



DOCKER + KUBERNETES

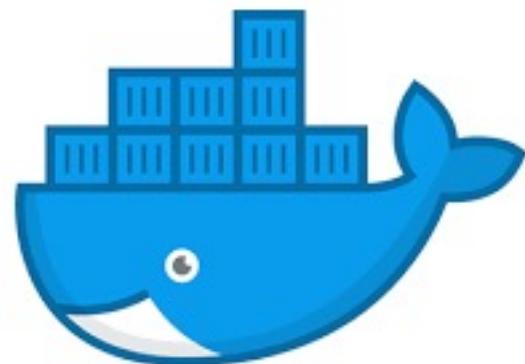
Profesor: Ginés Carrascal de las Heras



DOCKER + KUBERNETES

Contenedores

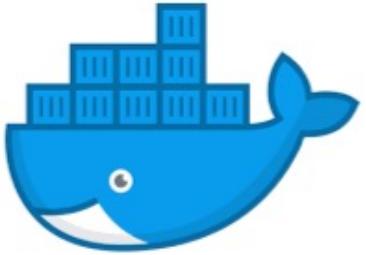
Fundamentos: Docker
Orquestación: Kubernetes





DOCKER + KUBERNETES

Conceptos básicos de contenedores



DOCKER

Conceptos fundamentales de Docker.

Ejecución de contenedores

Mapeo de puertos

Almacenamiento:

Volúmenes

Creación de imágenes:

Docker files, capas, variables

Gestión de imágenes:

Container Registry

Docker composer

Practica Docker con un modelo IA

Despliegue en cloud



DOCKER + KUBERNETES



Orquestación de Contenedores

KUBERNETES

Conceptos básicos de Kubernetes

Kubernetes en la nube

Pods

Despliegues declarativos en Kubernetes.

Servicios

Configuración: ConfigMaps y Secretos

Ingress

Healthchecks

Conceptos avanzados

Helm

Prometheus y Grafana

Open Shift



UNIDAD TEMATICA

CONCEPTOS BÁSICOS DE CONTENEDORES: DOCKER



DOCKER

Contenedores ligeros y portables para las aplicaciones





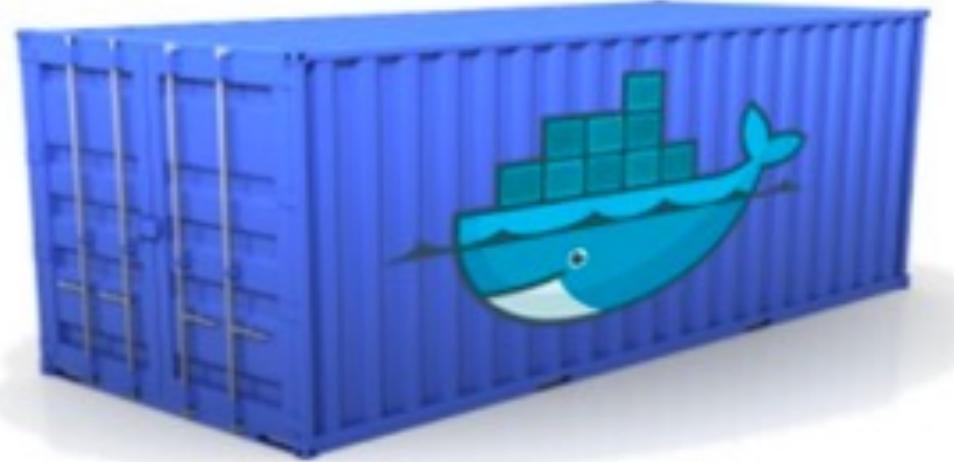
CAPITULO

Conceptos fundamentales de Docker



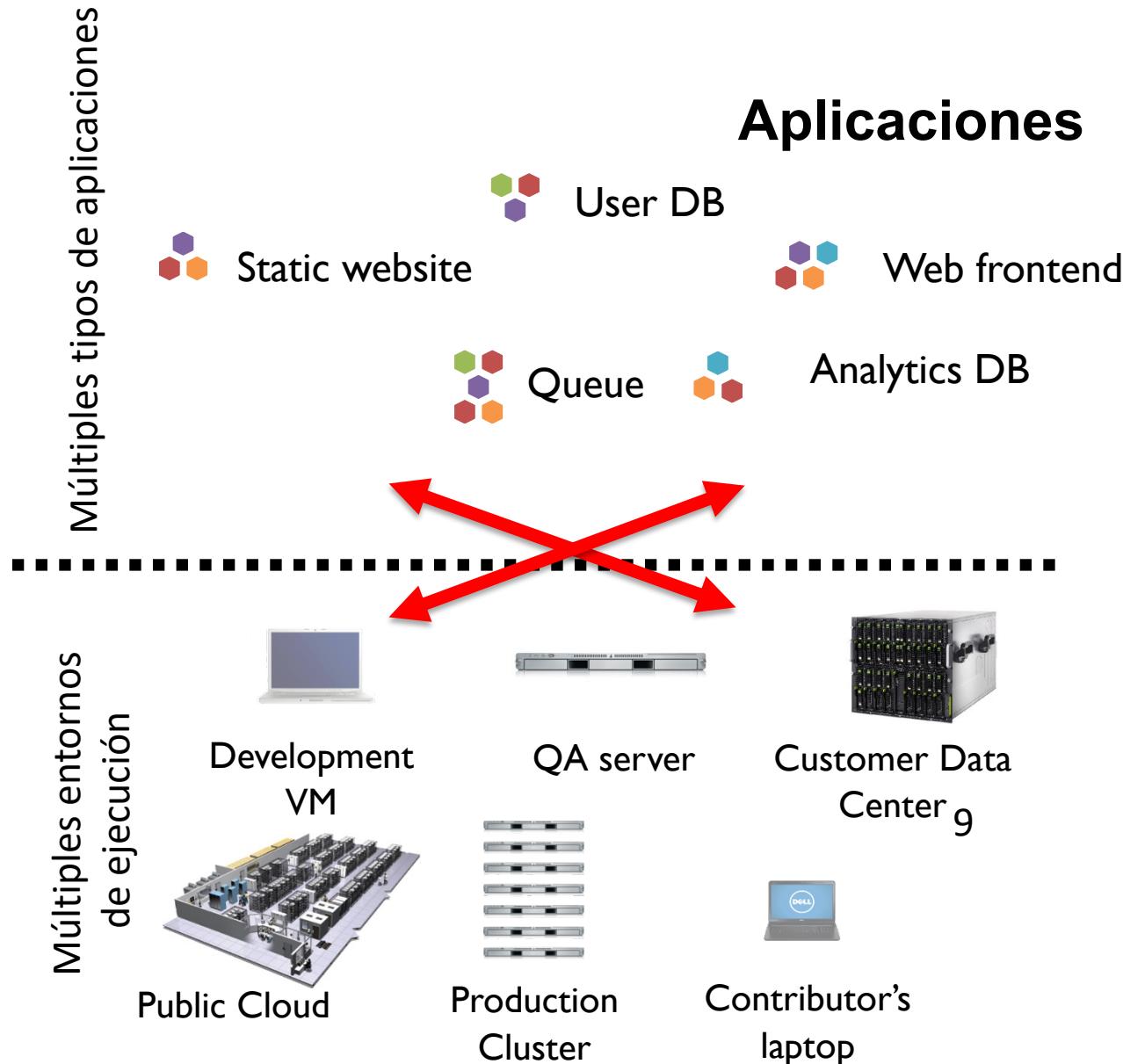
CONCEPTOS FUNDAMENTALES DE DOCKER

CONTENEDOR





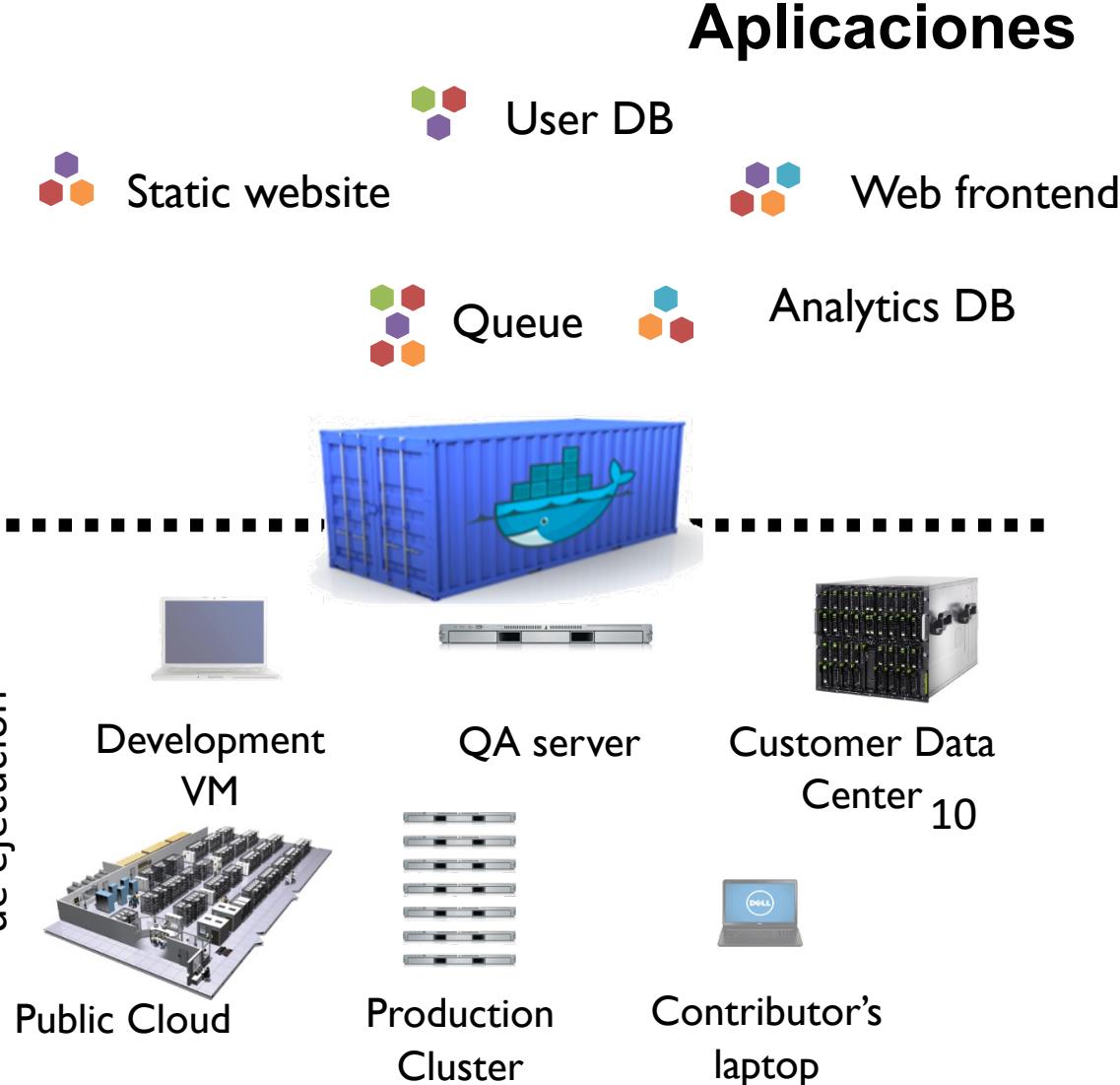
CONCEPTOS FUNDAMENTALES DE DOCKER





CONCEPTOS FUNDAMENTALES DE DOCKER

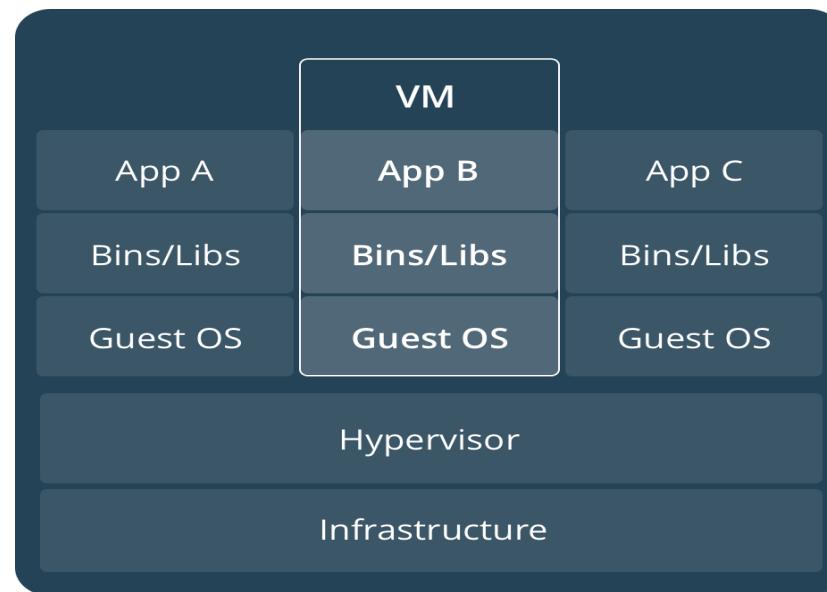
Múltiples tipos de aplicaciones





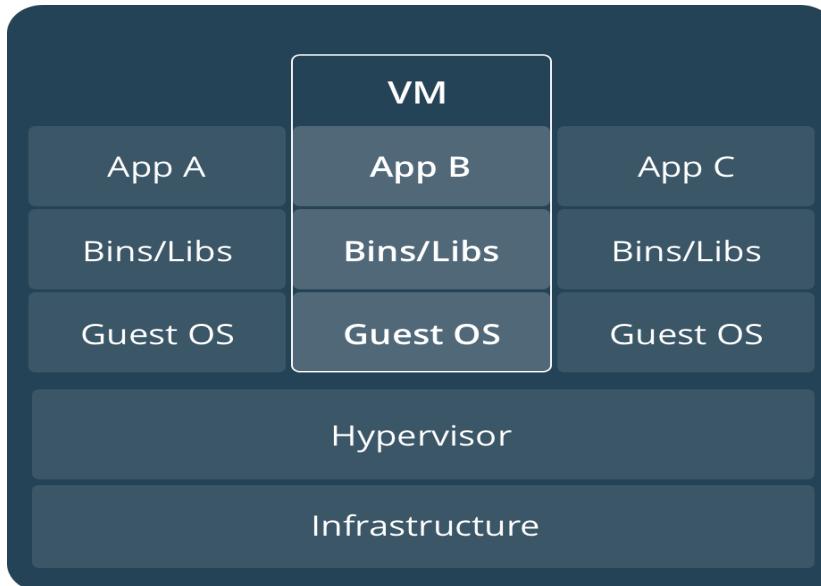
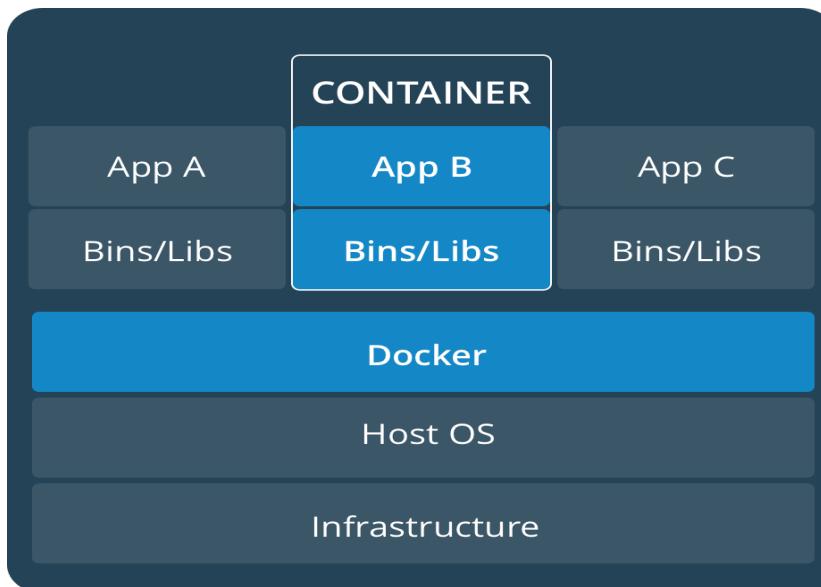
CONCEPTOS FUNDAMENTALES DE DOCKER

Contenedor vs Máquina virtual



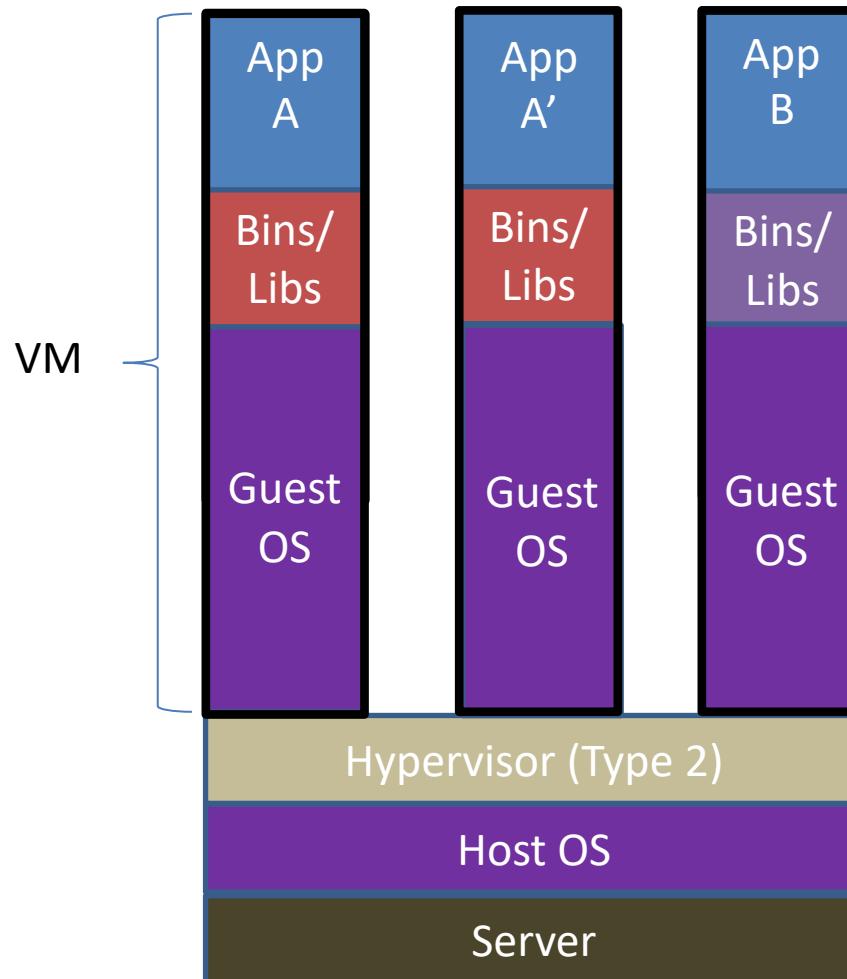


CONCEPTOS FUNDAMENTALES DE DOCKER



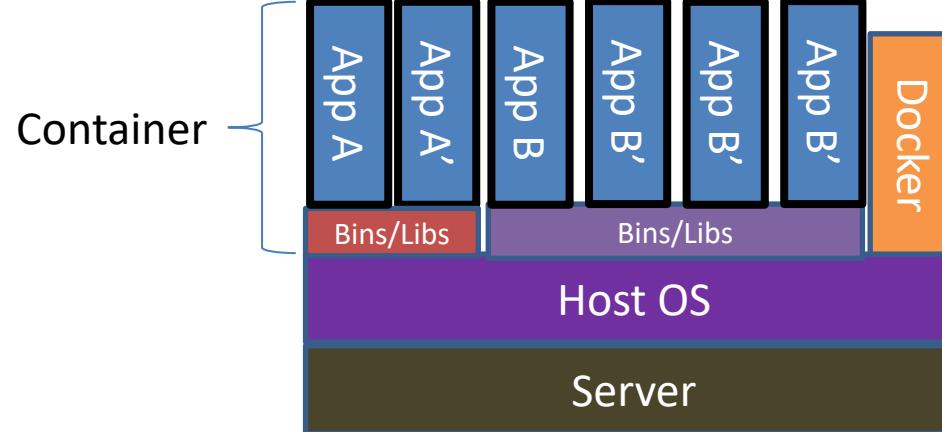


CONCEPTOS FUNDAMENTALES DE DOCKER





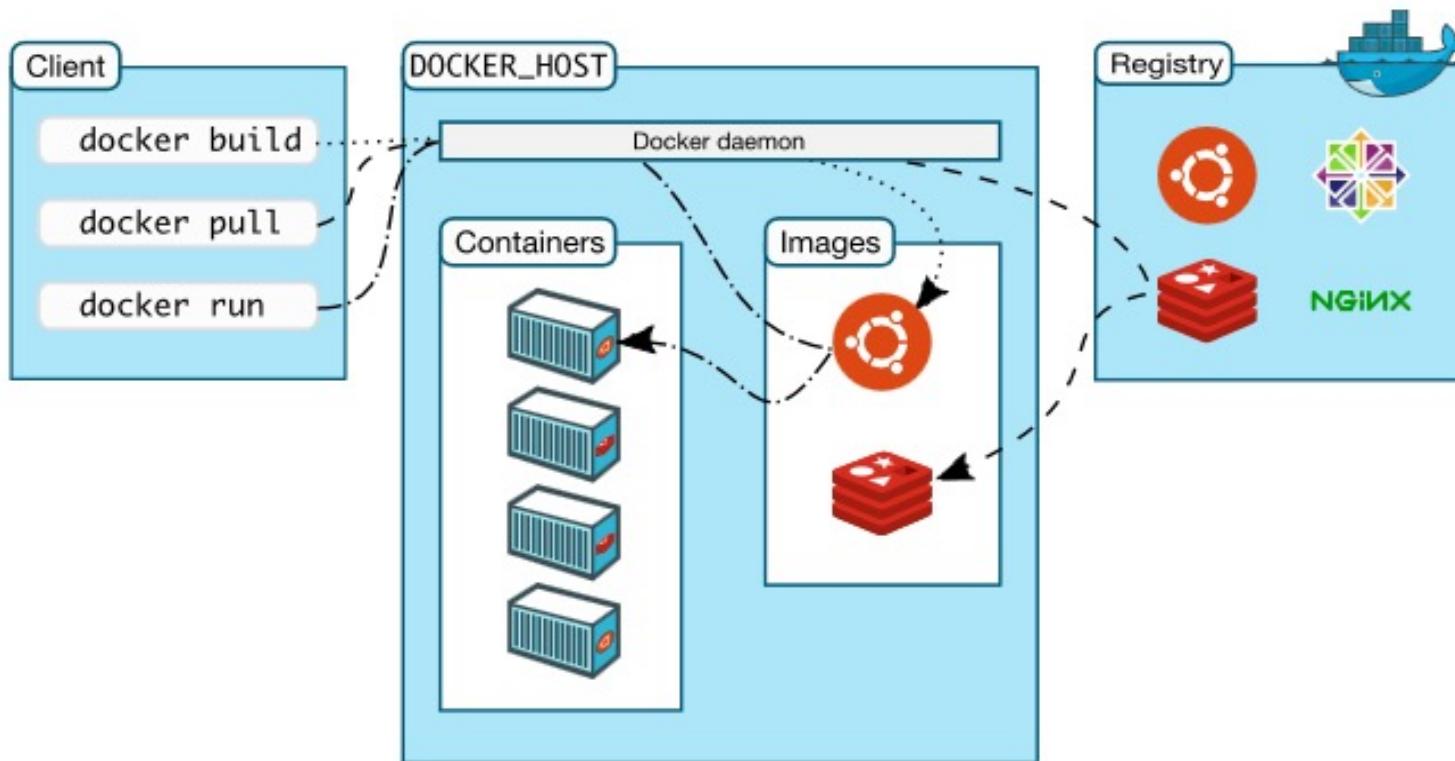
CONCEPTOS FUNDAMENTALES DE DOCKER





CONCEPTOS FUNDAMENTALES DE DOCKER

ARQUITECTURA





CONCEPTOS FUNDAMENTALES DE DOCKER

CASOS DE USO

CI/CD

DEVOPS

BIG DATA

OPTIMIZACION DE INFRAESTRUCTURA



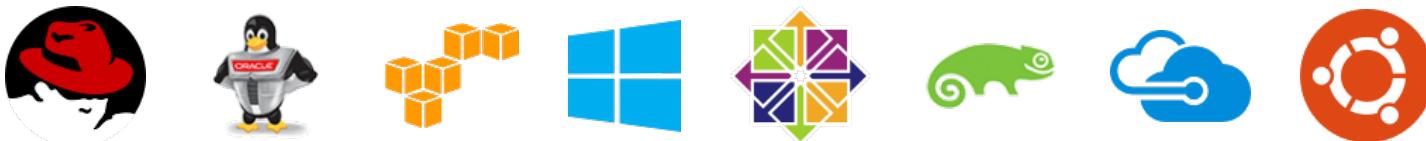
CONCEPTOS FUNDAMENTALES DE DOCKER

Versiones

Docker CE :

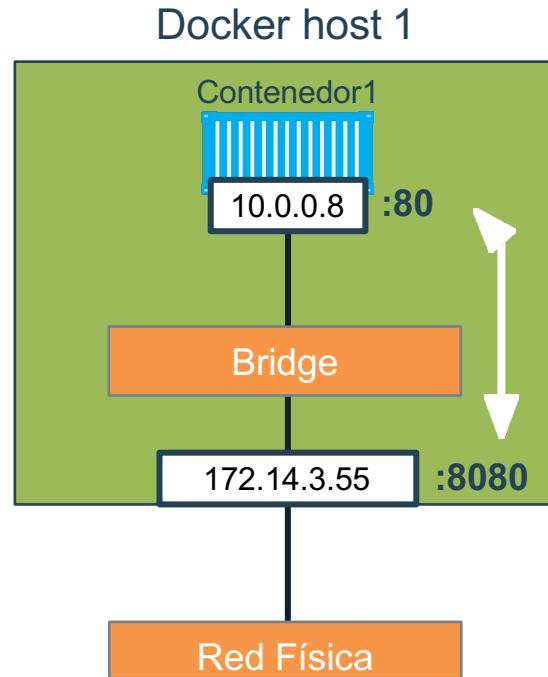


Docker EE :





MAPEO DE PUERTOS



Host port

Container port

```
$ docker container run -p 8080:80 ...
```



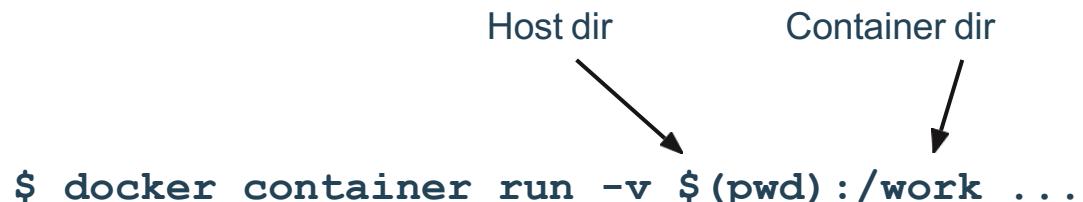
ALMACENAMIENTO

Volúmenes

Montar un directorio del host en un punto concreto del contenedor

Compartir datos entre contenedores

Datos persistentes tras apagar el contenedor





IMÁGENES

DOCKERFILE

Instrucciones para crear una imagen

Muy parecido a los commandos “nativos”



IMÁGENES

DOCKERFILE

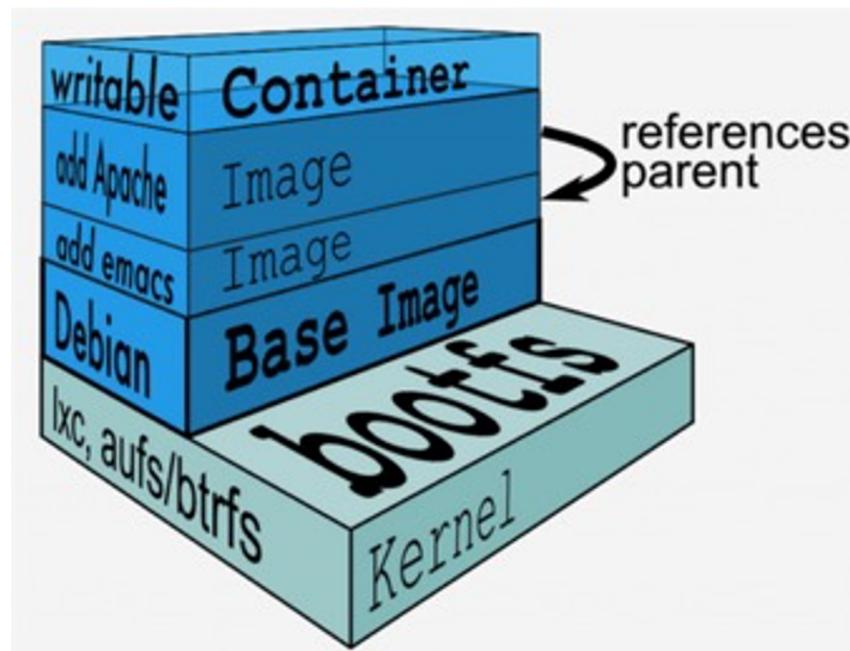
```
👉 Dockerfile ✘  
1  # Create image based on the official Node 6 image from dockerhub  
2  FROM node:latest  
3  
4  # Create a directory where our app will be placed  
5  RUN mkdir -p /usr/src/app  
6  
7  # Change directory so that our commands run inside this new directory  
8  WORKDIR /usr/src/app  
9  
10 # Copy dependency definitions  
11 COPY package.json /usr/src/app  
12  
13 # Install dependecies  
14 RUN npm install  
15  
16 # Get all the code needed to run the app  
17 COPY . /usr/src/app  
18  
19 # Expose the port the app runs in  
20 EXPOSE 4200  
21  
22 # Serve the app  
23 CMD ["npm", "start"]
```



IMÁGENES

SISTEMA DE ARCHIVOS EN PILA

Importante optimizar el DOCKERFILE





IMÁGENES

COMANDOS: CONTENEDORES

```
$ docker container run -d -p 5000:5000 --name node node:latest
```

```
$ docker container ps
```

```
$ docker container stop node(or <container id>)
```

```
$ docker container rm node (or <container id>)
```



IMÁGENES

COMANDOS: IMÁGENES

```
$ docker image pull node:latest
```

```
$ docker image ls
```

```
$ docker image rmi (or <image id>)
```

```
$ docker build -t node:2.0 .
```

```
$ docker image push node:2.0
```



CONTAINER REGISTRY

¿QUÉ ES UN REGISTRO?

Un lugar centralizado para almacenar y distribuir imágenes

Almacena las capas y las instrucciones de cómo componer las imágenes

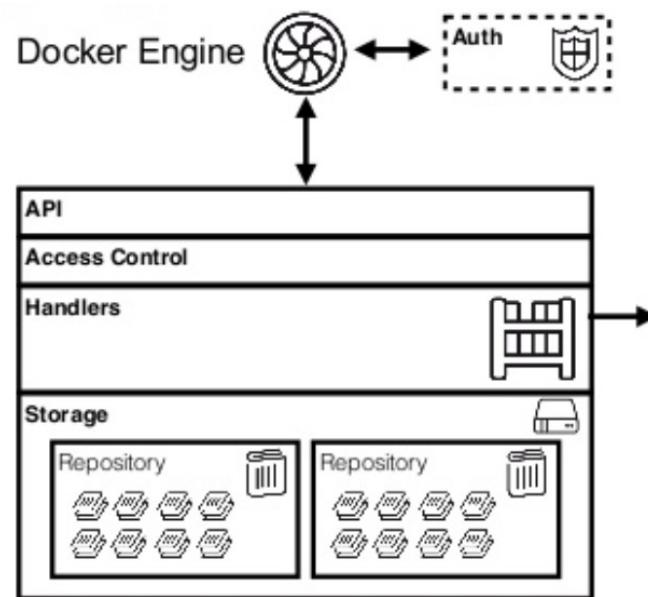
Implementa un API común a todos los clientes de Docker



CONTAINER REGISTRY

DOCKER REGISTRIES API V2

Todo el contenido está empaquetado en
Repositorios





CONTAINER REGISTRY

DOCKER-HUB



CONTAINER REGISTRY



Search for great content (e.g., mysql)

Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications.



CONTAINER REGISTRY

A screenshot of the Docker Hub website. At the top, there's a search bar with the placeholder "Search for great content (e.g., mysql)". Below the search bar are navigation links: "Explore", "Repositories", and "Organization".

dockerhub Search for great content (e.g., mysql)

Explore Repositories Organization

Docker

Containers

Plugins

Filters

1 - 25 of 5,986,880 available images.

Images

- Verified Publisher i
- Official Images i
Official Images Published By Docker

Categories i

- Analytics
- Application Frameworks
- Application Infrastructure
- Application Services
- Base Images
- Databases
- DevOps Tools
- Featured Images
- Messaging Services
- Monitoring
- Operating Systems
- Programming Languages
- Security



mongo

Updated 11 hours ago

MongoDB document databases provide high availability and easy scalability.

Container

Windows

Linux

IBM Z

x86-64

ARM 64

Databases



busybox

Updated 11 hours ago

Busybox base image.

Container

Linux

ARM

386

ARM 64

mips64le

x86-64

IBM Z

Pov



alpine

Updated 11 hours ago

A minimal Docker image based on Alpine Linux with a complete package index an

Container

Linux

PowerPC 64 LE

386

IBM Z

ARM

ARM 64

x86-64



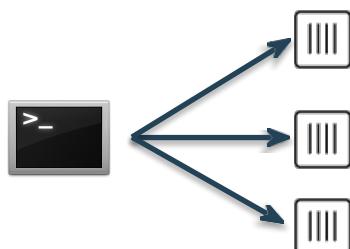
DOCKER COMPOSE

APLICACIONES MULTICONTENEDOR

Construir y ejecutar cada uno de los contenedores

Manualmente conectar los contenedores

Tener cuidado con las dependencias y el orden de arranque





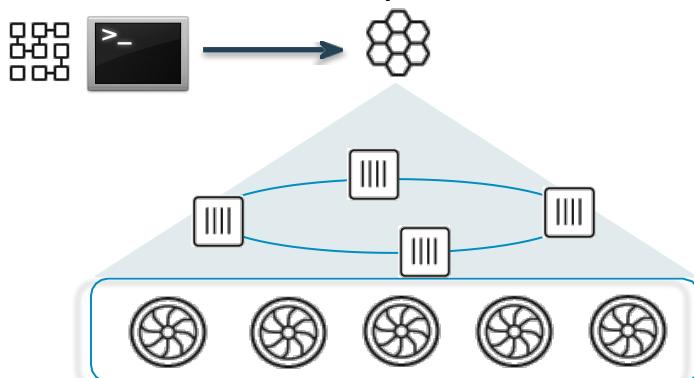
DOCKER COMPOSE

APLICACIONES MULTICONTENEDOR

Definir la aplicación multicontenedor en un fichero: **compose.yaml**

Un solo comando para desplegar la aplicación

Gestiona todas las dependencias



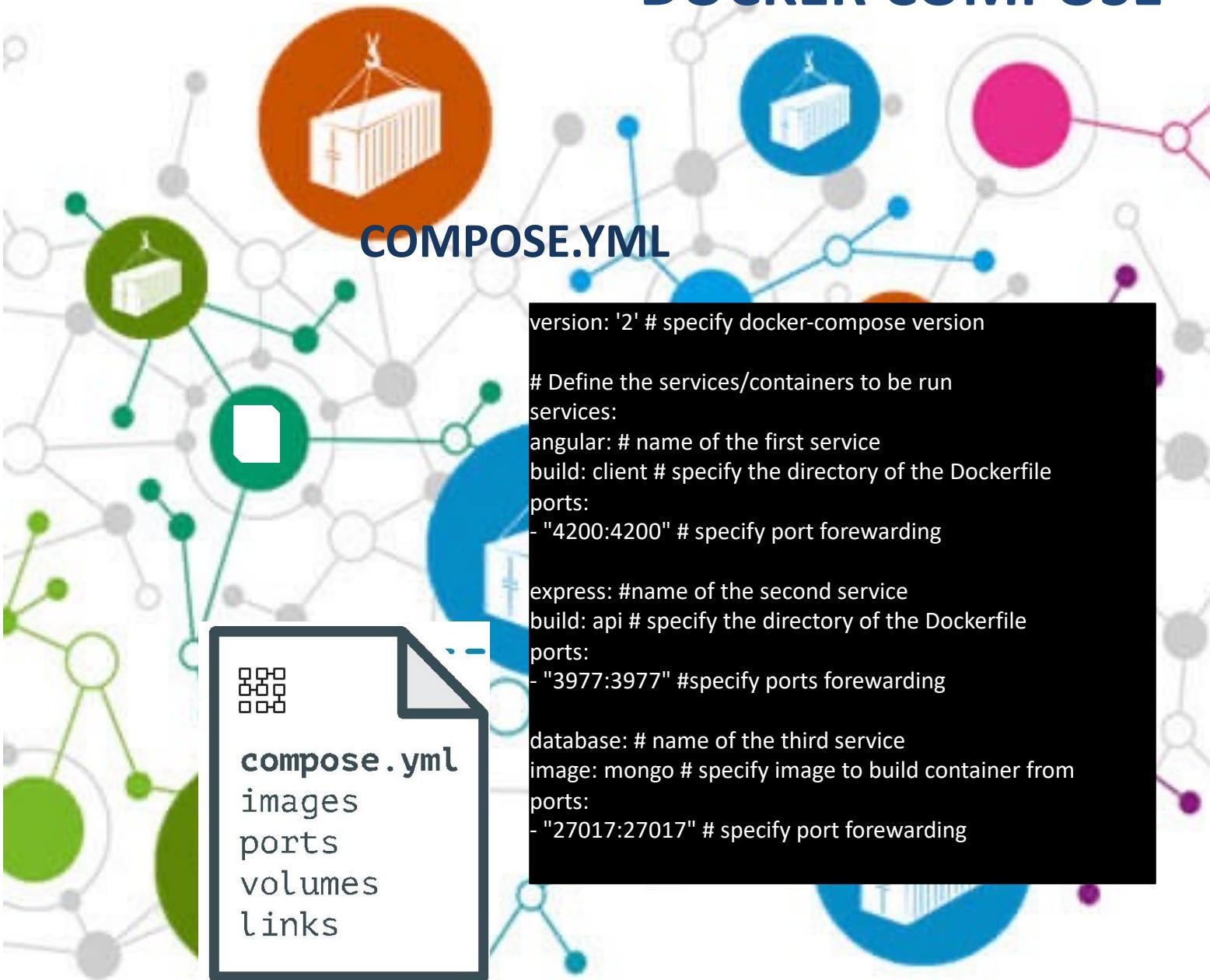


DOCKER COMPOSE





DOCKER COMPOSE





DOCKER CON UN MODELO DE AI

CASOS DE USO

CI/CD

DEVOPS

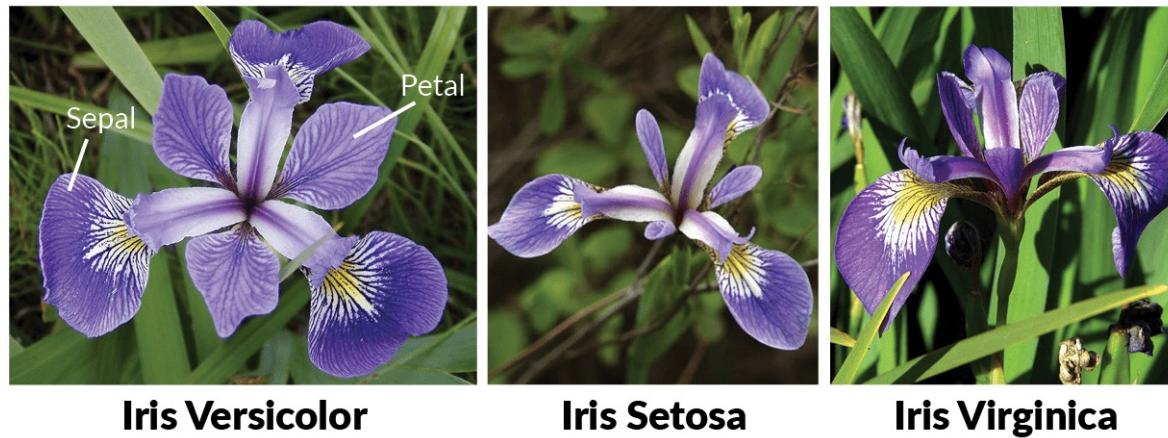
BIG DATA

OPTIMIZACION DE INFRAESTRUCTURA



DOCKER CON UN MODELO DE AI

CASOS DE USO





DOCKER CON UN MODELO DE AI

ESTRUCTURA DE LA APLICACIÓN

```
└── microsvc
    ├── app.py
    └── templates
        └── home.html
```

```
sudo apt install -y python3-pip
```





DOCKER CON UN MODELO DE AI

CREAR UN DOCKERFILE

```
# We always start from an existing container
FROM python:3.9.2

# Copy dependency lists into container
COPY requirements.txt .

# Install dependencies
RUN pip install -r requirements.txt

# Copy our code into the container
COPY app.py .
COPY iris_trained_model.pkl .

# Copy the HTML page templates directory
COPY templates templates

# Our code runs on port 5000, so allow access
EXPOSE 5000

# Set the FLASK_APP environment variable
ENV FLASK_APP app.py

# This is the command that is executed when the container starts
# Note we've added --host=0.0.0.0 as by default only local users would
# be able to access the application.
CMD [ "flask", "run", "--host=0.0.0.0" ]
```



DOCKER CON UN MODELO DE AI

CONSTRUIR LA IMAGEN

```
$ docker build -t IMAGE_NAME:TAG .
```

ó

```
$ docker build -t USERNAME/IMAGE_NAME:TAG .
```

Y probarla:

```
$ docker run -p 5000:5000 USERNAME/IMAGE_NAME:TAG
```



DOCKER CON UN MODELO DE AI

Iris Flower Category

Sepal Length Cm Sepal Width Cm Petal Length Cm Petal Width Cm Predict

```
$ docker build -t IMAGE_NAME:TAG .
```

ó

```
$ docker build -t USERNAME/IMAGE_NAME:TAG .
```

Y probarla:

```
$ docker run -p 5000:5000 USERNAME/IMAGE_NAME:TAG
```



UNIDAD TEMATICA

ORQUESTACIÓN DE CONTENEDORES: KUBERNETES



KUBERNETES

Orquestador de contenedores entre múltiples servidores





KUBERNETES



Orquestación de Contenedores

KUBERNETES

Orquestación de conjuntos de contenedores
desplegados en diferentes servidores

- Colocación automática, red, despliegue,
escalado, roll-out/-back, A/B testing

Declarativo – not procedural

- Declarar el estado deseado, reconciliar con él
- Se auto repara

Portabilidad de las cargas de trabajo

- Abstracción de las peculiaridades de los proveedores de cloud
- Multiples entoros de ejecución para los contenedores



CAPITULO

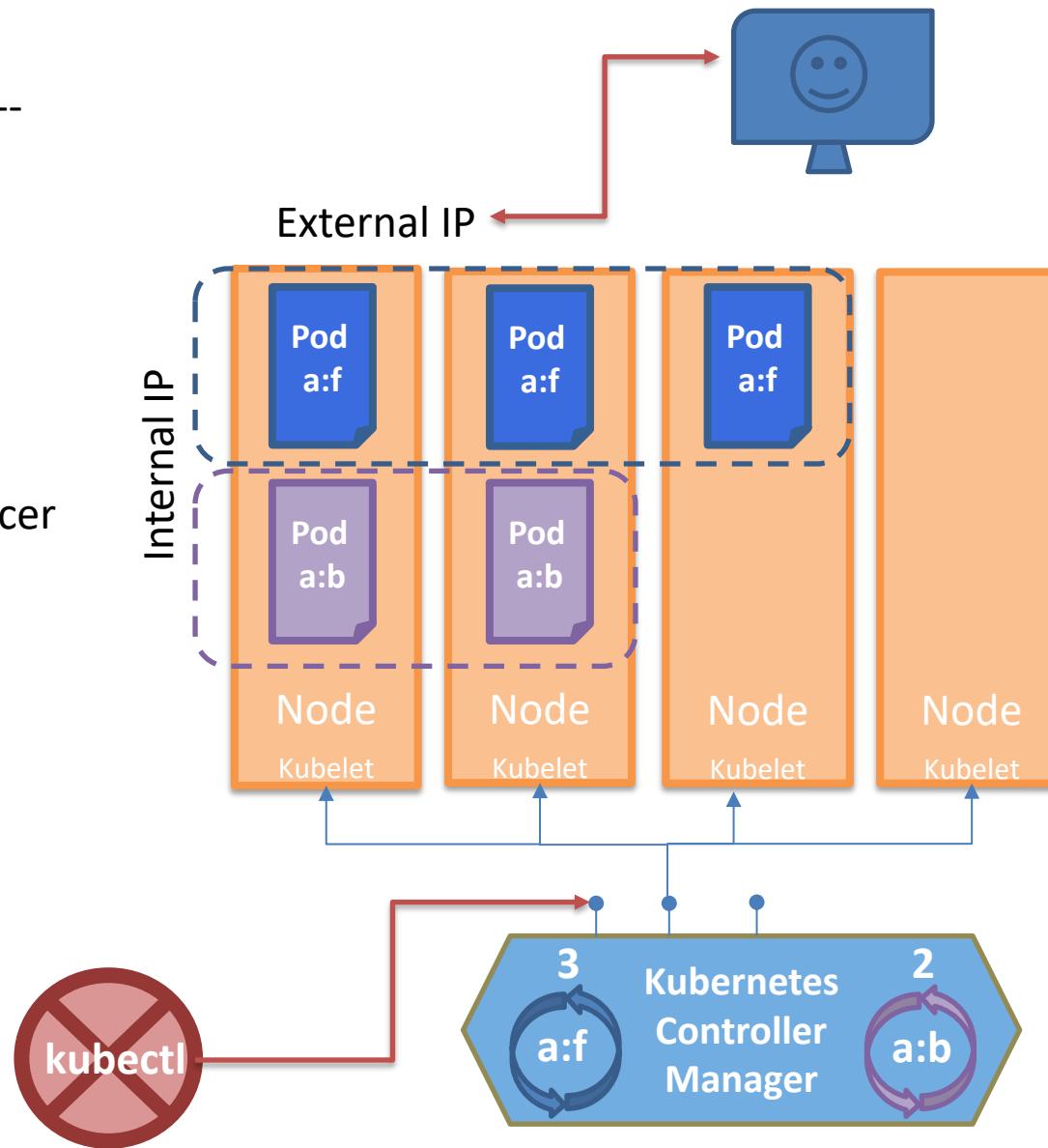
Conceptos básicos de Kubernetes



```
kind: Deployment  
selector: {a:f}  
replicas: 3  
template:  
  --kind: Pod----  
    image: f:v1  
    labels:  
      a:f
```

```
-----  
kind: Service  
selector: {a:f}  
type: LoadBalancer
```

KUBERNETES





CAPITULO

Pods



KUBERNETES

POD



Un Pod es una colección de contenedores que se ejecuta en un servidor

Representa una unidad lógica para agrupar aplicaciones

Los VOLUMENES pueden compartirse entre aplicaciones del mismo Pod

Los Pods son efímeros, y no pueden moverse a otro servidor



KUBERNETES

SERVICIO



Un servicio es una abstracción que define un conjunto lógico de Pods y la política de acceso a los mismos

El conjunto de Pods involucrados en un servicio se determina mediante un selector de etiquetas

Un servicio reserve un puerto TCP o UDP

Permite a diferentes aplicaciones Kubernetes conectarse sin que necesiten conocer las direcciones IP



CAPITULO

Replication Controllers y Replica Sets



KUBERNETES



CONTROLADOR DE REPLICAS

Un Controlador de replicas se asegura de que el número establecido de “copias” de un pod está en ejecución en cada momento

Utilizan Pod Templates para crear los Pods

Utilizan las etiquetas para vigilar y mantener el número correcto de réplicas



CAPITULO

Despliegues declarativos en Kubernetes



KUBERNETES

DEPLOYMENT

Un Controlador de Deployment proporciona actualizaciones declarativas para Pods y ReplicaSets

NO SON COMANDOS:
El controlador comprueba el estado actual y lo cambia por el estado deseado de forma controlada

Las modificaciones en los Pods y Replicas siempre deberían hacerse mediante Deployments



CAPITULO

Creación y uso de Namespaces



KUBERNETES

ESPACIOS DE NOMBRES

Los Namespaces en Kubernetes ayudan a que diferentes proyectos, equipos o clients compartan un mismo cluster

Proporciona un mecanismo para asignar diferentes políticas de autorización a secciones del cluster



CAPITULO

Configuración mediante ConfigMaps



KUBERNETES

CONFIG MAPS

Un ConfigMap es un almacén de propiedades de tipo clave-valor.

Los Pods pueden acceder a la información como variables de entorno, argumentos de línea de comandos, o ficheros de configuración en un volumen

No proporcionan encriptación

Su objetivo es separar la configuración del Código de las aplicaciones



CAPITULO

Secretos en Kubernetes



KUBERNETES

SECRETS

Los secretos permiten almacenar información sensible como passwords, tokens OAuth o claves ssh

Los Pods pueden acceder a la información como variables de entorno, argumentos de línea de comandos, o ficheros de configuración en un volumen

Proporcionan encriptación, pero para tener seguridad completa necesitan encriptación de los dispositivos



CAPITULO

Ingress



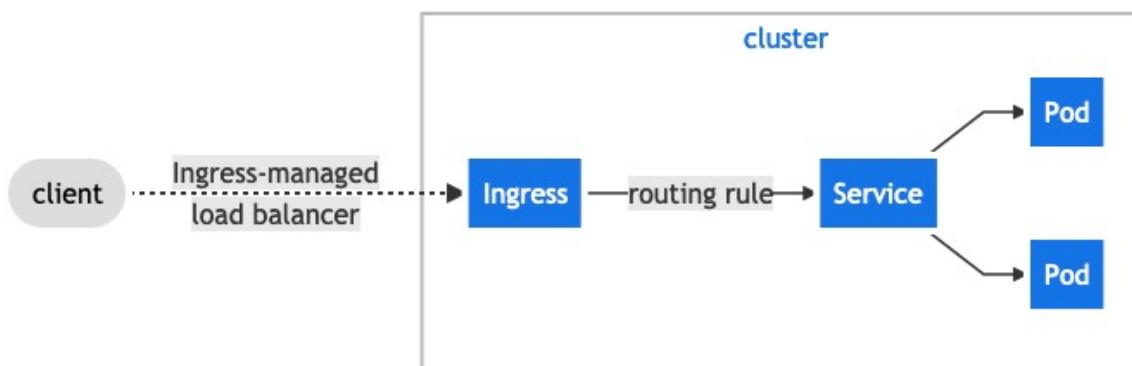
KUBERNETES

INGRESS

Un Ingress gestiona el acceso externo a los servicios de un cluster

Típicamente utiliza protocolo HTTP

Ingres puede proporcionar balanceo de carga, gestión de SSL y “virtual hosting” en base a nombres





CAPITULO

Dependencias



KUBERNETES

INIT CONTAINERS

El propósito es separar los Pods de su lógica de inicialización

Un **Init container** es un contenedor que se inicia y ejecuta *antes* del resto de los contenedores, en el mismo pod

Si el Init container falla, el pod completo es reinicializado

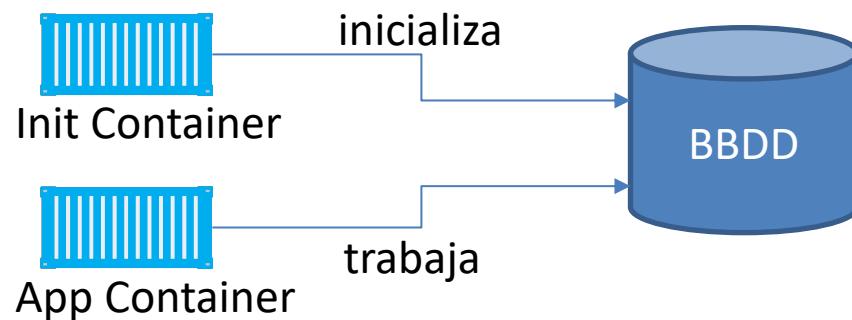


KUBERNETES: INIT CONTAINERS

INICIALIZAR UNA DDBB

El init container se conecta al motor de base de datos y crea las tablas y datos iniciales

La aplicación se inicia y trabaja contra una base de datos adecuada



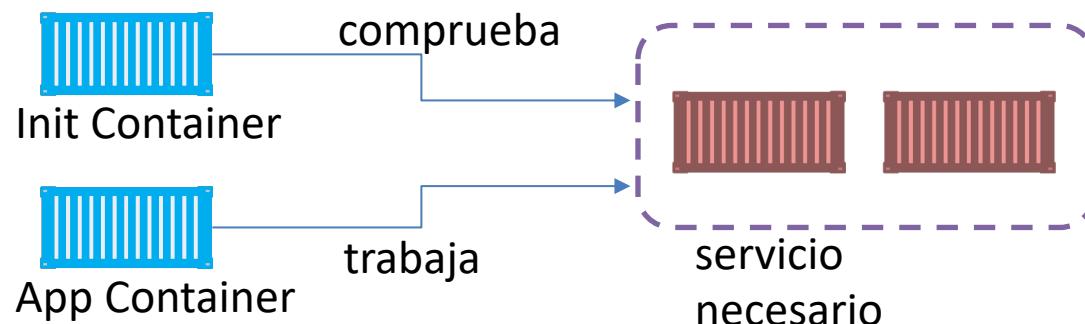


KUBERNETES: INIT CONTAINERS

Esperar a otro servicio

El init container comprueba
periodicamente la existencia de otro
servicio hasta que lo encuentra

La aplicación se inicia y trabaja contra
una base de datos adecuada





CAPITULO

Healthchecks: Liveness, Readiness y Startup Probes



KUBERNETES

HEALTHCHECKS

Probes son procesos que comprueban el estado de los contenedores para permitir tomar decisiones:

Kubelet utiliza **liveness probes** para decidir si debe reiniciar un contenedor

Kubelet utiliza **readiness probes** para decidir si un contenedor está listo para recibir peticiones

Kubelet utiliza **startup probes** para saber si un contenedor ya ha arrancado y se pueden utilizar en las otras *probes*



CAPITULO

Helm



KUBERNETES

HELM



Es conocido como el gestor de paquetes
de Kubernetes

Utiliza **Helm Charts** para agrupar
conjuntos de especificaciones (YAML) de
kubernetes

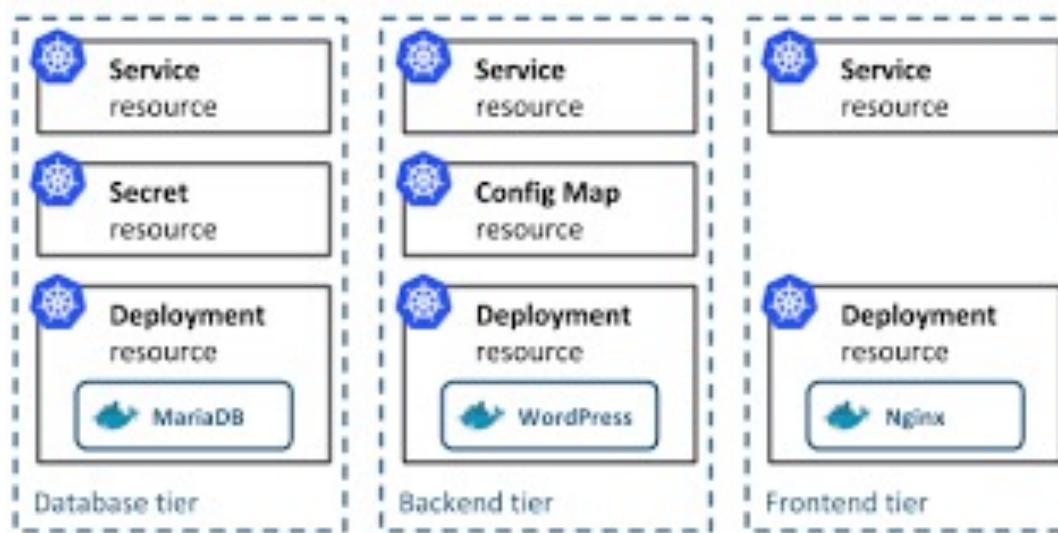
Un Chart puede depender de otros, y
Helm instala todo el arbol de
dependencias autométicamente con un
solo comando



KUBERNETES



Es conocido como el gestor de paquetes
de Kubernetes





CAPITULO

Prometheus y Grafana



KUBERNETES

Monitorización

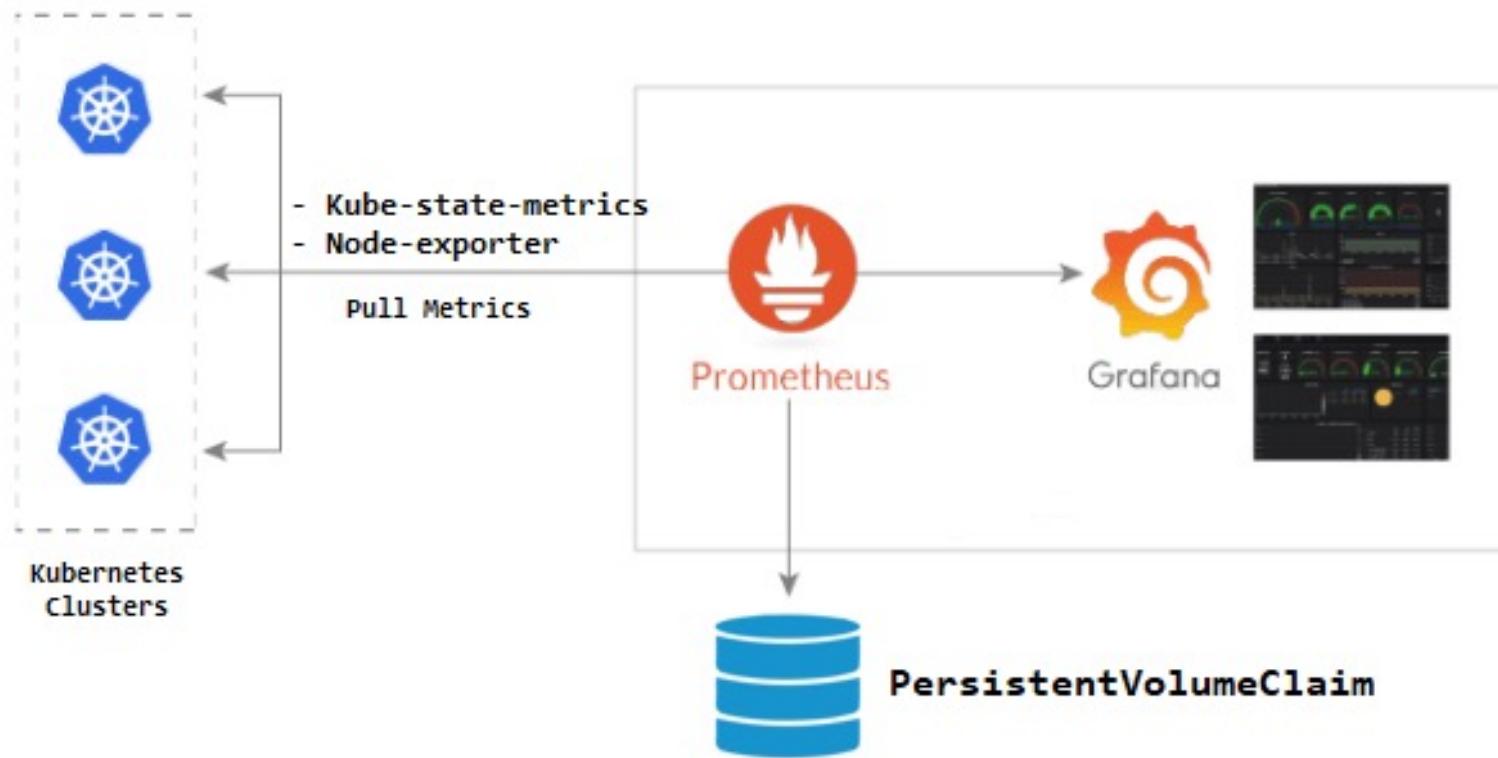
Prometheus hace peticiones a los diferentes elementos y guarda métricas de funcionamiento

Grafana crea dashboards visuals con la información recogida



KUBERNETES

Monitorización





Node



Zoom Out

Last 15 minutes

Refresh every 30s



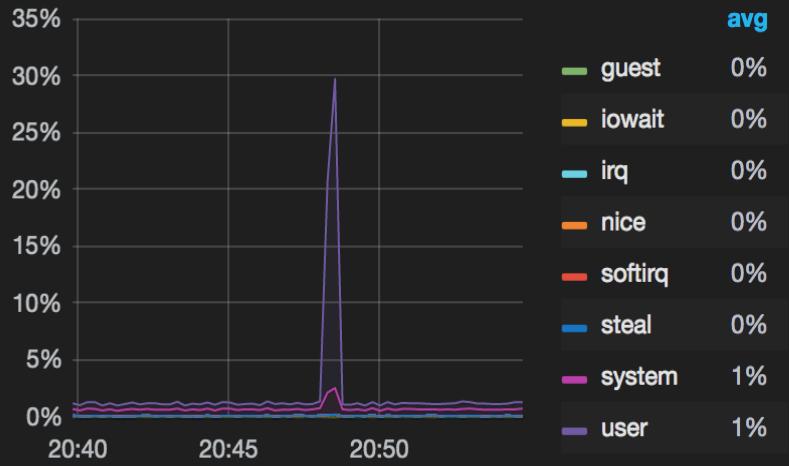
Environment: prod

Server Type: All

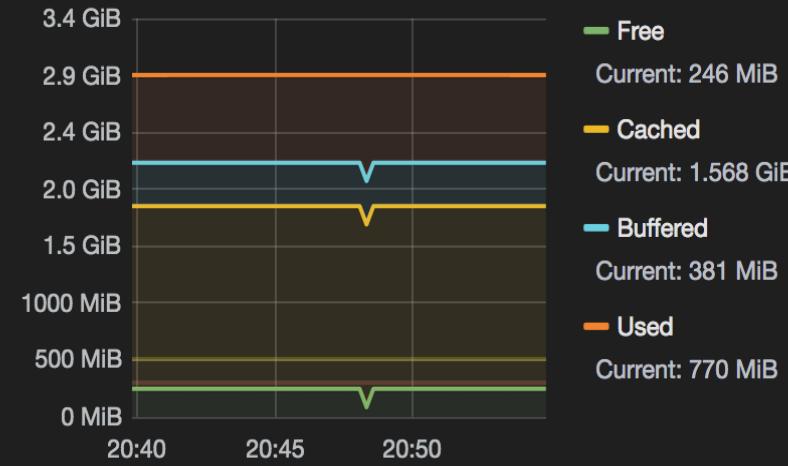
Node: rz-docker-registry

RZ-DOCKER-REGISTRY

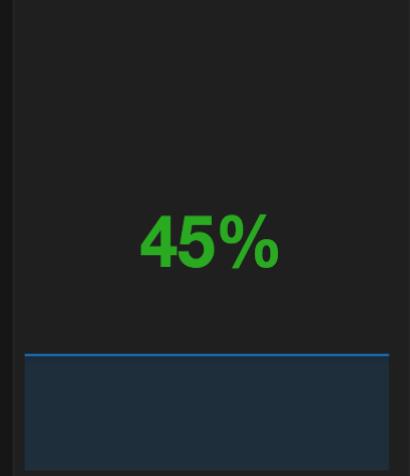
CPU



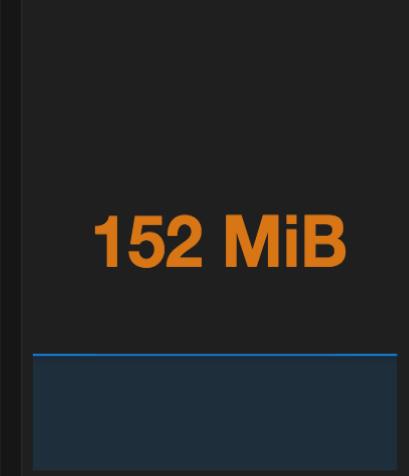
Memory



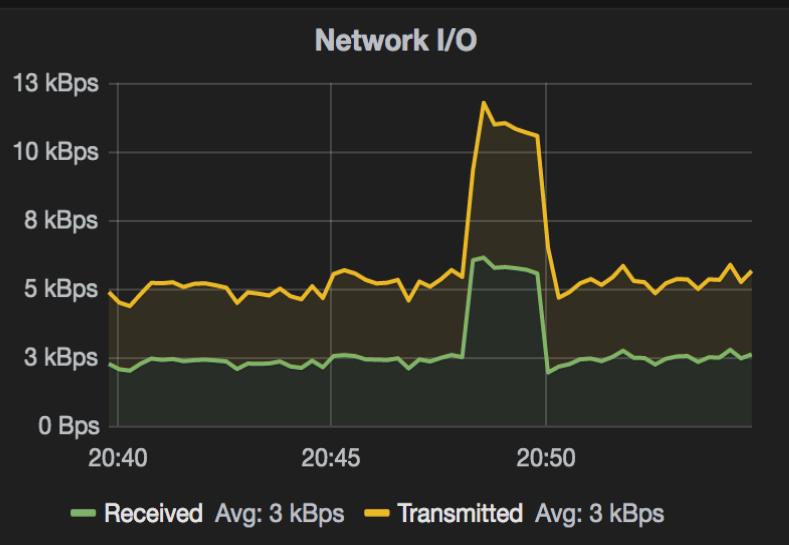
Remaining Disk Space

45%

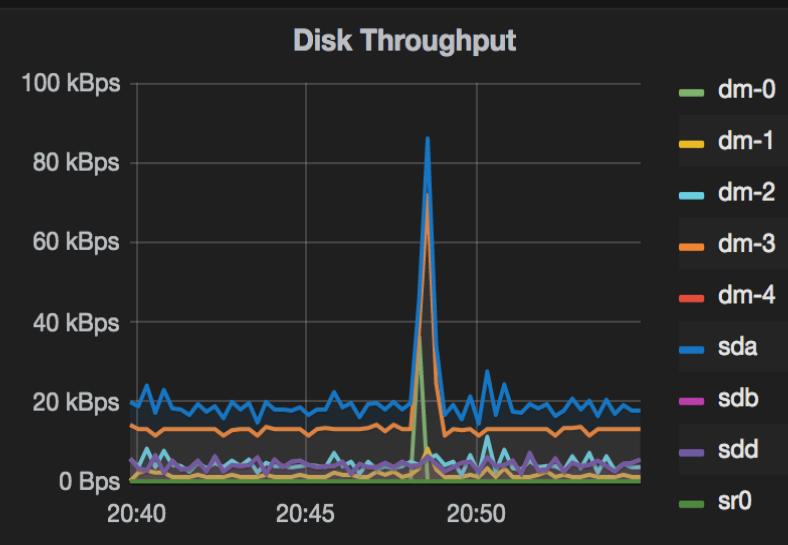
Swap

152 MiB

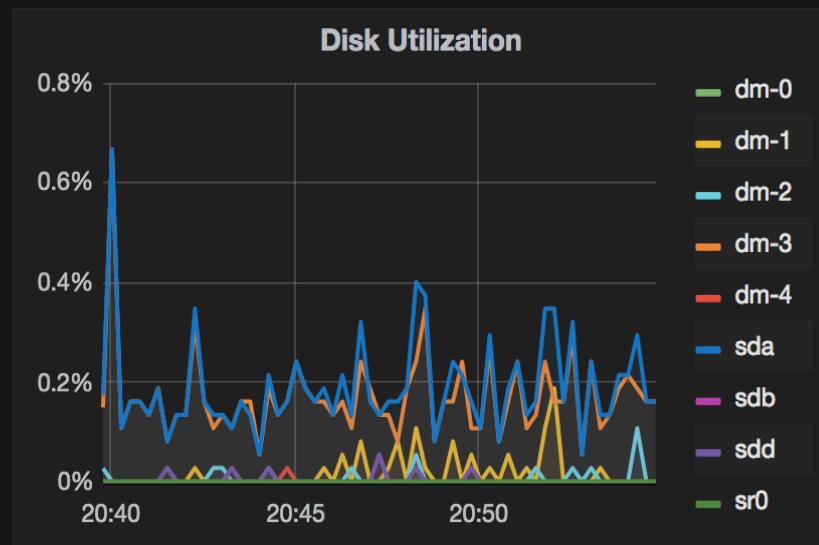
Network I/O



Disk Throughput



Disk Utilization





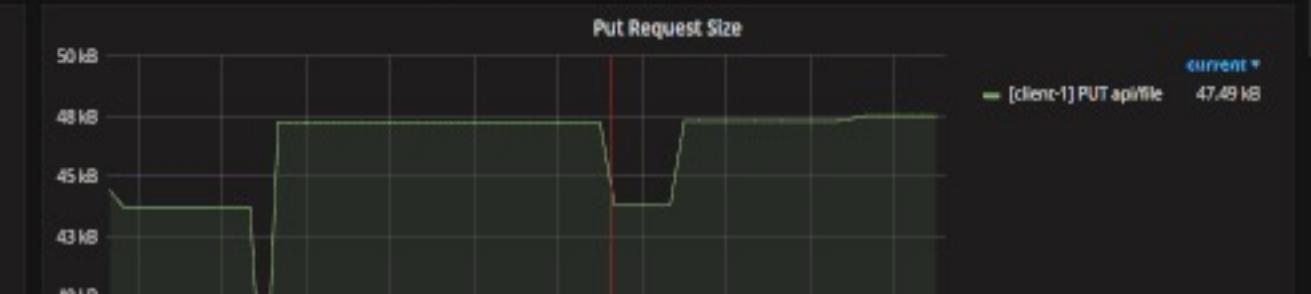
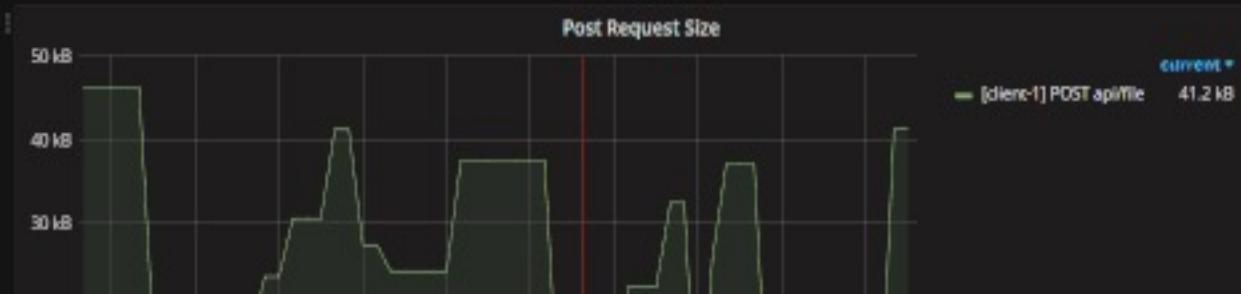
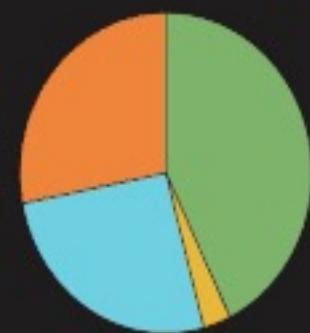


2017-03-18 21:48:15

- [client-1] GET api/satisfying: 29.95 rpm
- [client-1] GET api/frustrating: 29.46 rpm
- [client-1] GET api/tolerating: 27.63 rpm
- [client-1] GET api/randomstatusCode: 4.65 rpm
- [client-1] POST api/file: 2.79 rpm
- [client-1] PUT api/file: 2.68 rpm

49:30 21:40:40 21:40:50 21:50:00 21:50:10

Errors





CAPITULO

Open Shift



KUBERNETES

RED HAT OPEN SHIFT



Red Hat OpenShift es una plataforma de contenedores de Kubernetes empresarial

Centrada e la seguridad

Con operaciones automatizadas integrales que permite gestionar implementaciones de nube híbrida



RED HAT OPEN SHIFT

Red Hat OpenShift Container Platform

Home ▾ Project: default ▾

Projects Status Search Events

Catalog ▾ All Items Database

Developer Catalog Database

Installed Operators Integration & Delivery

Operator Hub OpenShift Optional

Operator Management Streaming & Messaging

Workloads Monitoring

Pods Logging & Tracing

Deployments Other

Deployment Configs

Stateful Sets

Secrets

Config Maps

Cron Jobs

Jobs

Filter by keyword...

INSTALL STATE

Installed (1)

Not Installed (5)

PROVIDER TYPE

Red Hat (0)

Certified (2)

Operator Hub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red H optional add-ons and shared services to your developers. Once installed, the capabilities prov a self-service experience.

6 items

 Couchbase Operator provided by MongoDB, Inc

The Couchbase Autonomous Operator allows users to easily deploy, manage, and monitor Couchbase Server in a Kubernetes environment. It provides a simple interface for managing Couchbase clusters, including creating, deleting, and updating configurations.

 MongoDB provided by MongoDB, Inc

The MongoDB Enterprise Operator makes it easy to deploy, manage, and monitor MongoDB databases in a Kubernetes cluster. It provides a simple interface for managing MongoDB instances, including creating, deleting, and updating configurations.

 PlanetScale Operator provided by Red Hat, Inc

PostgreSQL Enterprise Operator

provided by Red Hat, Inc



RED HAT OPEN SHIFT

OPENSIFT ORIGIN

My Project » Add to Project » Catalog » Node.js



Node.js

Build and run Node.js 6 applications on CentOS 7. For more information about using this builder image and OpenShift considerations, see <https://github.com/sclorg/s2i-nodejs-container/blob/master/6/README.md>

Version: 6

*** Name**
khs-api-example

Identifies the resources created for this application.

*** Git Repository URL**
`https://github.com/in-the-keyhole/khs-example-node-api`

Sample repository for nodejs: <https://github.com/openshift/nodejs-ex.git> [Try It ↑](#)

Show [advanced options](#) for source, routes, builds, and deployments.

Create **Cancel**



RED HAT OPEN SHIFT

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes the Red Hat logo and the text "Red Hat OpenShift Container Platform". A dropdown menu shows "Developer" and "Project: test Application: all applications". The main area displays a "Topology" view with three resources:

- A circular icon for a "mariadb" database, labeled "DC" (Deployment Config) with a red status indicator.
- A larger circular icon for a "java-sample" application, which contains a smaller icon of a hand holding a sword, labeled "D" (Deployment) with a green status indicator.
- A circular icon for a "sample-app" application, labeled "A" (Application) with a green status indicator.

On the left sidebar, there are several navigation items: Topology, Monitoring, Search, Builds, Pipelines, Environments, Helm, Project, ConfigMaps, and Secrets. The "Topology" item is currently selected, highlighted in grey.



RED HAT OPEN SHIFT

☰ Red Hat
OpenShift Container Platform

Project: all projects ▾

Add ▾

Home

Catalog

Workloads

Networking

Storage

Builds

Monitoring

Compute

Administration ▾

- Cluster Status
- Cluster Settings
- Namespaces
- Service Accounts
- Roles
- Role Bindings
- Resource Quotas
- Limit Ranges
- Custom Resource Definitions

Cluster Status

Cluster Status

Health

Kubernetes API	OpenShift Console	Alerts Firing	Crashlooping Pods
UP All good	UP All good	0 Alerts	2 Pods

Control Plane Status

API Servers Up	Controller Managers Up	Schedulers Up	API Request Success Rate
100%	100%	100%	98%

Capacity Planning

CPU Usage	Memory Usage	Disk Usage	Pod Usage
17%	36%	16%	15%

Events

All Types ▾

All Categories ▾

Filter Events by name or message...

Software Info

Kubernetes	v1.13.4+f2cc675
------------	-----------------

Documentation

[Full Documentation](#)

From getting started with creating your first application, to trying out more advanced build and deployment techniques, these resources provide what you need to set up and manage your environment as a cluster administrator or an application developer.

Additional Support

- [Interactive Learning Portal](#)
- [Local Development](#)
- [YouTube](#)
- [Blog](#)



RED HAT OPEN SHIFT

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar is dark-themed and includes sections for Home, Projects, Search, Explore, Events, Operators, Workloads (selected), and various deployment-related options like Pods, Virtualization, Deployments, Deployment Configs, Stateful Sets, Secrets, Config Maps, Cron Jobs, Jobs, Daemon Sets, Replica Sets, Replication Controllers, and Horizontal Pod Autoscalers. The main content area is titled "Create Virtual Machine" under "Project: default". The wizard has six steps: 1. General (active), 2. Networking, 3. Storage, 4. Advanced, 5. Review, and 6. Result. In the "General" step, the "Name" field is set to "new-virtual-machine" and the "Description" field contains "My new virtual machine". The "Template" dropdown shows "No template available". The "Source" dropdown is set to "Disk". Under "Operating System", "Red Hat Enterprise Linux 8.0 or higher" is selected. The "Flavor" dropdown is set to "Small". The "Workload Profile" dropdown is set to "server". The top right of the interface shows a navigation bar with icons for dashboard, alerts, add, search, and help, and a user account for "admin".



www.ceste.es