v in docker run

Docker Volumes

ALTEMISTA

everis

an NTT DATA Company

Examples

```
$ docker run -it --name alpha -v /var/log ubuntu bash
root@99020f87e695:/# date >/var/log/now
$ docker run --volumes-from alpha ubuntu cat /var/log/now
```

```
$ docker volume create --name=website
$ docker run -d -p 8888:80 -v website:/usr/share/nginx/html -v logs:/var/log/nginx nginx
$ docker run -v website:/website -w /website -ti alpine vi index.html
```

```
$ docker run -d --name redis28 redis:2.8
$ docker run -ti --link redis28:redis alpine telnet redis 6379
SET counter 42
INFO server
SAVE
QUIT
$ docker stop redis28
$ docker run -d --name redis30 --volumes-from redis28 redis:3.0
```

```
$ docker run -ti --link redis30:redis
alpine telnet redis 6379

GET counter
INFO server
QUIT
```

everis
an NTT DATA Company

Install Docker and complete exercises

Docker Compose

ALTEMISTA

everis

an NTT DATA Company

What is it?

- It allows to configure multicontainers applications declaratively
- The file must be called docker-compose.yml
- The command docker-compose up, allows us to define in a yml file all the instructions and options necessary to start the containers that make up our application
- The command must be executed in the same path as the yml file
- If there are dependencies between the different containers that are started, docker compose is responsible for managing those dependencies by lifting the containers in order

Docker Compose

ALTEMISTA

everis

an NTT DATA Company

Example of docker-compose.yml

```
version: "2"

services:
  www:
     build: www
  ports:
     - 8000:5000
  user: nobody
  environment:
     DEBUG: 1
  command: python counter.py
  volumes:
     - ./www:/src
redis:
    image: redis
```

Docker Swarm

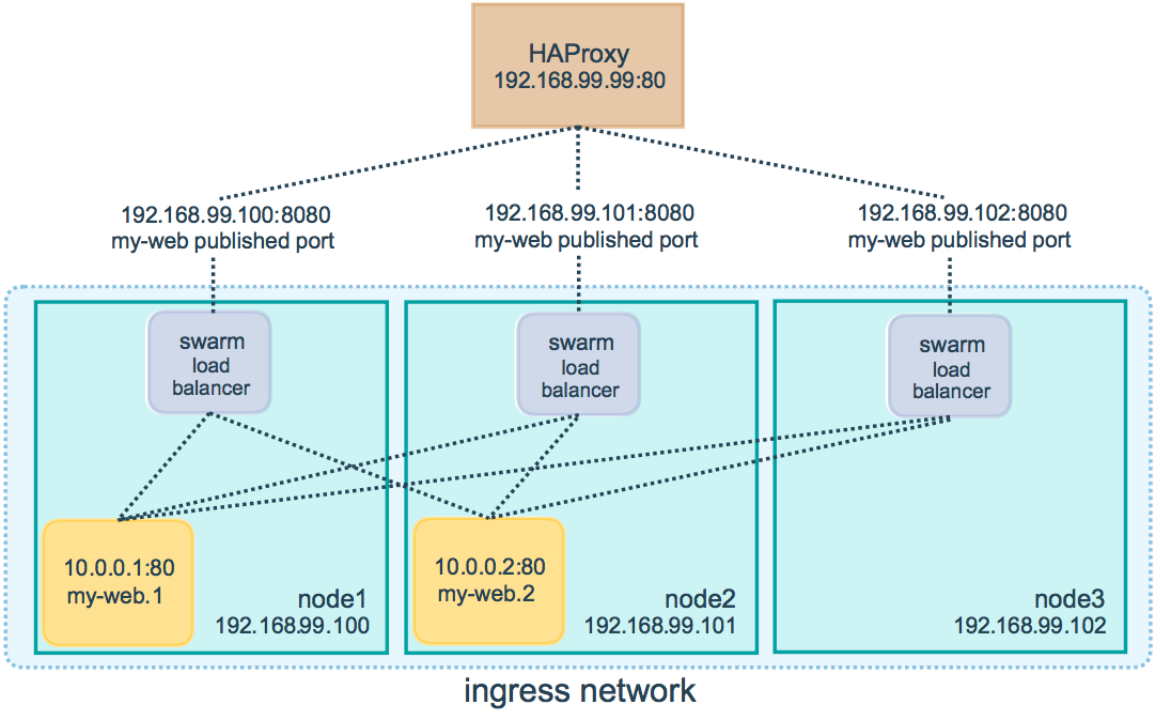
- Container Orchestrator integrated with Docker Engine
- Decentralized design
- Declarative model for the creation of services
- Scaled services in a simple way
- Always reconcile the desired state
- Multi-host networking
- Service discovery
- Load balancing
- Safe by default

an NTT DATA Company

Docker Swarm

Aspects	Features
Easy to start	• Create cluster definition • Join hosts to cluster • Create cluster manager
Management	• Swarm API. Easy and compatible with Docker Remote API. • Administration based in common Docker commands. • Small learning curve
Scheduling	• Support for tagging. Allow containers are only deployed in tagged cluster nodes.

Docker Swarm



Docker

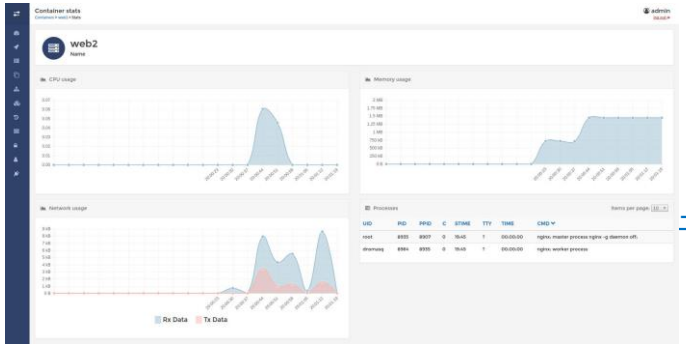
ALTEMISTA

everis

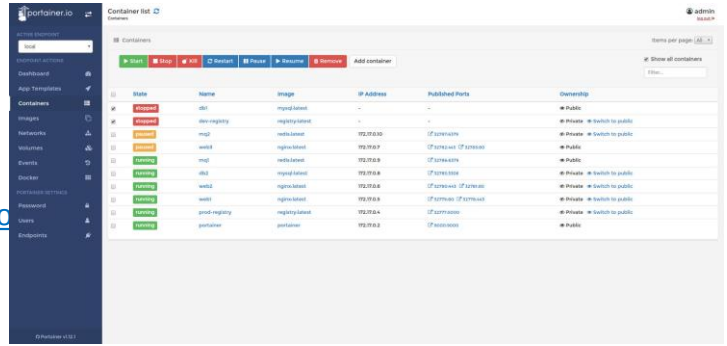
an NTT DATA Company

portainer.io

- Tool with interface (GUI) for the management of Docker containers.
- Docker container.
- Free. Creative Commons License (Commercial use).
- Compatible with Windows, Linux and MacOS.



-po



an **NTT DATA** Company

everis.com

Consulting, IT & Outsourcing Professional Services