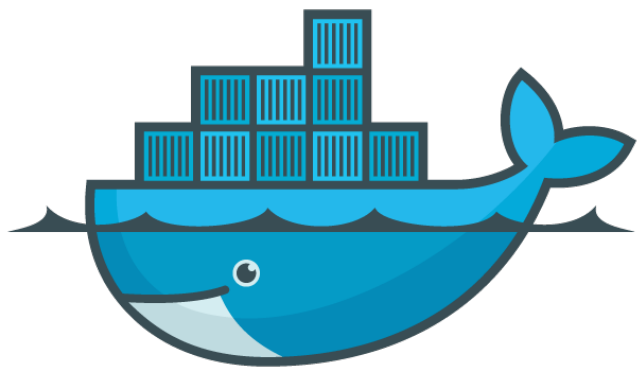
VIRTUALIZACIÓN BASADA EN CONTAINERS CON DOCKER.

Seminario CCTVal, UTFSM

27 de diciembre de 2015

Maximiliano Osorio
<mosorio@inf.utfsm.cl>

Universidad Técnica Federico Santa María



AGENDA

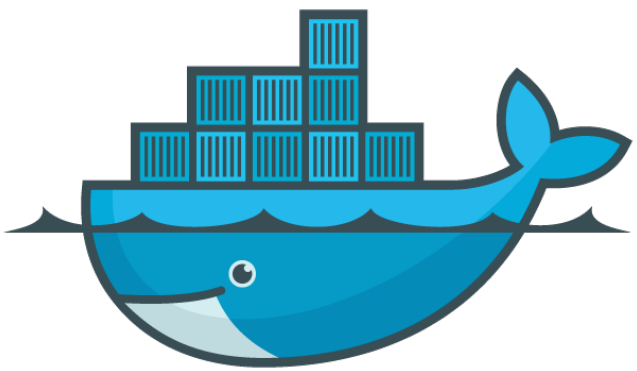
1. Motivación

2. Docker

2.1 Beneficios

2.2 Rendimiento

3. Demos



MOTIVACIÓN

ERASE UNA VEZ...

Imagen: docker.com

ERASE UNA VEZ...

Imagen: docker.com

HOY

django web frontend	?	?	?	?	?	?
node.js async API	?	?	?	?	?	?
background workers	?	?	?	?	?	?
SQL database	?	?	?	?	?	?
distributed DB, big data	?	?	?	?	?	?
message queue	?	?	?	?	?	?
	my laptop	your laptop	QA	staging	prod on cloud VM	prod on bare metal

Imagen: docker.com

HOY

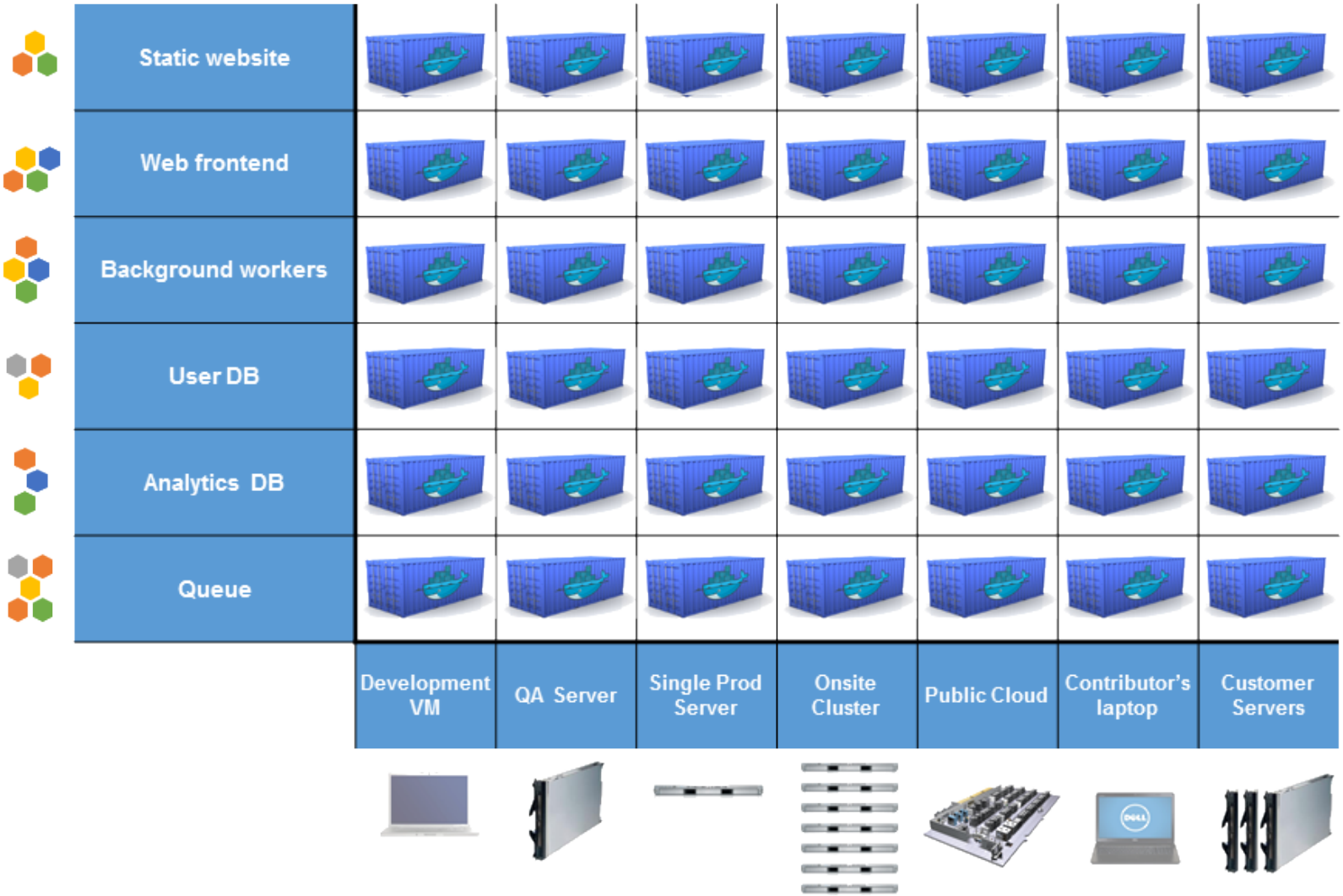


Imagen: docker.com

DOCKER CONTAINERS

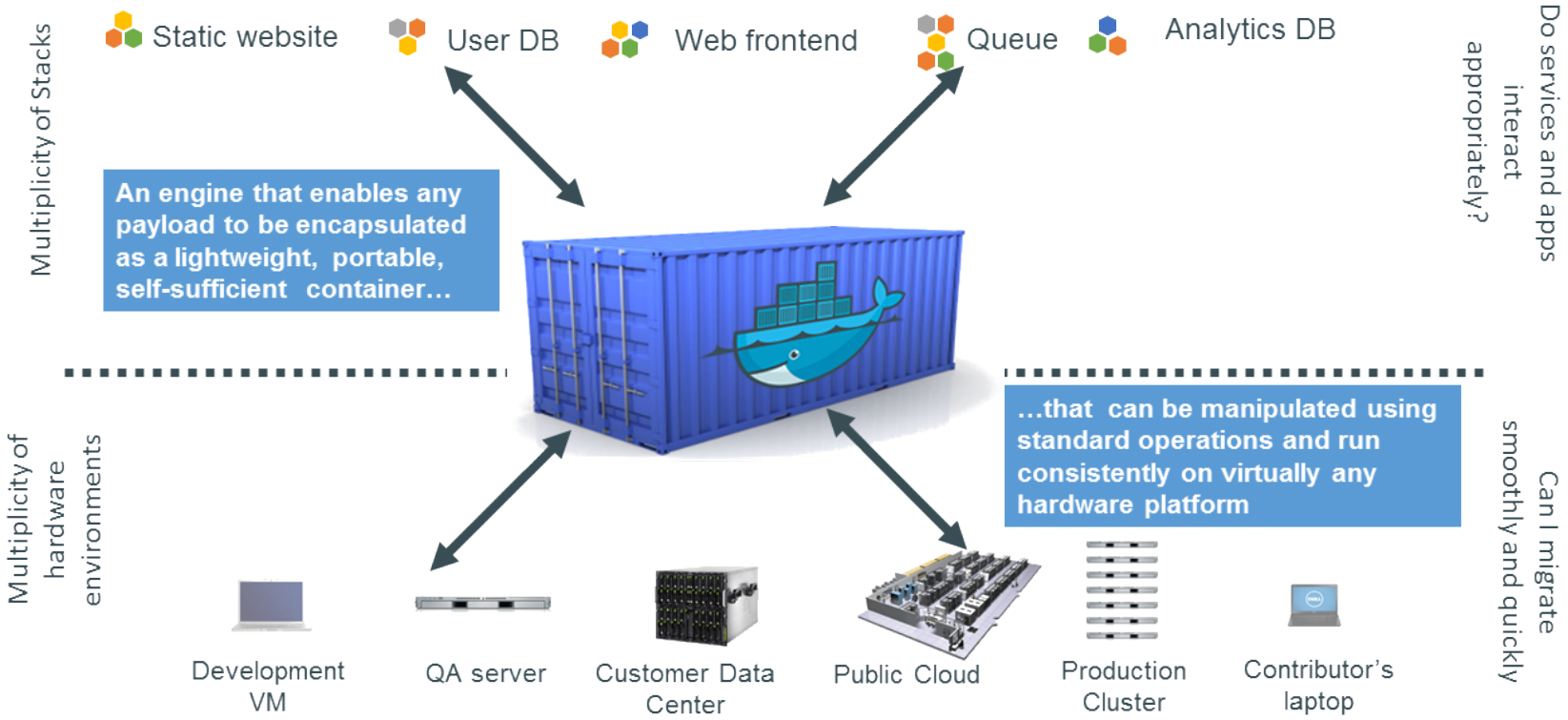


Imagen: docker.com

DOCKER

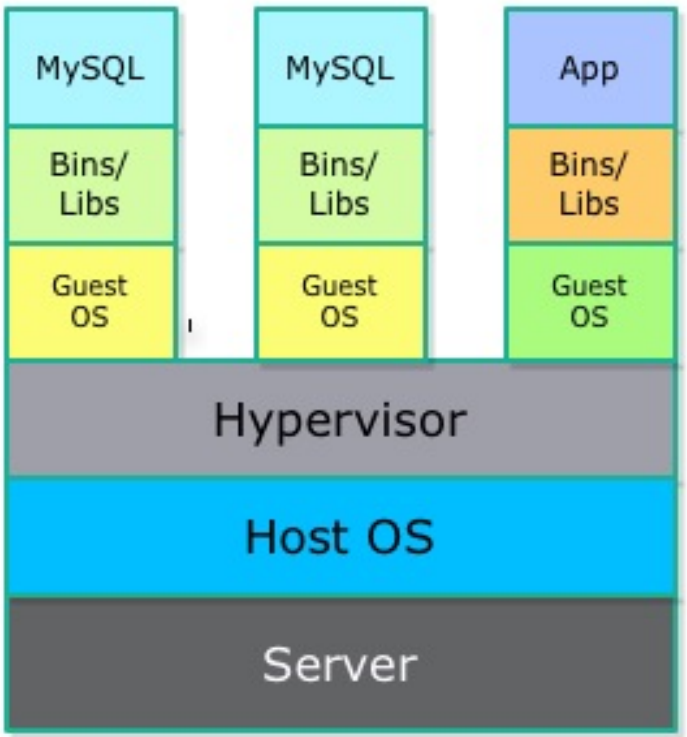
DOCKER

- Docker es un proyecto open-source
- Utiliza la tecnología de los containers (libcontainer).
- Para “construir, migrar y correr aplicaciones distribuidas".

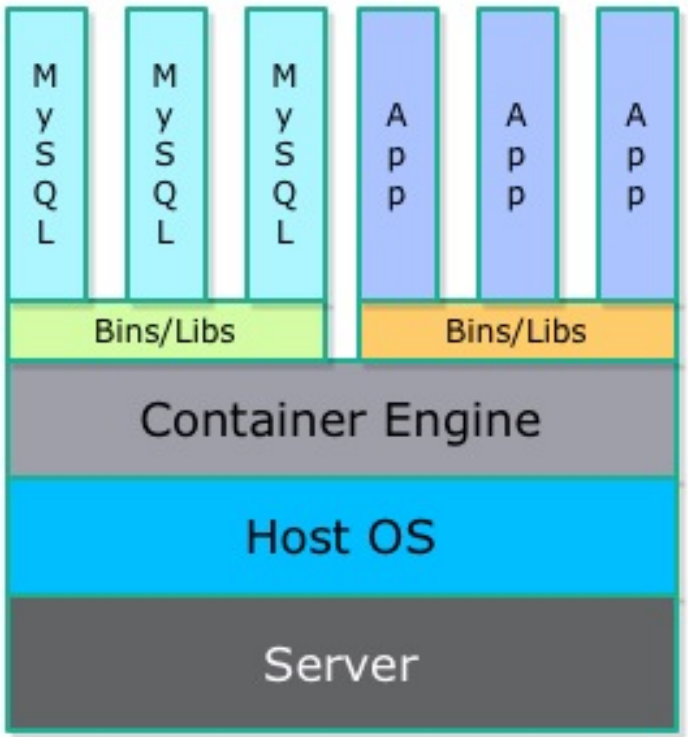
¿QUÉ SON LOS CONTAINERS?

- ¿Qué son los containers?
  - Desde lejos, parecen ser como VM.
  - Puedo instalar aplicaciones, permisos root, red filesystems, etc.
  - Pero son ambientes virtuales livianos, rápidos de iniciar (boot en ms), fácil de migrar, reproducibles y aislados.

Virtual Machines



Containers



qafe.com

DOCKER CONTAINERS

- Las aplicaciones corren en los containers.

Iniciar un container

```
$ docker run [OPTIONS] IMAGE [COMMAND] arg
$ docker run -d ubuntu apache2ctl
```

DOCKER IMAGE

- Template read-only se obtienen:
  - Repositorio público o privado
  - Construidas por Dockerfile
- Son usadas para construir Docker containers.
- Los cambios realizados se hacen a través de capas.

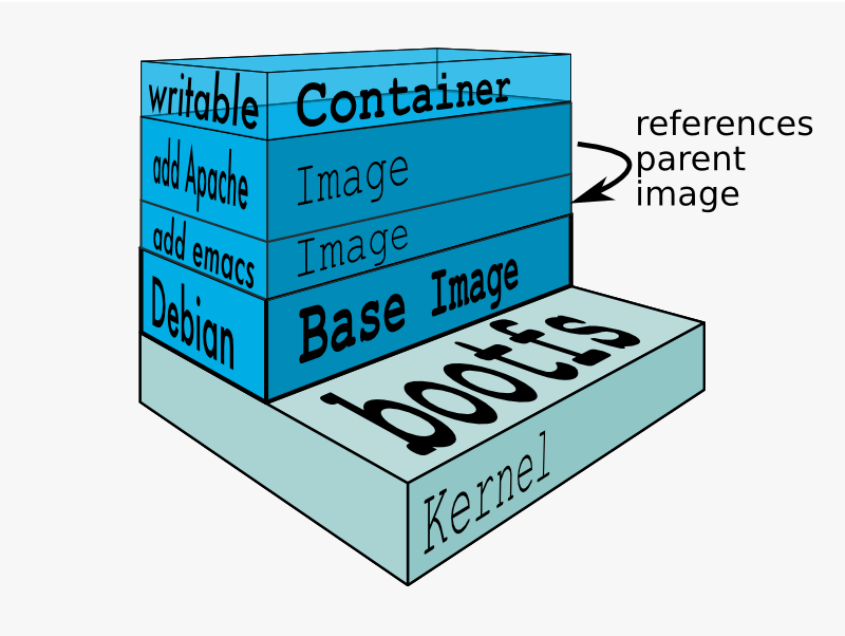


Imagen: docker.com

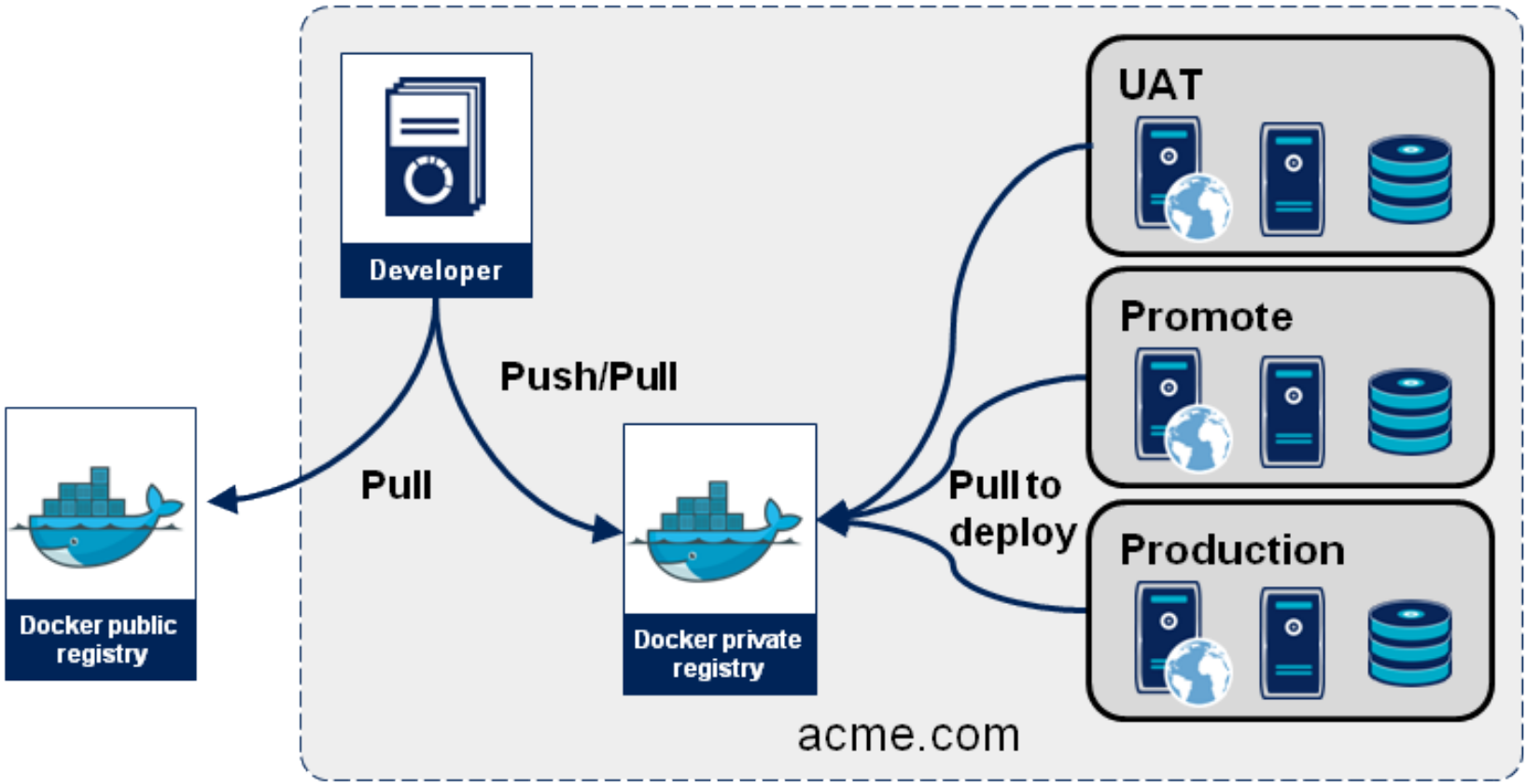
DOCKERFILE

```
1 FROM ubuntu
2 MAINTAINER Maximiliano Osorio
3 RUN apt-key adv --keyserver keyserver.ubuntu.
    com --recv 7F0CEB10
4 RUN echo "deb http://downloads-distro.mongodb
    .org/repo/ubuntu-upstart dist 10gen" | tee
     -a /etc/apt/sources.list.d/10gen.list
5 RUN apt-get update
6 RUN apt-get -y install apt-utils
7 RUN apt-get -y install mongodb-10gen
8 CMD ["/usr/bin/mongod", "--config", "/etc/
    mongodb.conf"]
```

DOCKER REGISTRY

- Repositorios públicos (Docker Hub) y privados de imágenes.
- Permite subir y bajar imágenes con sistemas operativos o aplicaciones ya configuradas.
- Imágenes son verificadas autenticidad y integridad.

DOCKER REGISTRY



http://cloud.51cto.com/

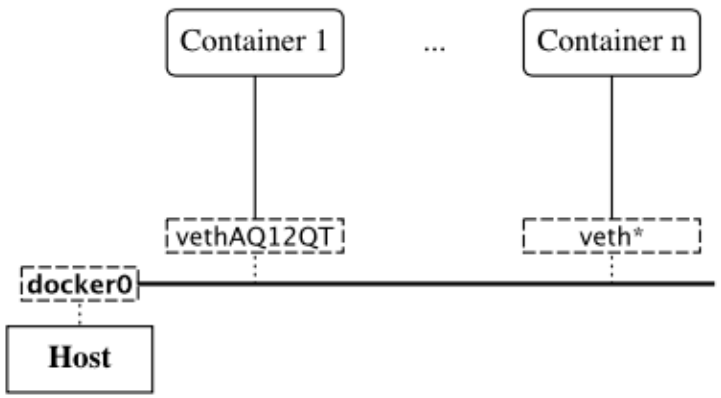
AISLAMIENTO

- **Limite de recursos:** Cgroups controlan la cantidad de recursos como CPU, memoria y disk I/O.
- **Process:** Utiliza PID namespaces (kernel $\geq$ 2,6,32), el container solo puede ver los procesos del container.
- **Filesystem:** Mismo mecanismo y mismo resultado que con los procesos. Existen filesystem que deben ser montados (ej. /sys).
- **Device:** Cgroups permite usar algunos devices [1] y bloquea la posibilidad de crear y usar otros devices.
- **IPC:** Los procesos corriendo en los containers utilizan *IPC namespaces* que permite la creación de un *IPC* separado y independiente para cada container

[1] /dev/console, /dev/null, /dev/zero, /dev/full, /dev/tty*, /dev/urandom, /dev/random, /dev/fuse

AISLAMIENTO

- **Network:** Para cada container, Docker crea una red independiente usando *network namespaces*, compuesta de su propia IP, rutas, *network devices*. Por defecto, la conexión se realiza gracias al host que provee un *Virtual Ethernet bridge* en la máquina, llamado docker0 que automáticamente realiza un *forward* de los paquetes entre las interfaces del container.



BENEFICIOS

- Despliegue rápido de aplicaciones: Los requerimientos son mínimos por lo que se reduce el tamaño y por lo tanto el despliegue es rápido.
- Portabilidad: La aplicación y todas las dependencias quedan en el container que es independiente al kernel o plataforma.
- Ambiente reproducible: La aplicación se comportará de la misma manera en otro host.
- Control de versión: Inspeccionar diferencias y hacer rollbacks.
- Re uso de componentes: Re usa capas anteriores lo cual lo hace más liviano.
- Compartir: Se puede utilizar repositorios públicos o privados de imágenes ya configuradas.
- Escalabilidad.

PRUEBAS

An updated performance comparison of virtual machines and linux containers [1]

IBM System x3650 M4 server: 2 2.4-3.0 GHz Intel Sandy Bridge-EP Xeon E5-2665 procesadores con 16 cores (plus HyperThread- ing) y 256 GB de RAM.

- CPU-PXZ: PXZ es utilidad de compresión en paralelo.
- HPC - Linpack: Linpack soluciona un sistema de ecuaciones lineales usando un algoritmo de factorización LU.
- Memory bandwidth - Stream: Un programa para benchmarking que hace operaciones simples sobre vectores.
- Random Memory Access: Estrés de forma aleatoria a la memoria

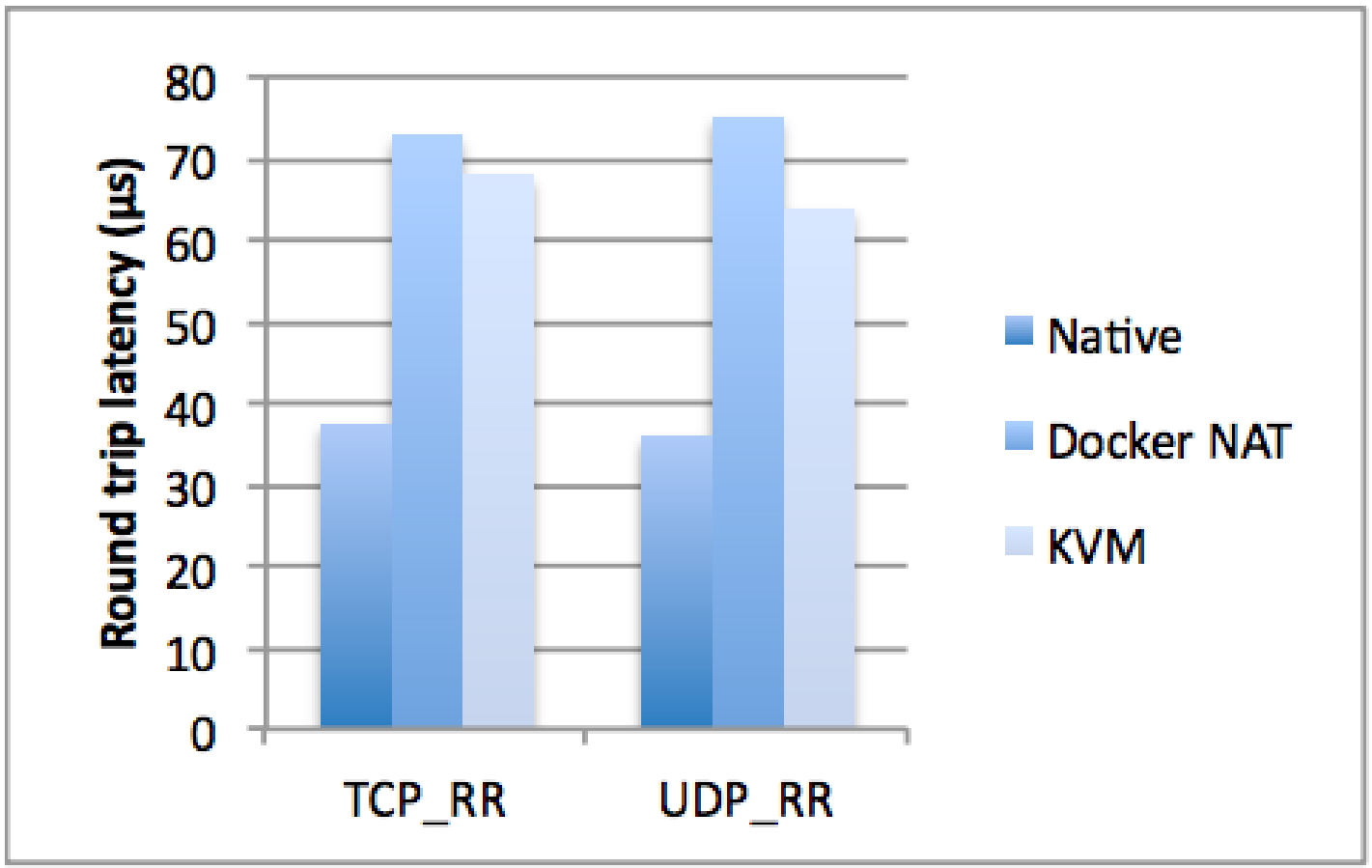
RESULTADOS

Workload		Native	Docker	KVM-untuned	KVM-tuned
PXZ (MB/s)		76.2	73.5 (-4 %)	59.2 (-22 %)	62.2 (-18 %)
Linpack (GFLOPS)		290.8	290.9 (-0 %)	241.3 (-17 %)	284.2 (-2 %)
RandomAccess (GUPS)		0.0126	0.0124 (-2 %)	0.0125 (-1 %)	Tuned run not warranted
Stream (GB/s)	Add	45.8	45.6 (-0 %)	45.0 (-2 %)	
	Copy	41.3	41.2 (-0 %)	40.1 (-3 %)	
	Scale	41.2	41.2 (-0 %)	40.0 (-3 %)	
	Triad	45.6	45.6 (-0 %)	45.0 (-1 %)	

Cuadro: Resultados

LATENCIA

- Dos máquinas idénticas conectadas por 10Gbps Ethernet.
- Cliente envia un paquete 100-byte.
- Servidor responde con un paquete 200-byte.



BLOCK I/O

- 20 TB IBM FlashSystem 840 flash SSD
- 8 Gbps Fibre Channel links

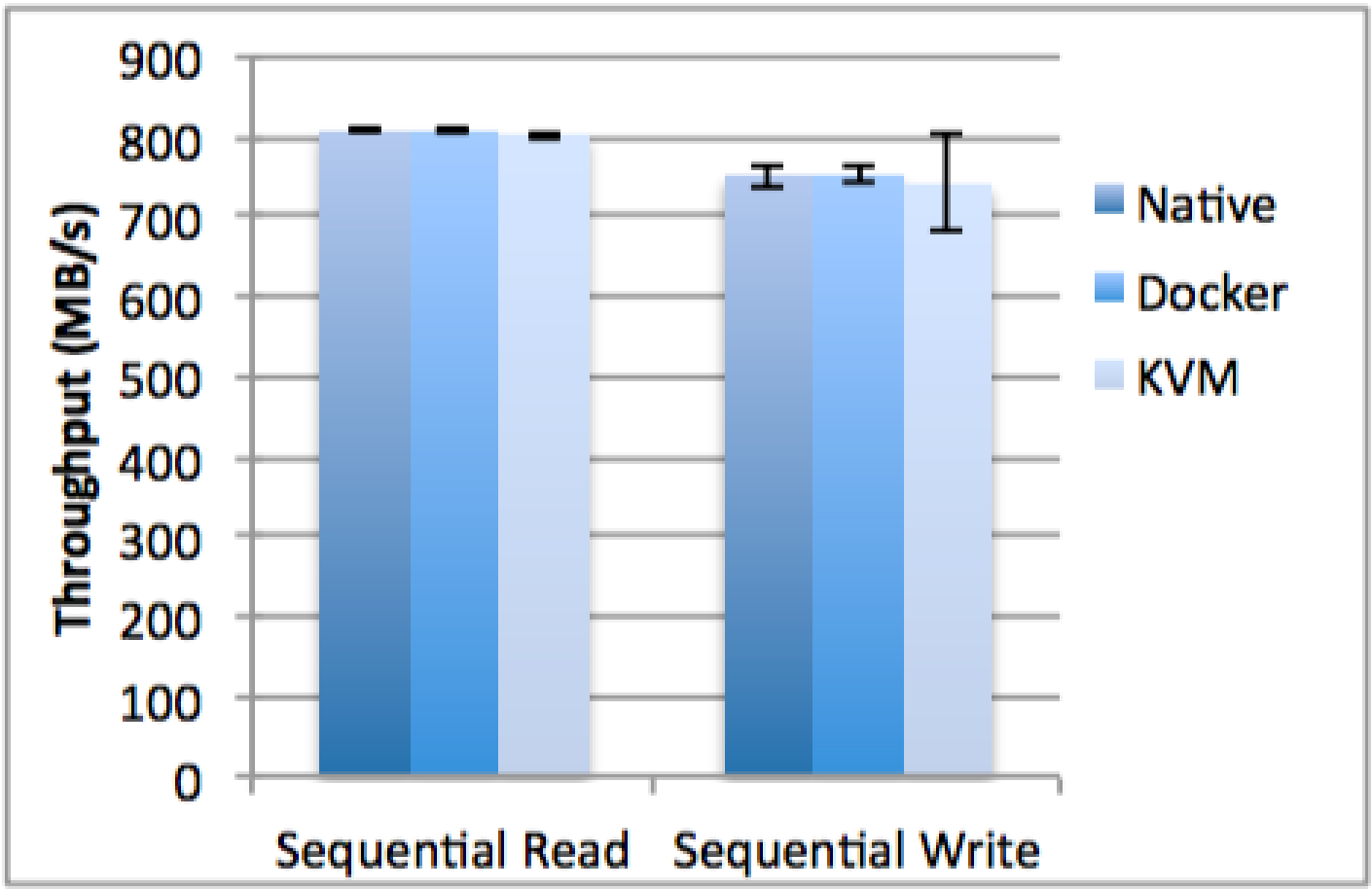


Fig. 5. Sequential I/O throughput (MB/s).

BLOCK I/O

- 20 TB IBM FlashSystem 840 flash SSD
- 8 Gbps Fibre Channel links

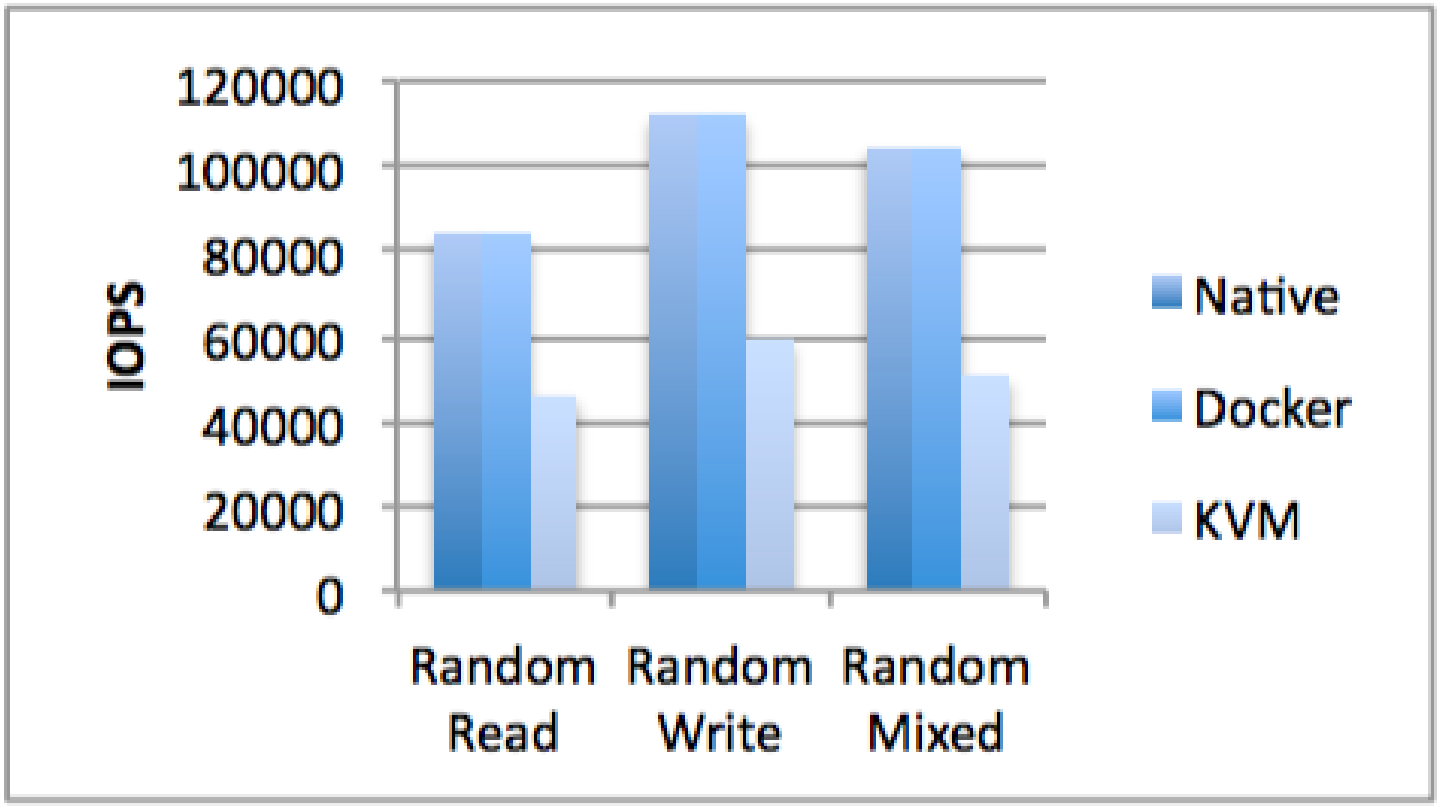


Fig. 6. Random I/O throughput (IOPS).

MYSQL

- SysBench
- Base de datos de 2 millones de registros.
- Operaciones: SELECT, 2 UPDATES, DELETE, INSERT

Configuration	Network	Storage
Native	Native	Native
Docker net=host Volume	Native	Native
Docker NAT Volume	NAT	Native
Docker NAT AUFS	NAT	AUFS
KVM	vhost-net	virtio + qcow

Cuadro: mysql configurations

MYSQL

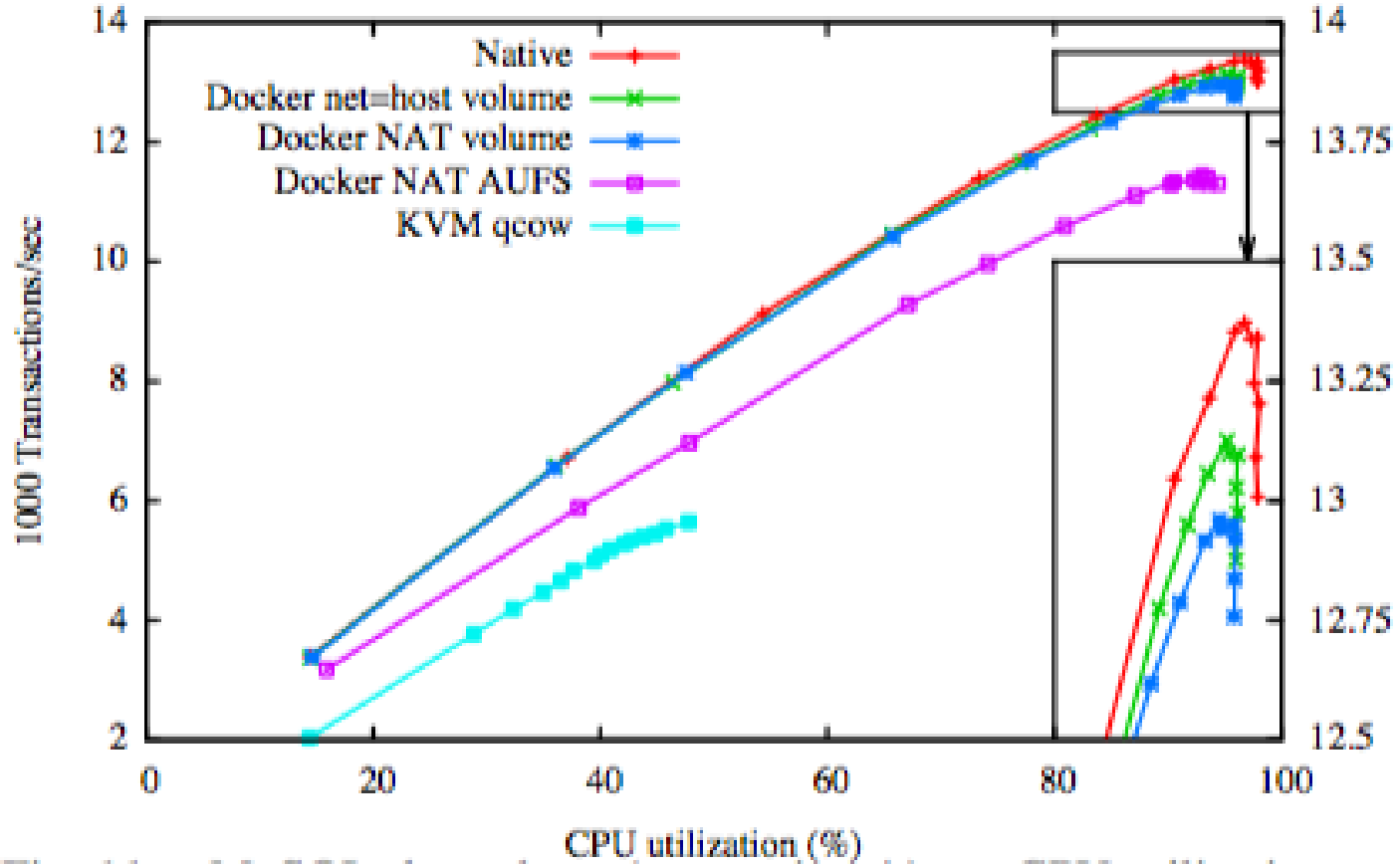


Fig. 11. MySQL throughput (transactions/s) vs. CPU utilization.

MYSQL

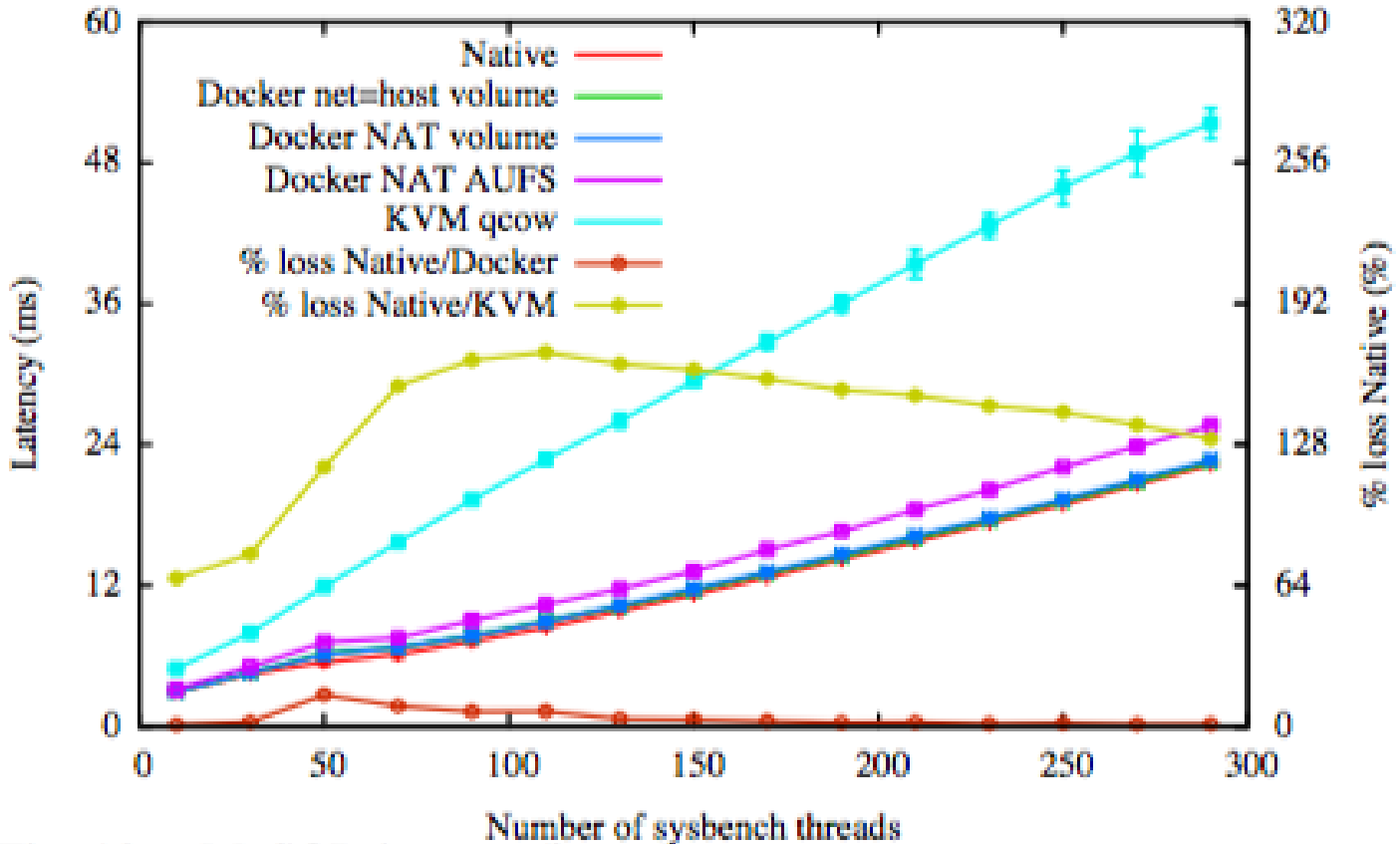
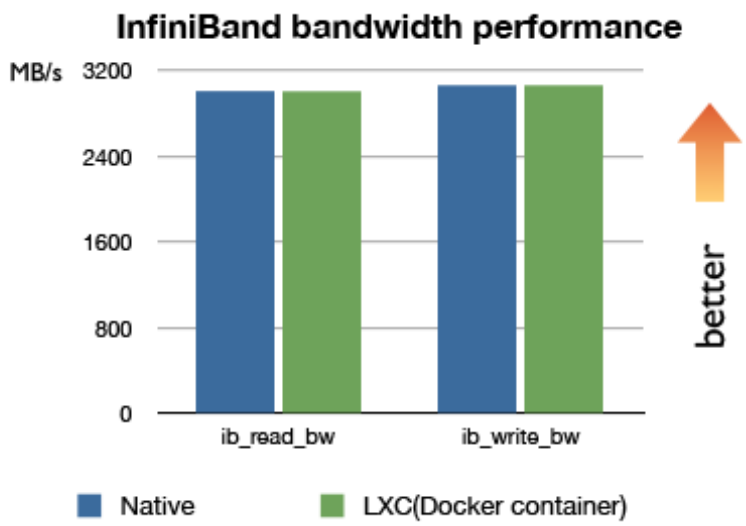
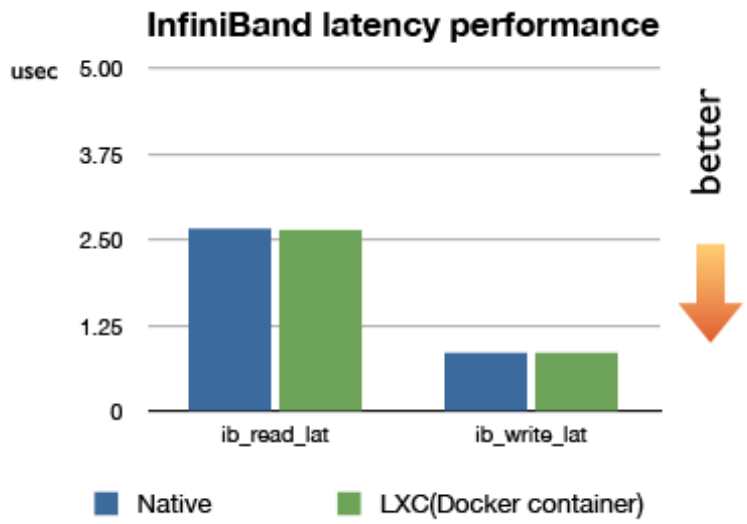


Fig. 12. MySQL latency (in ms) vs. concurrency.

INFINIBAND

- Sin Docker-nat





COMUNIDAD

Partner Ecosystem

REFERENCIAS

An updated performance comparison of virtual machines and linux containers
Felter, Wes and Ferreira, Alexandre and Rajamony, Ram and Rubio, Juan
2014

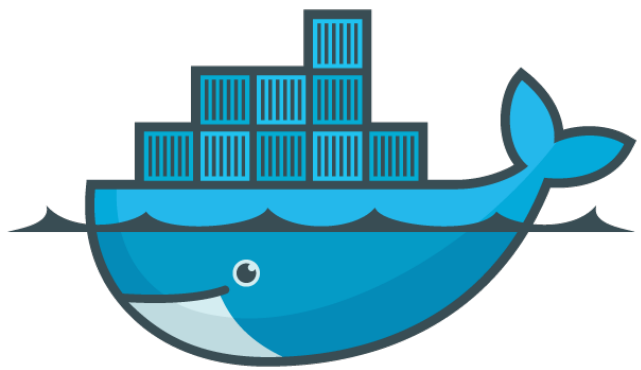
VIRTUALIZACIÓN BASADA EN CONTAINERS CON DOCKER.

Seminario CCTVal, UTFSM

27 de diciembre de 2015

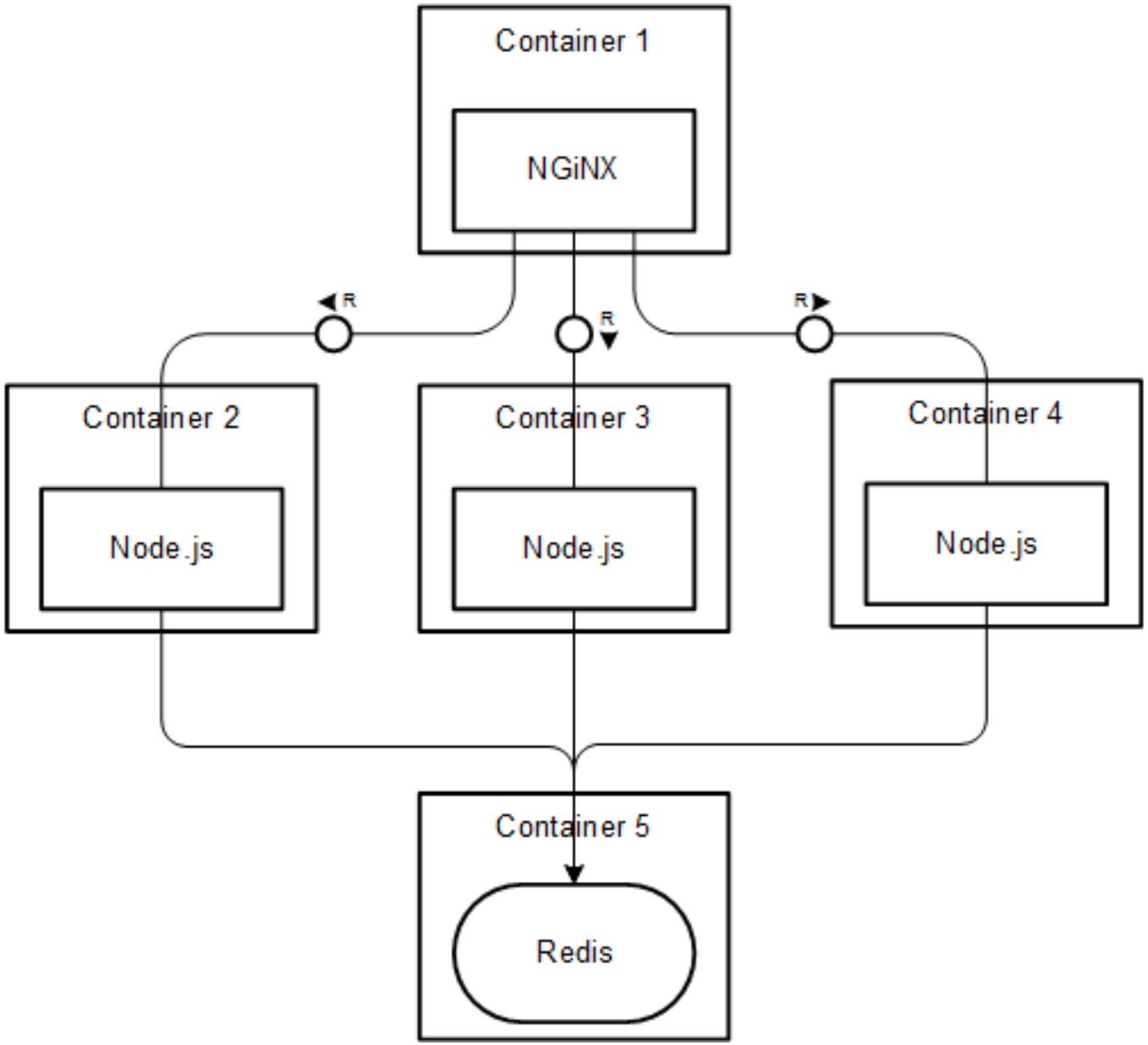
Maximiliano Osorio
<mosorio@inf.utfsm.cl>

Universidad Técnica Federico Santa María



DEMOS

DEMO


Container 1
NGiNX
R
R
R
Container 2
Node.js
Container 3
Node.js
Container 4
Node.js
Container 5
Redis

LISTANDO CONTAINERS Y IMÁGENES

```
[root@ip122 ~]# docker ps
```

```
[root@ip122 ~]# docker images
```

EJERCICIO

¿Qué vamos hacer?

- Construir una imagen de nginx.
- Correr un container que corra nginx
- Exponer el puerto para que sea accesible desde intenet.
- Usar volúmenes.

CONSTRUIR

```
1 docker build -t nombre_imagen:tag .
```

CONSTRUIR

```
1 docker run --name docker-nginx -p 8080:80
    nginx
```

- run es el comando para crear un nuevo container
- **–name** es un flag para especificar el nombre.
- **-p** especifica el puerto mapping **-p local-machine-port:internal-container-port**.
- **nginx** el nombre de la imagen.

CORRER

```
1 docker run --name docker-nginx  nginx ping
    google.cl
```

- run es el comando para crear un nuevo container
- **–name** es un flag para especificar el nombre.
- **-p** especifica el puerto mapping **-p local-machine-port:internal-container-port**.
- **nginx** el nombre de la imagen.

DETACH

```
1 docker run --name docker-nginx -p :80 -d
    nginx
```

- run es el comando para crear un nuevo container
- **–name** es un flag para especificar el nombre.
- **-p** especifica el puerto mapping **-p local-machine-port:internal-container-port**.
- **nginx** el nombre de la imagen.

CONTROL

```
sudo docker stop docker-nginx
```

```
sudo start stop docker-nginx
```

VOLUMENES

```
1 mkdir -p ~/docker-nginx/html
2 cd ~/docker-nginx/html
```

VOLUMENES

```
1 sudo docker run --name docker-nginx -p 80:80
    -d -v ~/docker-nginx/html:/usr/share/nginx
    /html nginx
```

-v especifica un enlace entre un directorio entre el host y el container

VOLUMENES

```
sudo docker stop docker-nginx
sudo docker rm docker-nginx
```

VOLUMENES

```
1 sudo docker run --name docker-nginx -p 80:80
    -v ~/docker-nginx/html:/usr/share/nginx/
    html -v ~/docker-nginx/default.conf:/etc/
    nginx/conf.d/default.conf -d nginx
2 sudo docker restart docker-nginx
```