

## Introducción a docker

Benito Cuesta & Salvador González

01/03/2016



MINISTERIO  
DE DEFENSA



Instituto Nacional de  
Técnica Aeroespacial

SECRETARIA DE ESTADO DE DEFENSA



# Índice



- Presentación
- Dockers. Definición y características.
- Getting Started
- Delivery con Kubernetes
- Microservicios. Estilo arquitectónico. Ventajas de dockers para una arquitectura de microservicios.
- Ciclo de vida con Devops / Integración Continua
- Caso práctico: INTA. Cluster de Libreoffice con Docker



# Open Canarias



---

Empresa de servicios y suministros TIC fundada en 1996

---

Actualmente tiene implantación nacional y con presencia internacional en México

---

Está especializada en productos y servicios IT, además de proyectos de alto riesgo tecnológico

---

Business partner de IBM

---

## Áreas de servicio

Servicios de Ingeniería del Software

Servicios de Infraestructura TIC

Servicios externos



# Servicios de Ingeniería del Software



---

Proporcionan soluciones a diferentes aspectos del ciclo de vida de aplicaciones, con especial énfasis en el incremento de la productividad y la calidad en los procesos y productos de trabajo vinculados a dicho ciclo de vida

---

Fundamentan estas soluciones en las capacidades en MDE (Model Driven Engineering), un nuevo paradigma de la Ingeniería del Software en el que Open Canarias ha sido pionero y contribuidor a varios estándares

---

Con MDE se construyen herramientas que automatizan procesos y tareas del ciclo de vida del software, reduciendo los tiempos de entrega, al mismo tiempo que se mejora la calidad al reducir las tareas manuales propensas a errores

## Estructura de los Servicios

Desarrollo de Herramientas

Modernización del Software

QA (Aseguramiento de la calidad)



# Antes de empezar...



“La **complejidad** cada vez mayor de las aplicaciones y la necesidad de **acelerar el desarrollo** están ejerciendo aún más presión sobre la infraestructura, procesos y equipos de TI.”



Fuente: <https://www.redhat.com/es/insights/containers>



# Otra cita...



“Los líderes en I&O deben mejorar las estrategias de implementación de servidores o crearán **infraestructuras incapaces** de dar soporte a la siguiente generación de aplicaciones.”

*Fuente: IT MARKET CLOCK FOR SERVER VIRTUALIZATION  
AND OPERATING ENVIRONMENTS, 16/09/2014.  
GARTNER N.º: G00262842*



# Un buen libro para empezar

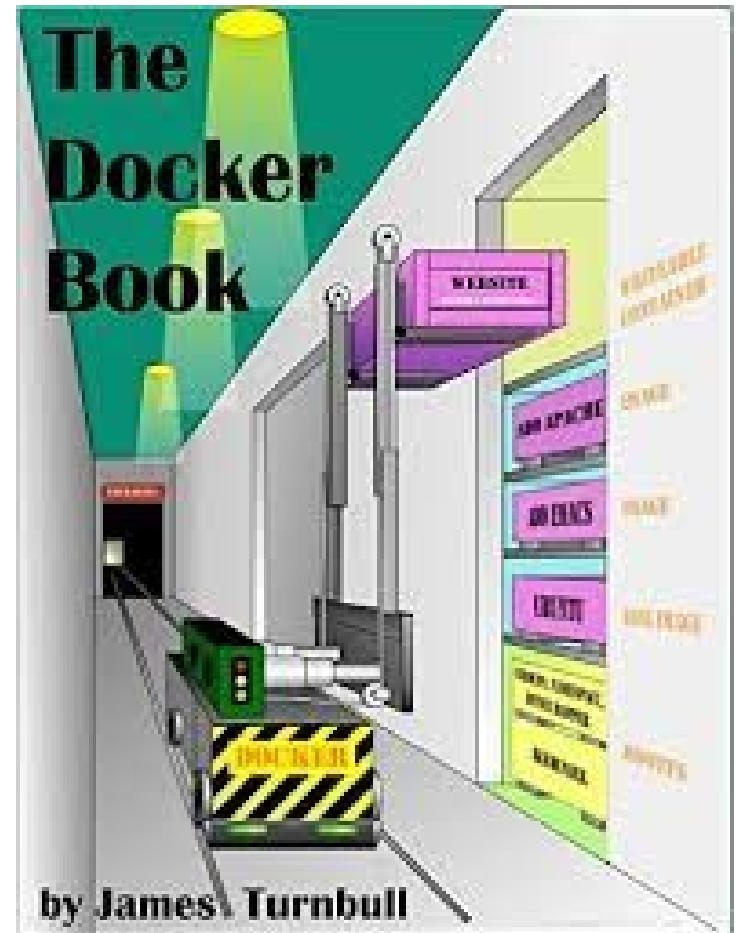


## The Docker Book

James Turnbull

August 4, 2014

Website: <http://www.dockerbook.com>





# Dockers

## Definición y características





# ¿Qué es Docker?



- “Es un proyecto open source para empaquetar, transportar y ejecutar cualquier aplicación como un contenedor ligero”.
- Su versión inicial se publica el 13 de Marzo de 2013 y está escrito en el lenguaje *GO*.
- Se trata de un nuevo modelo de virtualización que crea una capa de abstracción con el S.O.
- Un contenedor conforma el contexto de ejecución de una aplicación. Está compuesto de:
  - Sistema de ficheros
  - Procesos
  - Memoria
  - ... y todo aislado del S.O huésped



# Componentes



- Libcontainer (Derivado de LXC).
- Linux kernel namespaces (aislamiento del sistema de ficheros, red y procesos).
- Aislamiento de los recursos (Cpu, Memoria, etc) por medio de los cgroups.
- Copy-on-write
- Logging. Salidas de la línea de comando accesibles con el cliente docker.
- Consola interactiva.



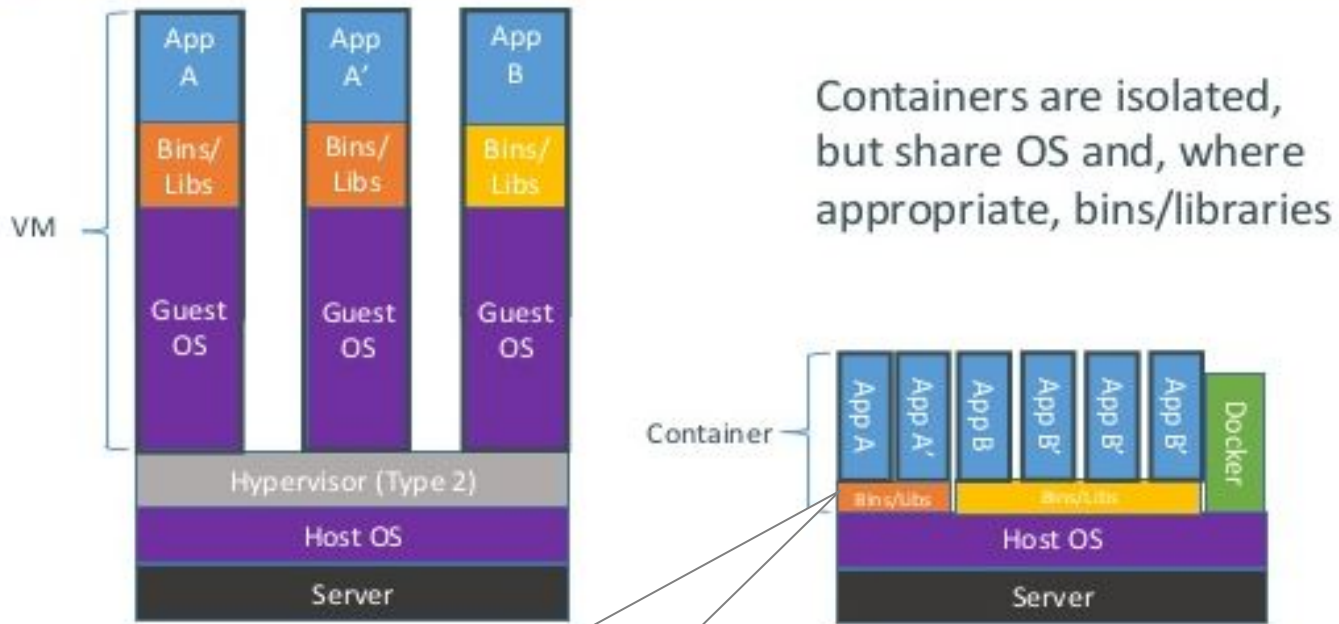
# La comunidad de Docker



- 1200 contribuidores al proyecto Docker
- 100,000 aplicaciones dockerizadas
- Entre 3 y 4 millones de desarrolladores usando Docker
- 300 millones de descargas
- 32,000 proyectos relacionados con Docker
- El 70% de las empresas TI están adoptando Docker (fuente: SD Times)

# Nuevo modelo

## Containers vs. VMs



Partes comunes



# Nueva visión del S.O.



- Con el nuevo modelo de contenedores podemos abstraernos del S.O. donde corre el motor Docker.
- Diseñados para un mundo Cloud, distribuido.
- Los nuevos Sistemas Operativos (BareMetal/Virtual)



# Registro



- Registro
  - Registro público (Docker Hub)
    - > 10.000 imágenes disponibles
  - Registro privado
    - Sólo accesible bajo autorización. Es de pago.
- Registro local
  - Este registro se instala por medio de un contenedor Docker para centralizar las imágenes de los contenedores



## DESARROLLADORES DE APLICACIONES

1. Lanzamientos de calidad superior
2. Mejor escalabilidad de aplicaciones
3. Mayor aislamiento de aplicaciones



## ARQUITECTOS DE TI

1. Escalabilidad horizontal más rápida
2. Ciclos de pruebas más cortos
3. Menos errores de implementación



## OPERACIONES DE TI

1. Lanzamientos de calidad superior
2. Sustitución eficiente de todas las máquinas virtuales en producción
3. Gestión de aplicaciones más fácil

Fuente: <https://www.redhat.com/es/insights/containers>



# Getting Started





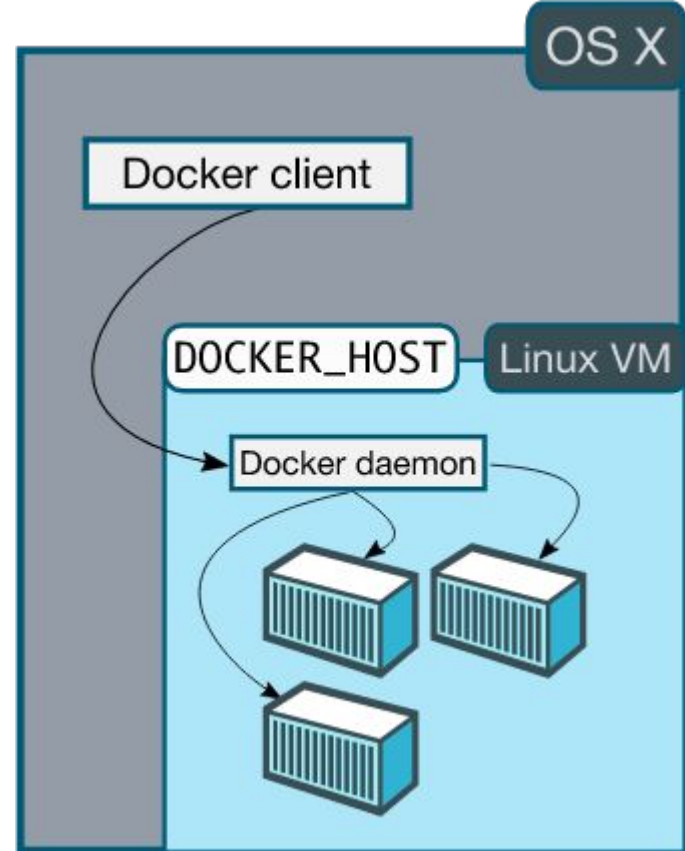
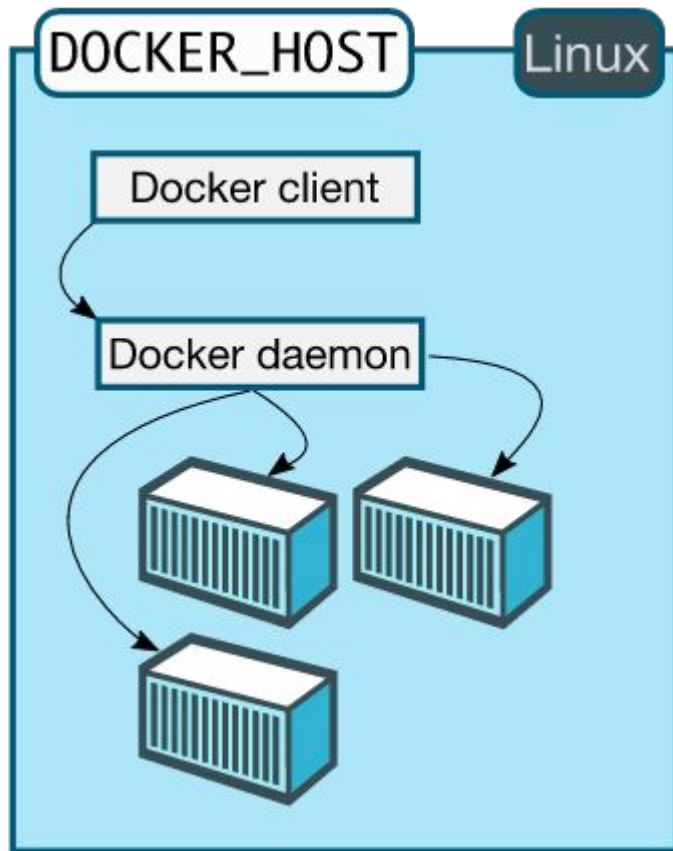
# Instalación

- En linux se hace uso de los comandos de gestión de paquetes habituales

```
/bin/bash
/bin/bash 80x24
sgonzalez@Think3 ~ $ sudo yum install docker
```

```
/bin/bash
/bin/bash 80x24
sgonzalez@Think3 ~ $ sudo apt-get install docker.io
```

- Para Windows y Mac se hace uso de un programa llamado “docker-machine”
- Docker es un demonio que expone una interfaz REST





# Comandos iniciales



- docker (lista los posibles comandos)
- docker version (versión del demonio y del cliente)
- docker info (información del sistema)

```
sgonzalez@Think3 ~ $ docker version
Client version: 1.0.1
Client API version: 1.12
Go version (client): go1.2.1
Git commit (client): 990021a
Server version: 1.0.1
Server API version: 1.12
Go version (server): go1.2.1
Git commit (server): 990021a
```

```
sgonzalez@Think3 ~ $ docker info
Containers: 0
Images: 10
Storage Driver: aufs
  Root Dir: /var/lib/docker/aufs
  Dirs: 10
Execution Driver: native-0.2
Kernel Version: 3.13.0-37-generic
WARNING: No swap limit support
```

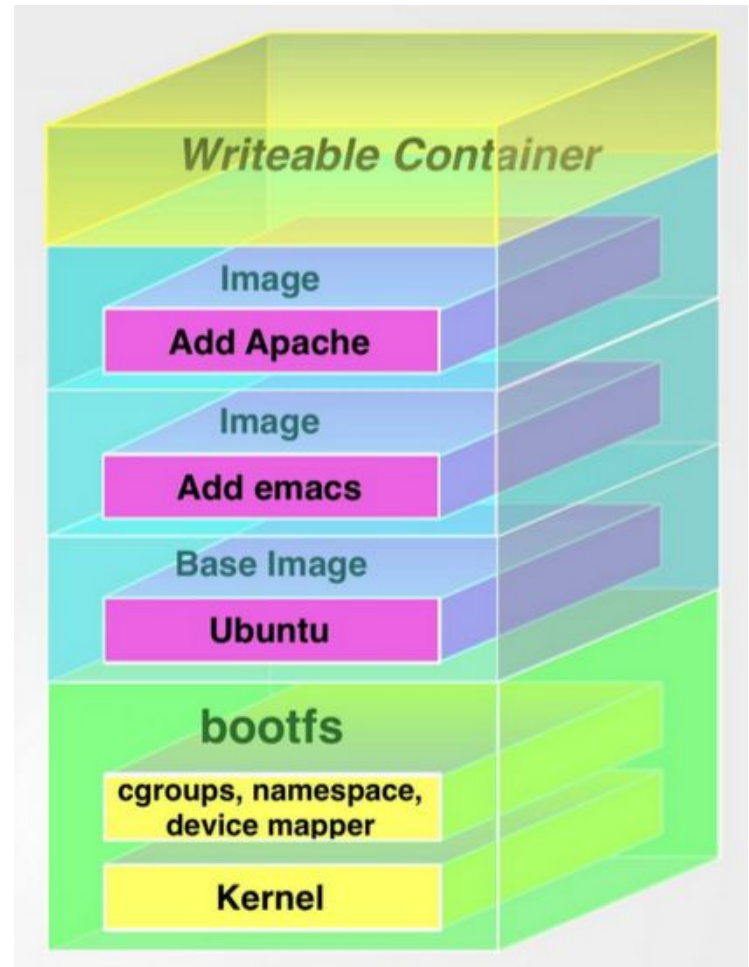


# Nuestro primer contenedor



- *docker run -i ubuntu echo Hello World*
  - run → Ejecutar un contenedor
  - -i → Modo interactivo
  - ubuntu → Imagen a usar en el contenedor
  - echo Hello World → Comando a ejecutar
- Flujo:
  - Busca la imagen localmente, luego en repositorios centrales.

- Una imagen está formada por capas (*layers*) que se montan unas encima de otras. Todas en modo sólo lectura.
- La última capa se monta como lectura/escritura y da lugar al contenedor.
- Las capas usan el patrón “copy on write”





# Comandos imágenes



- *docker images*
- *docker run -t -i --name test01 ubuntu /bin/bash*  
(ubuntu:latest)
- *docker run -t -i --name test02 centos:centos6*  
*/bin/bash*
- *docker pull ubuntu:14.04*
- *docker pull sgonzalez/centosopen:centos5*
- *docker search centos*



# Crear imágenes



- docker **commit** (no recomendado)
  - *docker run -it centos /bin/bash*
  - *yum upgrade && yum install httpd*
  - *vi /var/www/html/index.html*
  - *exit*
  - *docker commit <containerID> sgonzalez/apache:webopen*
- docker **build**
  - Consiste en un fichero (Dockerfile) de instrucciones para construir las imágenes

```
# version: 0.0.1
FROM centos:latest
MAINTAINER Salvador Gonzalez "sgonzalez@opencanarias.es"
RUN yum clean all && yum update -y
RUN yum install httpd -y
```



# Dockerfile



- Comandos más habituales
  - **FROM** → De que imagen partimos para crear la nueva
  - **MAINTAINER** → Quien mantiene el contenedor
  - **RUN** → Ejecuta una instrucción en el contenedor
  - **ADD** → Añade un fichero o carpeta al contenedor
  - **ENV** → Establece una variable de entorno en el contenedor
  - **EXPOSE** → Indica que se va a exponer un puerto del contenedor
  - **ENTRYPOINT / CMD** → Qué se ejecuta





# ENTRYPOINT / CMD



- ENTRYPOINT define el proceso a ejecutar
  - Por defecto es “/bin/sh -c”
- CMD son los parámetros del proceso
- Por ejemplo:
  - *CMD [“/usr/bin/top”]*
  - *docker run -i -t myubuntu (/bin/sh -c /usr/bin/top)*
  - *docker run -i -t myubuntu **bash** (/bin/sh -c bash)*
  
  - *ENTRYPOINT [“/bin/cat”]*
  - *docker run -it catimg /etc/passwd (/bin/cat /etc/passwd)*



# Contenedores

- Hacen uso de una imagen como base y pueden contener uno o más procesos.
- Un contenedor Docker es:
  - Una instancia de una imagen Docker
  - Un conjunto de operaciones asociadas a su ciclo de vida:
    - Crear, Destruir, Arrancar, Reiniciar o Parar
  - Un entorno de ejecución



# Comandos contenedores



- `docker run -d ubuntu /bin/sh -c "while true; do echo Hello World; sleep 1; done"`*

<code>docker ps [-a]</code>	<code>docker inspect &lt;ID&gt;</code>
<code>docker attach &lt;ID&gt;</code>	<code>docker stop &lt;ID&gt;</code>
<code>docker start &lt;ID&gt;</code>	<code>docker rm [-f] &lt;ID&gt;</code>
<code>docker logs [-f] &lt;ID&gt;</code>	<code>docker create &lt;image&gt;</code>
<code>docker top &lt;ID&gt;</code>	<code>docker build [-f dockerfile] &lt;dir&gt;</code>
<code>docker exec &lt;ID&gt; &lt;cmd&gt;</code>	<code>docker load / save</code>



# Uso de contenedores



- Data Volumes

- Dockerfile

- VOLUME [ “/var/log/http” ]

- Docker run

- *docker run -v [rutahost:]rutacontenedor[:rw,:ro]*

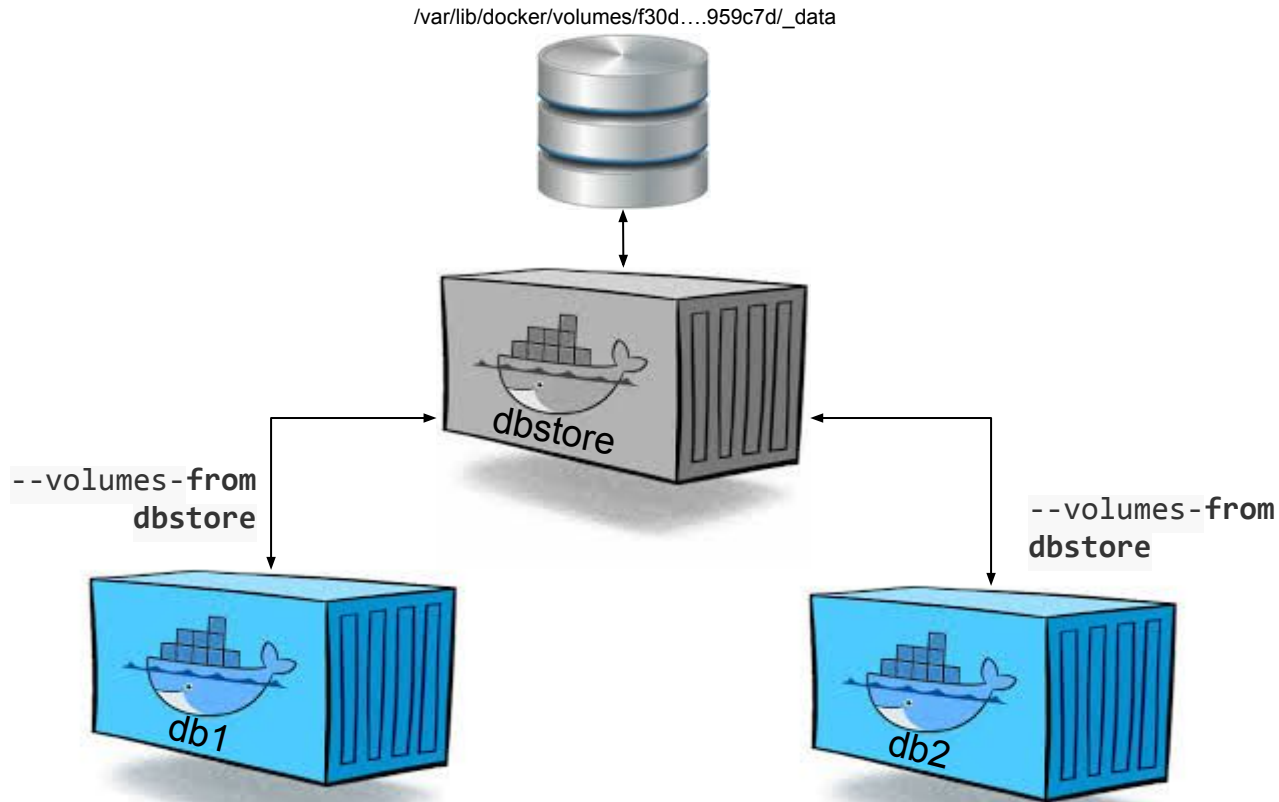
- Data Volume Containers

- *docker run --volumes-from datacontainer ...*

- Links

- (1) *docker run -d --name database01 ...*

- (2) *docker run -d --name web --link database01:db ...*



```
docker create -v /dbdata --name dbstore ubuntu /bin/true
```

```
docker run -d --volumes-from dbstore --name db1 training/postgres
```

```
docker run -d --volumes-from dbstore --name db2 training/postgres
```

Con EXPOSE:

- Se expone un puerto del contenedor al host.

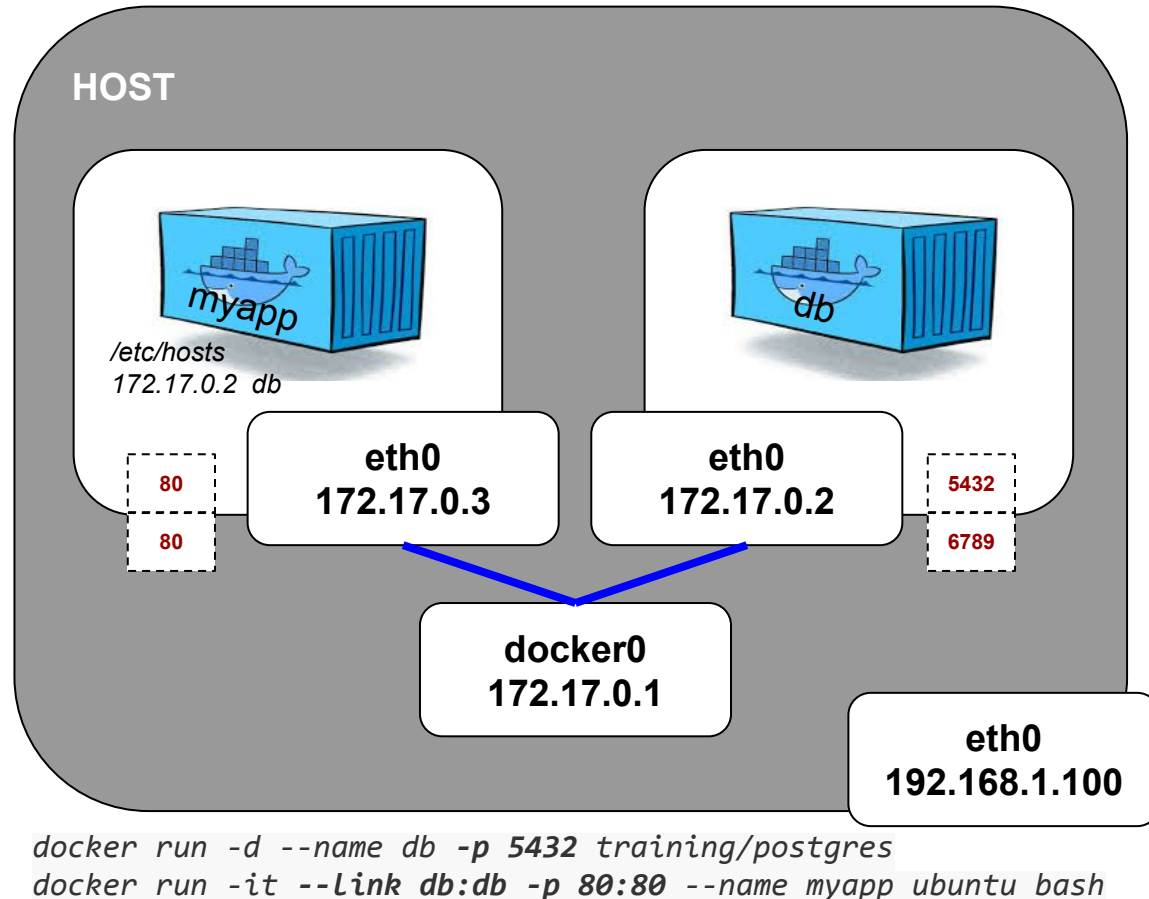
```
-p [<host_port>:]<cont_port>
```

Quando se “linkan” dos contenedores pasan dos cosas:

- Se crean variables de entorno

```
<name>_PORT_<port>_<protocol>
```

- Se modifica el `/etc/hosts`





# Delivery con Kubernetes



# Orquestación

- Diferentes plataformas de orquestación

Kubernetes (Proyecto de Google, 2014)

Mesos (Proyecto apache, 2015)

Docker Swarm (Proyecto de Docker, 2015)

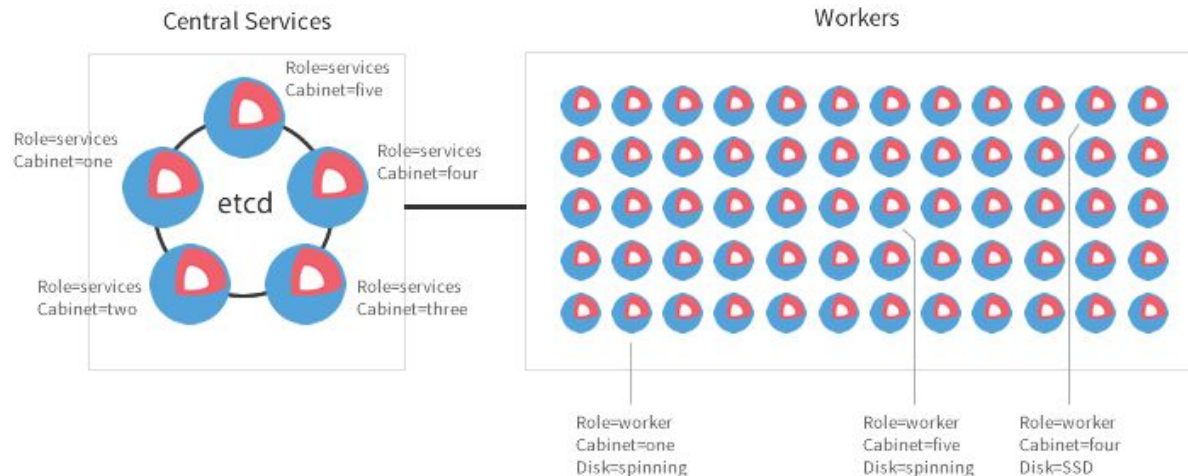
- Kubernetes (2014)

Plataforma para automatizar el despliegue, escalado y uso de contenedores sobre un cluster de servidores.

El sistema es altamente portable (*public, private, hybrid, multi-cloud*)

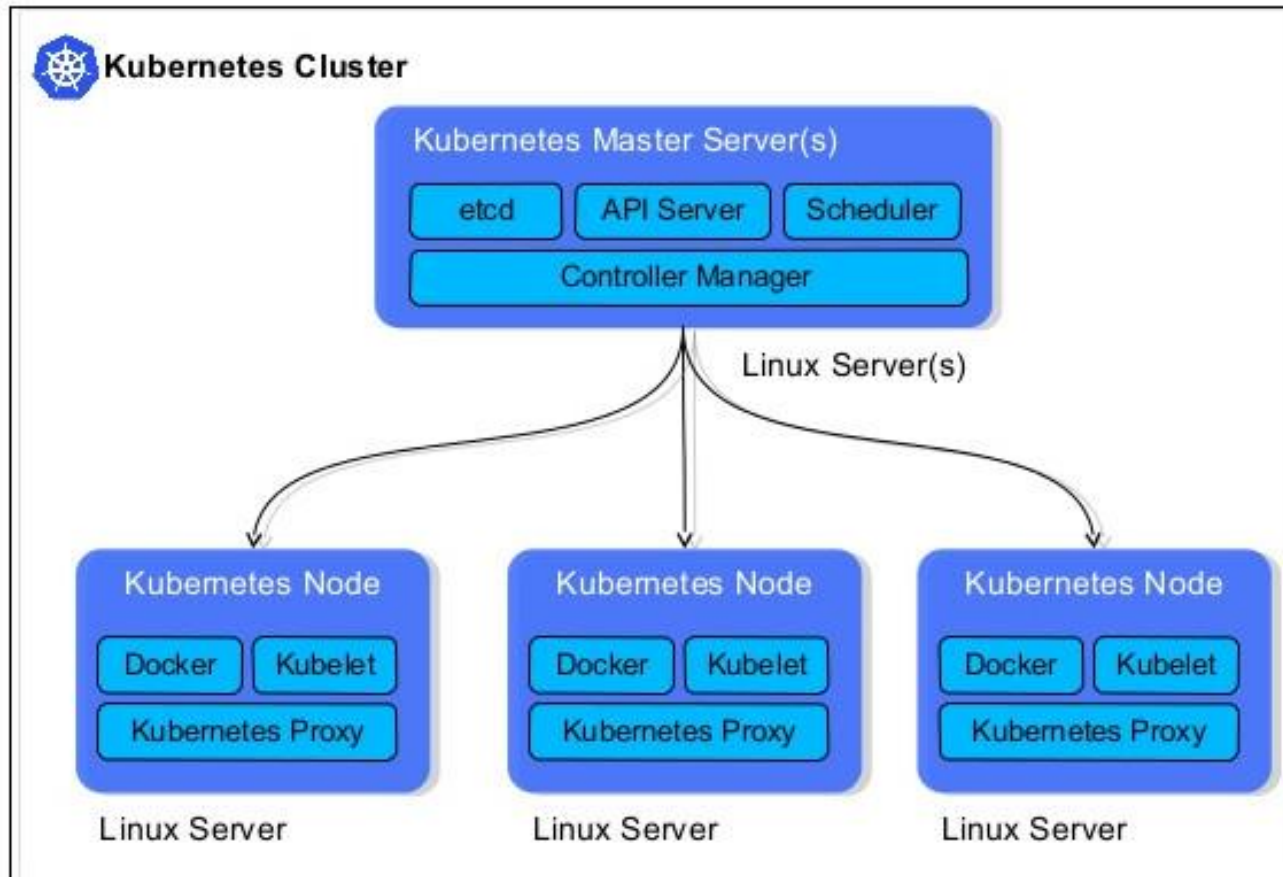


- El Cluster está compuesto por...
  - Número impar de servidores de gestión ( $\geq 3$ )
  - Ilimitado número de nodos de trabajo (minions)
- Cita
  - “Hay que tratar los servidores como ganado y no como mascotas”



# Arquitectura

## Kubernetes Architectural Overview





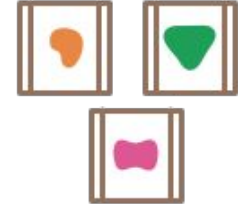
# Microservicios Estilo Arquitectónico

# Microservices Architecture (MSA)

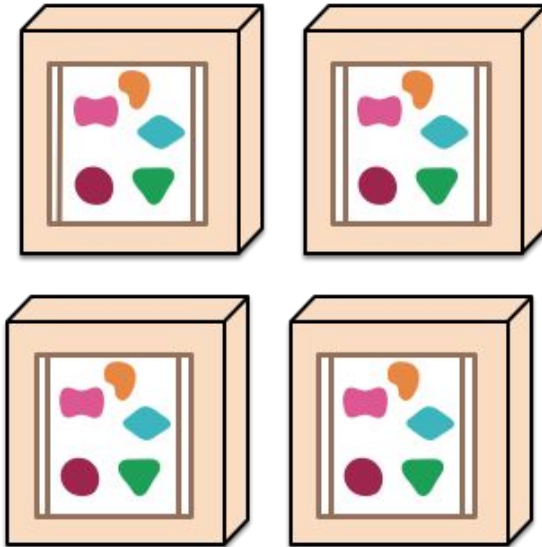
*A monolithic application puts all its functionality into a single process...*



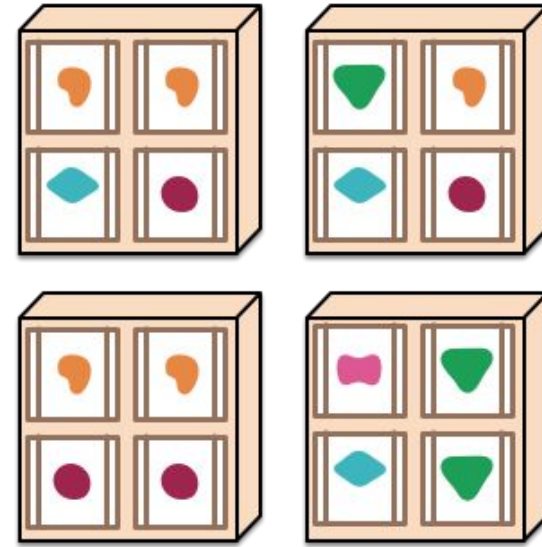
*A microservices architecture puts each element of functionality into a separate service...*



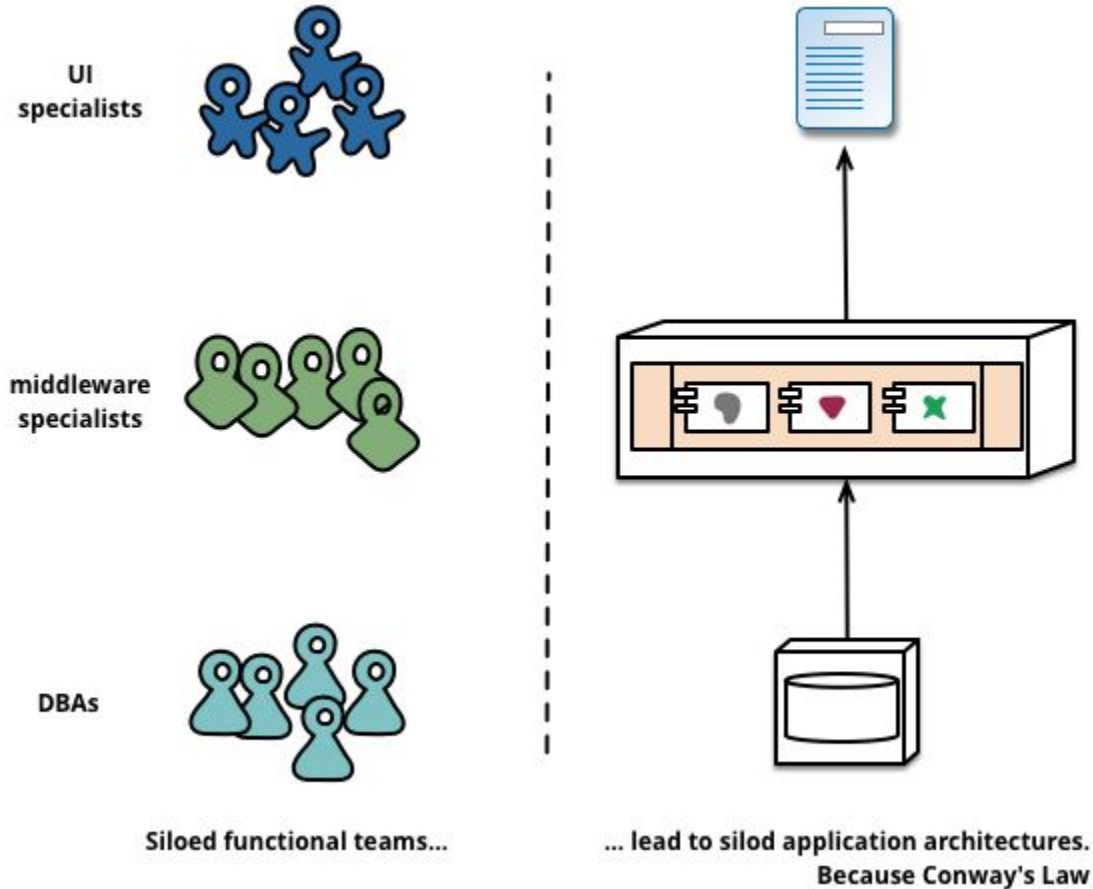
*... and scales by replicating the monolith on multiple servers*

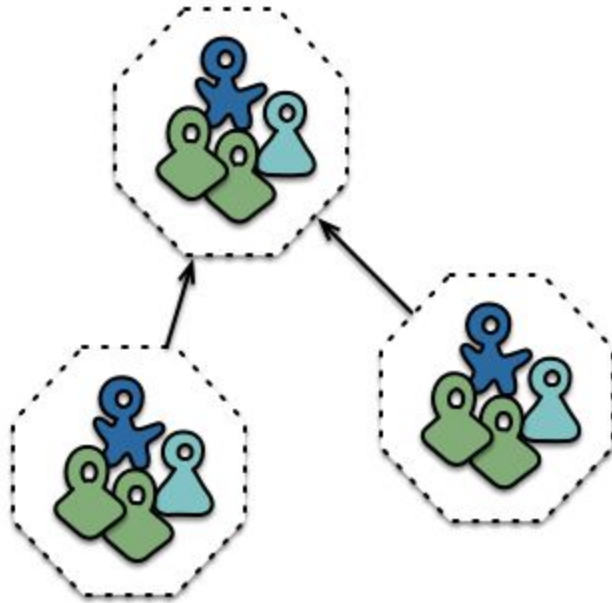


*... and scales by distributing these services across servers, replicating as needed.*

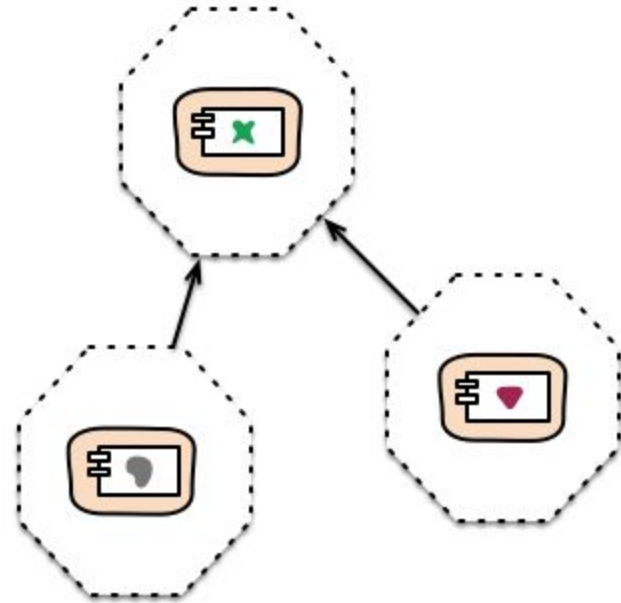


*Fuente: <http://martinfowler.com/articles/microservices.html>*



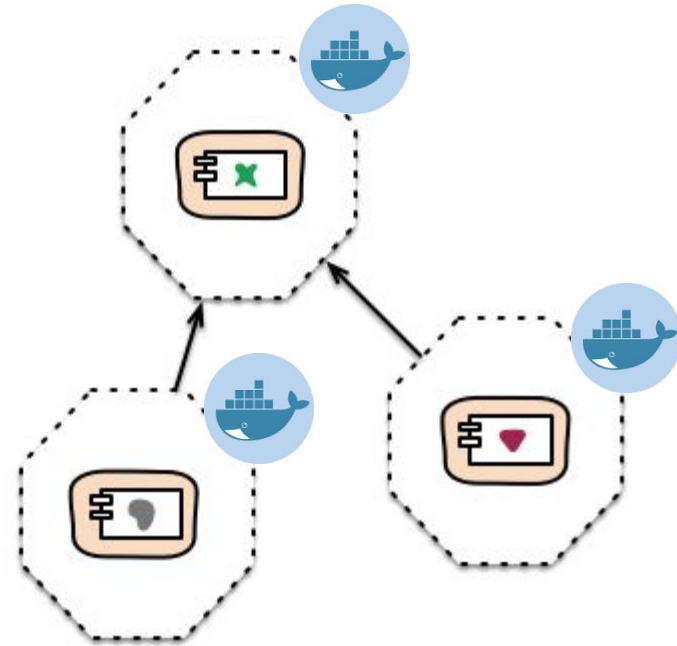


Cross-functional teams...



... organised around capabilities  
Because Conway's Law

- Cloud Ready
- Simplificación de los despliegues
- Capacidad de escalado dinámico y automático
- Portabilidad de las soluciones
- Reutilización
- Facilita la Integración Continua



... organised around capabilities  
Because Conway's Law

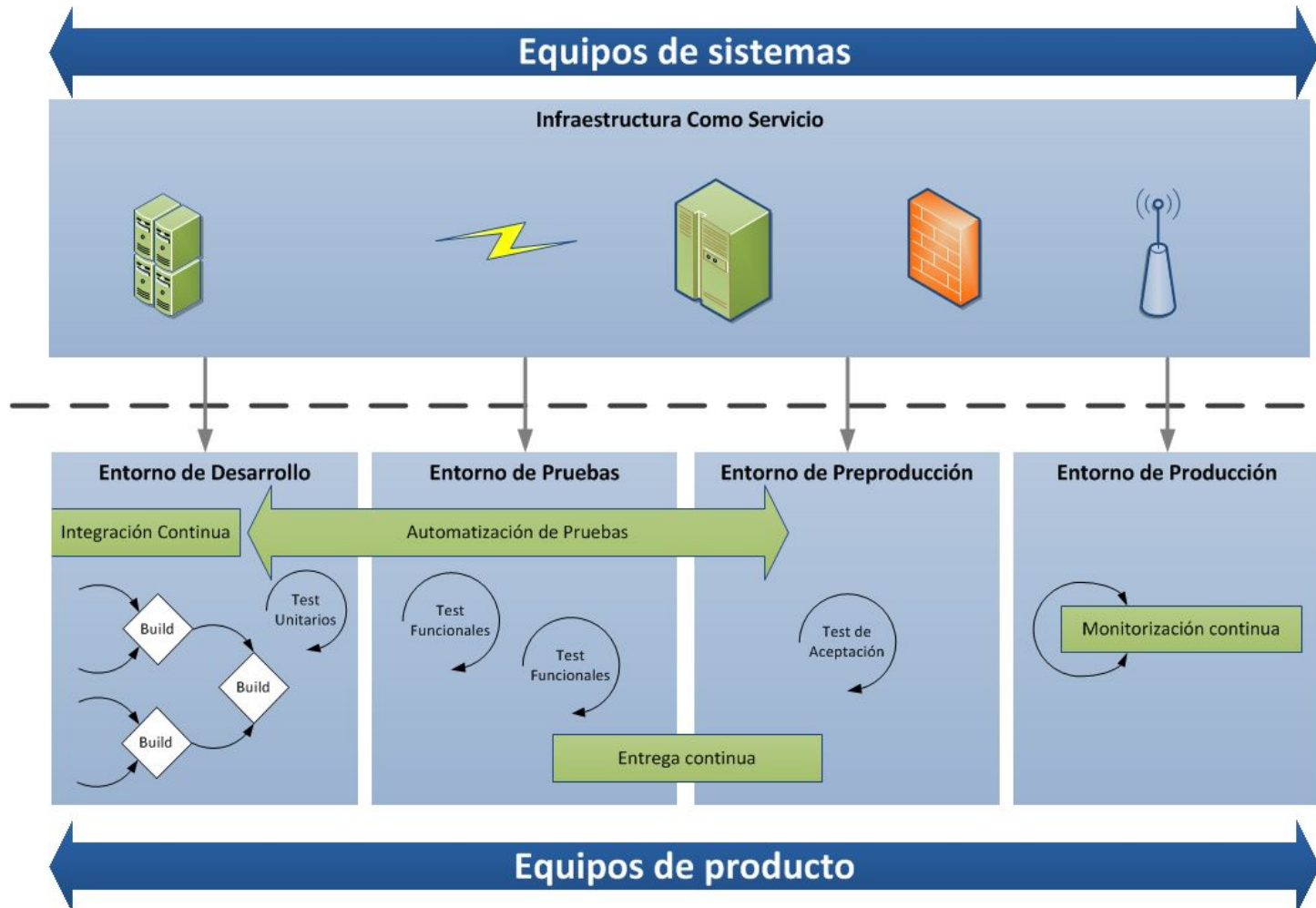


# Ciclo de vida con Devops

## Integración Continua con Dockers



# DevOps





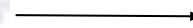
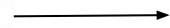
# Integración Continua



## Jenkins

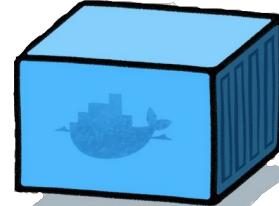


git





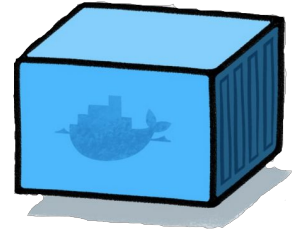
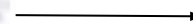
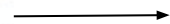
# Integración Continua



## Jenkins



git





# Dockers for CI (I)



```
FROM ubuntu:14.04
MAINTAINER bcuesta@opencanarias.es
ENV REFRESHED_AT 2016-03-01

RUN apt-get update -qq && apt-get install -qqy curl↵
apt-transport-https
RUN curl https://get.docker.io/gpg | apt-key add -
RUN echo deb http://get.docker.io/ubuntu docker main > /etc/apt/↵
sources.list.d/docker.list
RUN apt-get update -qq && apt-get install -qqy iptables ca-↵
certificates lxc openjdk-7-jdk git-core lxc-docker

ENV JENKINS_HOME /opt/jenkins/data
ENV JENKINS_MIRROR http://mirrors.jenkins-ci.org

RUN mkdir -p $JENKINS_HOME/plugins
RUN curl -sf -o /opt/jenkins/jenkins.war -L $JENKINS_MIRROR/war-↵
stable/latest/jenkins.war

RUN for plugin in chucknorris greenballs scm-api git-client git ↵
ws-cleanup ;\
do curl -sf -o $JENKINS_HOME/plugins/${plugin}.hpi \
-L $JENKINS_MIRROR/plugins/${plugin}/latest/${plugin}.hpi ↵
; done

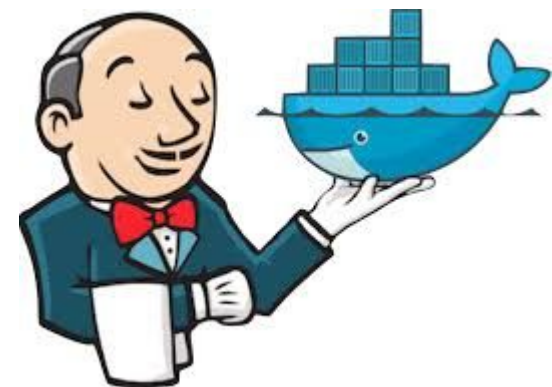
ADD ./dockerjenkins.sh /usr/local/bin/dockerjenkins.sh
RUN chmod +x /usr/local/bin/dockerjenkins.sh

VOLUME /var/lib/docker

EXPOSE 8080

ENTRYPOINT [ "/usr/local/bin/dockerjenkins.sh" ]
```

- `$ docker build -t↵ opencanarias/jenkins .`
- `$ docker run -p 8080:8080↵ --name jenkins↵ --privileged↵ -d opencanarias/jenkins`



Fuente: *The Docker Book*. James Turnbull



# Dockers for CI (II)



Proyecto nombre

Descripción

[Plain text] [Visualizar](#)

- Desechar ejecuciones antiguas
- Esta ejecución debe parametrizarse
- Desactivar la ejecución (No se ejecutará nuevamente hasta que el proyecto sea reactivado.)
- Lanzar ejecuciones concurrentes en caso de ser necesario

**Opciones avanzadas del proyecto**

- Periodo de espera
- Contador de reintentos
- Congelar el lanzamiento cuando haya un proyecto padre ejecutándose
- Bloquear la ejecución cuando un proyecto relacionado está en ejecución
- Utilizar un directorio de trabajo personalizado

Directorio

Nombre a mostrar

¿Conservar los 'logs' de dependencias de las ejecuciones?

**Configurar el origen del código fuente**

- Ninguno
- CVS
- CVS Projectset
- Git

Repositories

Repository URL

Credentials



# Dockers for CI (II)



Proyecto nombre  
Descripción

- Desechar ejecución
- Esta ejecución de
- Desactivar la ejecución
- Lanzar ejecución

Opciones avanzadas

- Período de espera
- Contador de reinicios
- Congelar el lanzamiento
- Bloquear la ejecución
- Utilizar un directorio

Nombre a mostrar

- ¿Conservar los resultados?

Configurar el origen

- Ninguno
- CVS
- CVS Projectset
- Git

Repositories

Consultar repositorio (SCM)

Ejecutar periódicamente

**Entorno de ejecución**

Delete workspace before build starts

**Ejecutar**

**Ejecutar línea de comandos (shell)**

Comando

```
# Build the image to be used for this job.
IMAGE=$(docker build . | tail -1 | awk '{ print $NF }')
# Build the directory to be mounted into Docker.
MNT="$WORKSPACE/.."
# Execute the build inside Docker.
CONTAINER=$(docker run -d -v "$MNT:/opt/project" $IMAGE /bin/bash -c 'cd
# Attach to the container so that we can see the output.
docker attach $CONTAINER
# Get its exit code as soon as the container stops.
RC=$(docker wait $CONTAINER)
# Delete the container we've just used.
docker rm $CONTAINER
# Exit with the same value as that with which the process exited.
exit $RC
```

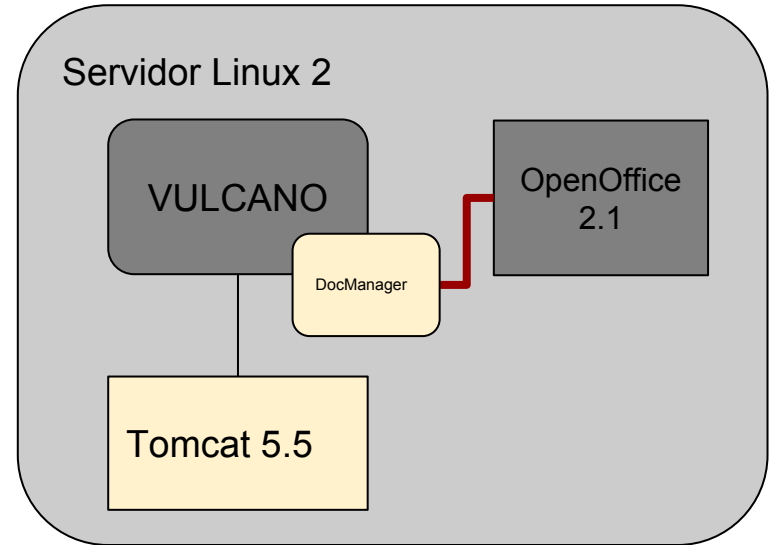
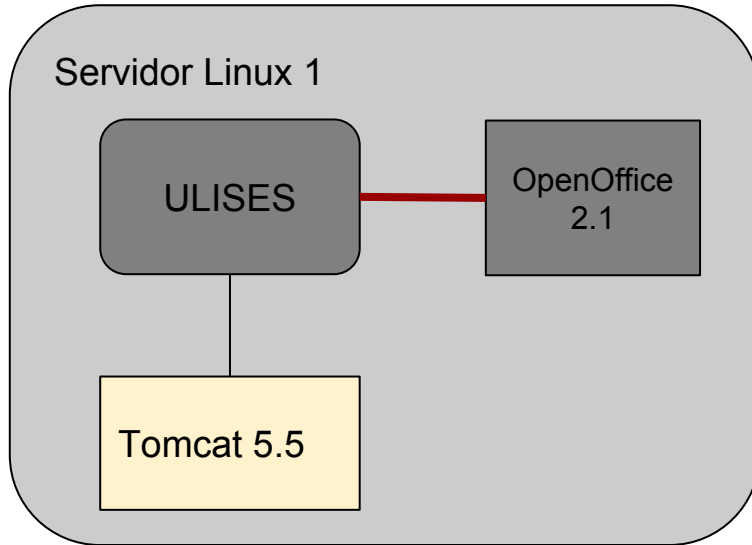
[Visualizar la lista de variables de entorno disponibles](#)



# Ejemplos Prácticos



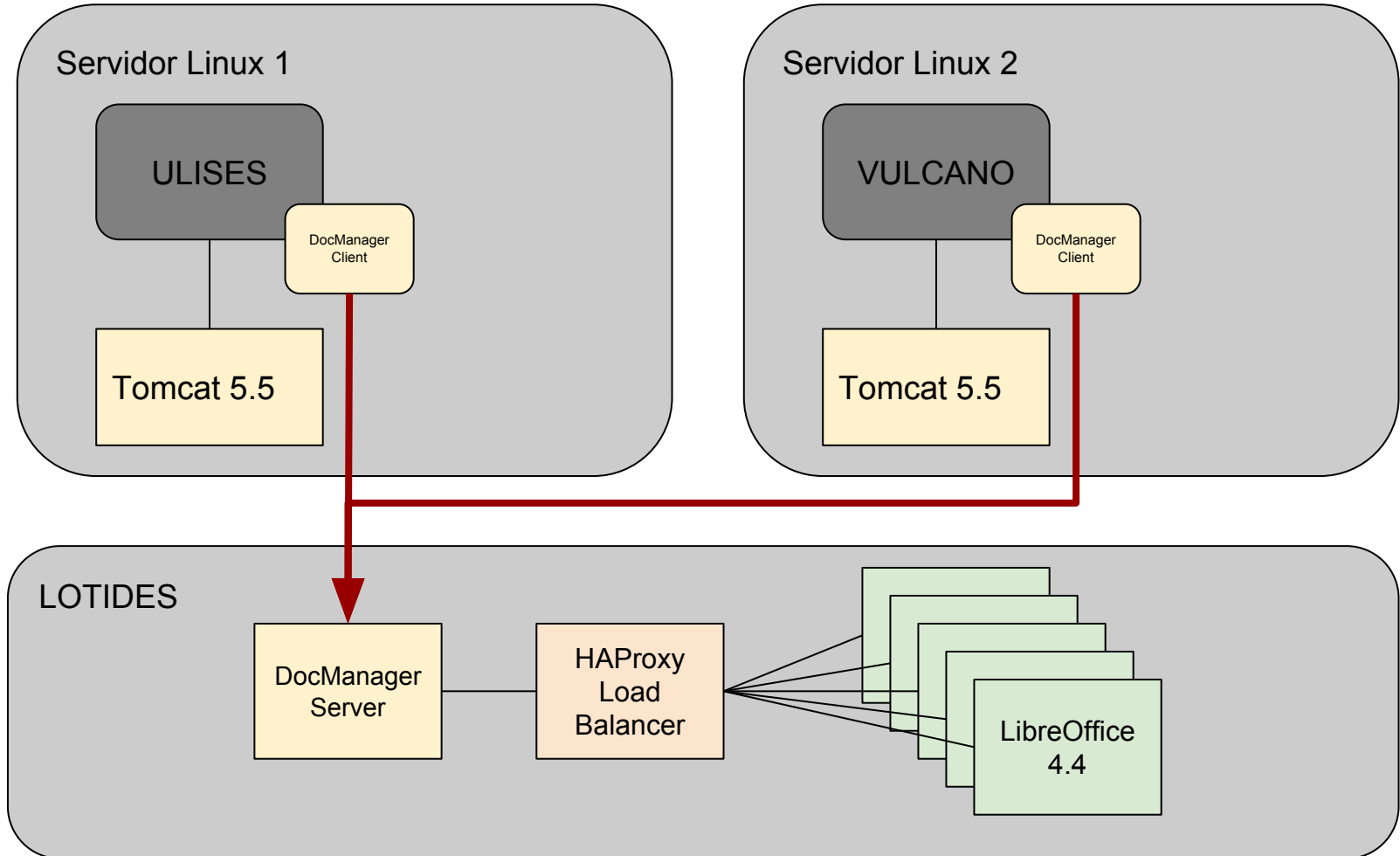
# DOCMANAGER (INTA)



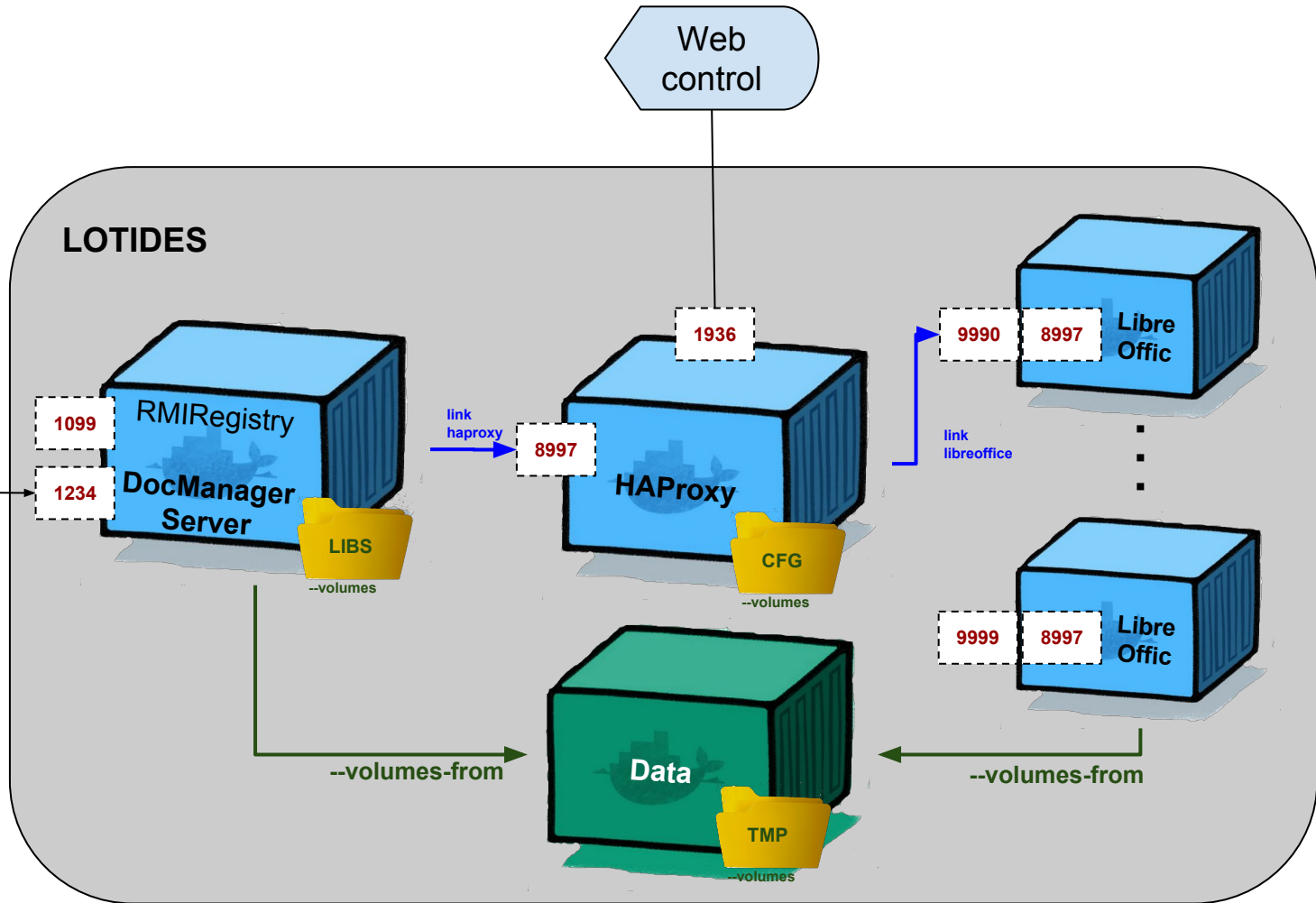




# DOCMANAGER (INTA)



# DOCMANAGER (INTA)





# Compose



## DocManagerData:

image: ubuntu

### volumes:

- "/tmp/DOCManager"
- "/tmp/DOCManager/temp"

## DocManagerServer:

image: inta/docmanagerserver

### links:

- "haproxy:haproxy"

### volumes:

- "/usr/local/src/DockerImages/libs:/opt/DocManagerServer/lib"

### volumes\_from:

- DocManagerData

### ports:

- "1099:1099"
- "1234:1234" ↵

## haproxy:

image: inta/haproxy

### links:

- "libreoffice:libreoffice"

### volumes:

- /etc/haproxy/haproxy.cfg:/etc/haproxy/haproxy.cfg

### ports:

- "8997:8997"
- "1936:1936"

## libreoffice:

image: inta/libreoffice

environment:

- TCP\_PORTS=8997

### expose:

- "8997"

### volumes\_from:

- DocManagerData



# Compose

## DocManagerData:

image: ubuntu

### volumes:

- "/tmp/DOCManager"
- "/tmp/DOCManager/temp"

## DocManagerServer:

image: inta/docmanagerserver

### links:

- "haproxy:haproxy"

### volumes:

- "/usr/local/src/DockerImages/libs:/opt/DOCManagerServer/lib"

### volumes\_from:

- DocManagerData

### ports:

- "1099:1099"
- "1234:1234" ↵

## haproxy:

image: inta/haproxy

### links:

- "libreoffice:libreoffice"

### volumes:

- /etc/haproxy/haproxy.cfg:/etc/haproxy/haproxy.cfg

### ports:

- "8997:8997"
- "1936:1936"

## libreoffice:

image: inta/libreoffice

### environment:

- TCP\_PORTS=8997

### expose:

- "8997"

### volumes\_from:

- DocManagerData

*docker-compose up -d*



# Compose

## DocManagerData:

```
image: ubuntu
volumes:
  - "/tmp/DOCManager"
  - "/tmp/DOCManager/temp"
```

## DocManagerServer:

```
image: inta/docmanagerserver
links:
  - "haproxy:haproxy"
volumes:
  - "/usr/local/src/DockerImages/libs:/opt/DOCManagerServer/lib"
volumes_from:
  - DocManagerData
ports:
  - "1099:1099"
  - "1234:1234" ↵
```

## haproxy:

```
image: inta/haproxy
links:
  - "libreoffice:libreoffice"
volumes:
  - /etc/haproxy/haproxy.cfg:/etc/haproxy/haproxy.cfg
ports:
  - "8997:8997"
  - "1936:1936"
```

## libreoffice:

```
image: inta/libreoffice
environment:
  - TCP_PORTS=8997
expose:
  - "8997"
volumes_from:
  - DocManagerData
```

*docker-compose scale libreoffice=5*



# Usos de Docker




- Ayudar a que nuestro entorno de desarrollo sea más rápido, eficiente y más ligero. Los desarrolladores pueden compartir los contenedores entre ellos.
- Ejecución de micro-servicios y aplicaciones consistentemente entre diferentes entornos. (AWS, GCE, etc).
- Creación de entornos de test para la integración continua como Jenkins por ejemplo.
- Creación y prueba de entornos complejos en máquinas locales, antes de pasarlos a producción.
- Construcción de “sandboxes” ligeras para desarrollo, test, etc.
- SaaS y PaaS



# Gracias



 [info@opencanarias.com](mailto:info@opencanarias.com)

 [www.opencanarias.com](http://www.opencanarias.com)

 [www.linkedin.com/company/open-canarias-sl](http://www.linkedin.com/company/open-canarias-sl)

 [www.facebook.com/opencanarias](http://www.facebook.com/opencanarias)

 [www.google.com/+opencanarias](http://www.google.com/+opencanarias)

 [@OpenCanarias](https://twitter.com/OpenCanarias)



Open Canarias, S.L.  
info@opencanarias.es

C/ Elías Ramos González, 4 Of 304  
38001 Santa Cruz de Tenerife

C/ Alejandro Hidalgo, 3  
35005 Las Palmas de Gran Canaria

[www.opencanarias.com](http://www.opencanarias.com)