

# Docker básico

- 1.Introducción a Docker:  
Docker versus Máquinas virtuales.**
- 2.Orquestadores que trabajan con docker:  
OpenStack,openNebula.**
- 3.Integración continua:  
modelos de éxito basado en docker (spotify) y fracasos.**
- 4.Aprovisionamiento rápido de entornos de desarrollo:  
LAB**
- 5.Más allá...  
Mesosphere/Panamax**



**Juan Carlos Rubio**

# Introducción a Docker: Docker VS Máquinas virtuales

- ¿containers Docker o Máquinas virtuales?
  - Depende...
  - Existen escenarios idóneos para cada aproximación.
  - Puede que la combinación de ambos sea la opción más propicia.

# Introducción a Docker: Docker VS Máquinas virtuales

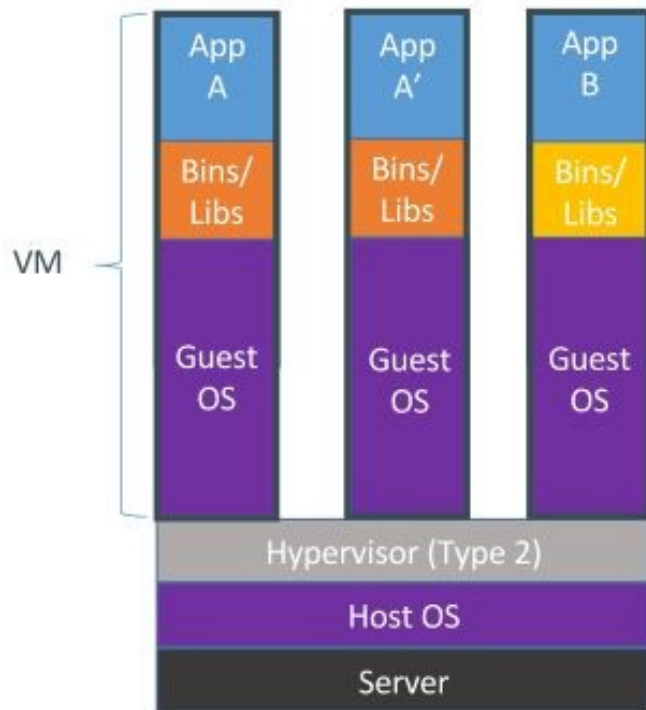
- Docker es idóneo para...
  - Cloud. Promueve la portabilidad entre diferentes entornos (cloud providers)
  - Es una plataforma abierta para desarrolladores. Permite aislar dependencias de las aplicaciones aislándolas en contenedores.
  - Los contenedores son más escalables y seguros que otras estrategias.

# Introducción a Docker: Docker VS Máquinas virtuales

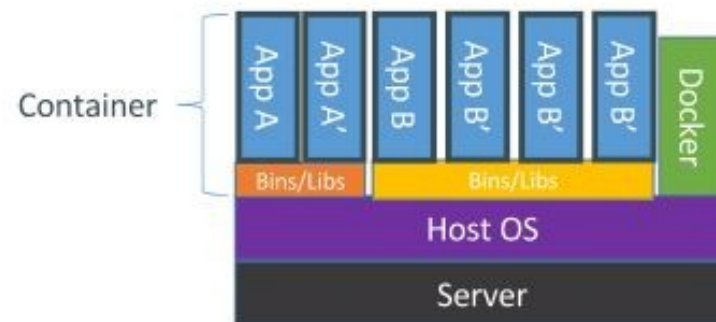
- ¿En qué difieren los containers Docker de las máquinas virtuales?
  - VM tienen un SO completo con su propia gestión de memoria y la sobrecarga de los virtual device drivers. Tienes más “capas” de complejidad que los containers Docker.
  - Docker usa UAFS para ahorrar espacio. Los containers son “procesos” limitados o enjaulados. Menos complejidad.
  - Los servicios son PROCESOS dentro del docker host

# Introducción a Docker: Docker VS Máquinas virtuales

- Diferencias de almacenamiento:



Los Containers están aislados pero comparten un mismo SO y, cuando pueden, binarios y/o librerías.



# Introducción a Docker: Docker VS Máquinas virtuales

- **IMPORTANTE:**
  - Los contenedores Docker son ejecutados por el Docker engine (NOTA: instalar docker-engine, no “docker.io”) en lugar de por un hypervisor.
  - Los containers son más pequeños,
  - se inician antes,
  - rinden mejor y
  - comparten kernel

# Introducción a Docker:

## Docker VS Máquinas virtuales

- Docker containers (DC) versus VM
  - DC comparten kernel y librerías=> cargan menos memoria.
    - DC 1, MV 0
  - Las MV tienen mejor aislamiento (Intel Vtd/Vtx). DC no tiene aislamiento hardware
    - DC 1, MV 1
  - En media, una aplicación puede llegar a ejecutarse dos veces más rápido que en MV
    - DC **2**, MV 1 => **DC WINS!**



# Máquinas Virtuales y Containers: *mejor juntos*

- Existen escenarios de distribución de cargas de trabajo en los que es más apropiado utilizar máquinas físicas.
  - Caso 1: BD Oracle.
    - Oracle no ofrece soporte de su BD en contenedores.
    - Oracle sólo ofrece soporte de su BD en instalación bare metal o con su propio hypervisor.
  - Caso 2: appliances de terceros, con los que nuestros containers deben poder comunicarse.
  - Caso 3: clusters de BD preexistentes o remotos.
  - Caso 4: Software PARA WINDOWS o PARA SPARC
- Puede utilizarse un escenario mixto con máquinas físicas y contenedores

# Máquinas Virtuales y Containers: *mejor juntos*

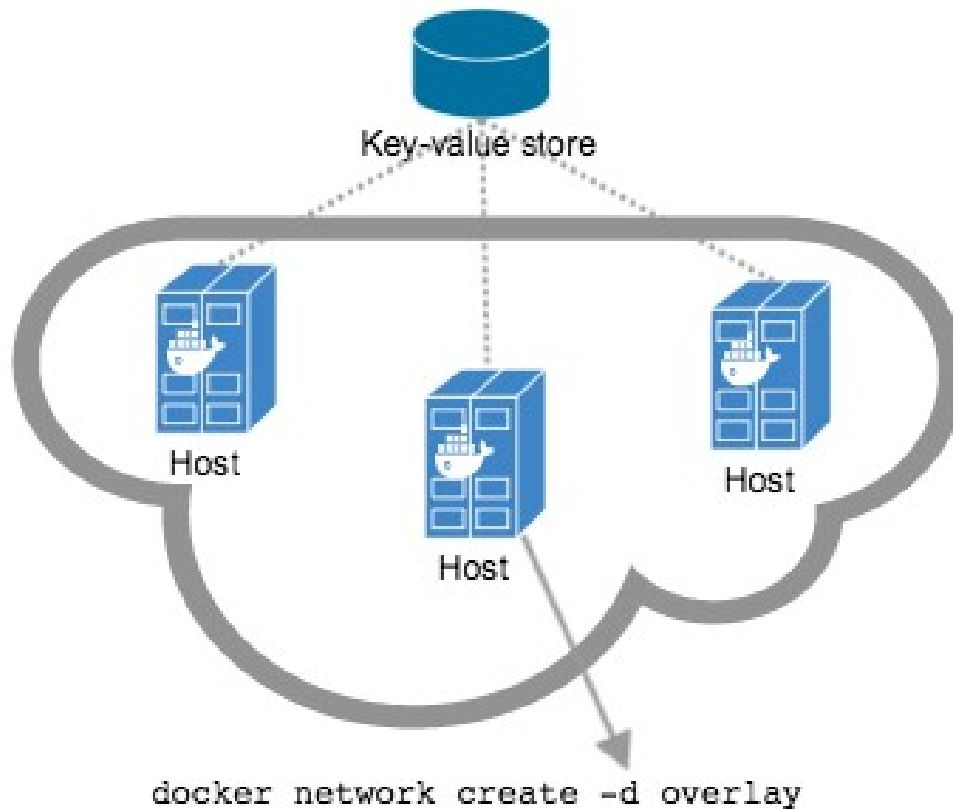
- Esquemas híbridos:
  - Los entornos productivos no se sostienen únicamente con MV y contenedores.
  - La gestión de la capacidad, seguridad y rendimiento pueden requerir herramientas aún no disponibles para entornos de contenedores, aunque están apareciendo.

# Máquinas Virtuales y Containers: *mejor juntos*

- En esquemas híbridos, tendremos que habilitar modos de red “más compleja”, en “OVERLAY”:



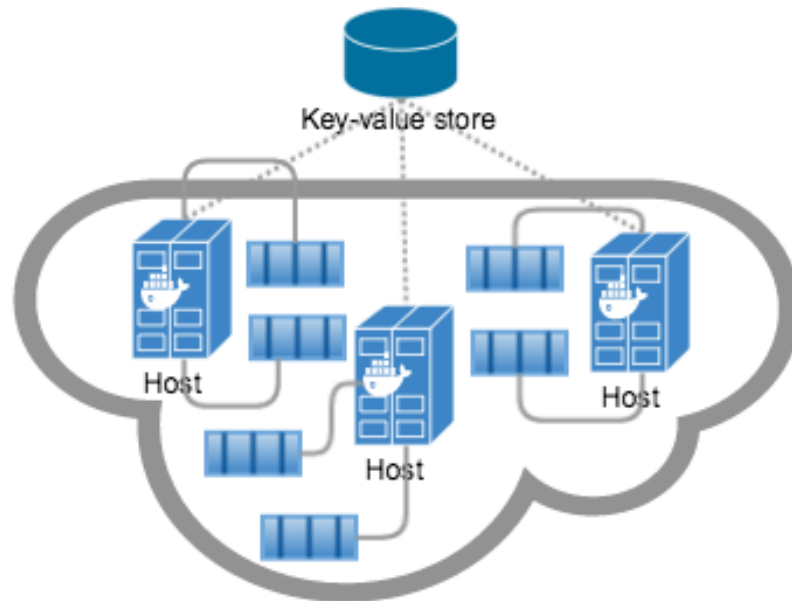
# Máquinas Virtuales y Containers: *mejor juntos*



# Máquinas Virtuales y Containers: *mejor juntos*

```
$ docker run -itd --net=my-multi-host-network busybox
```

Once connected, each container has access to all the containers in the network regardless of which Docker host the container was launched on.

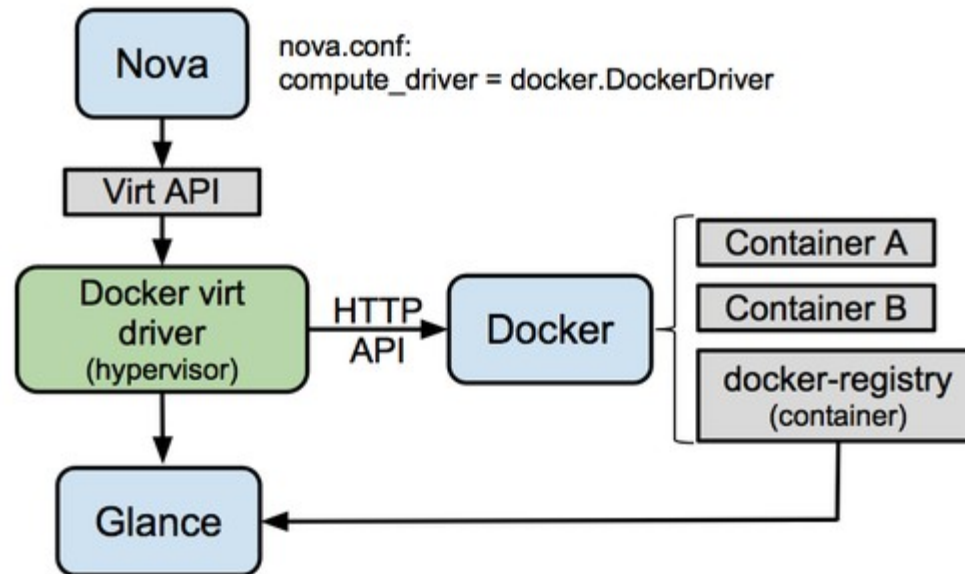


# Máquinas Virtuales y Containers: *mejor juntos*

- Conclusión:
  - VM proporcionan alta flexibilidad
  - DC pone el foco en aplicaciones y sus dependencias.
  - DC promete sencillez al traspasar pilas de aplicación entre proveedores de cloud.
  - Disponemos de herramientas y documentación suficientes para montar nuestro cluster Docker y comenzar a trabajar en este modelo.

# Orquestadores que trabajan con Docker.

- OpenStack: incorpora un “driver” de hypervisor para OpenStack Nova Compute:

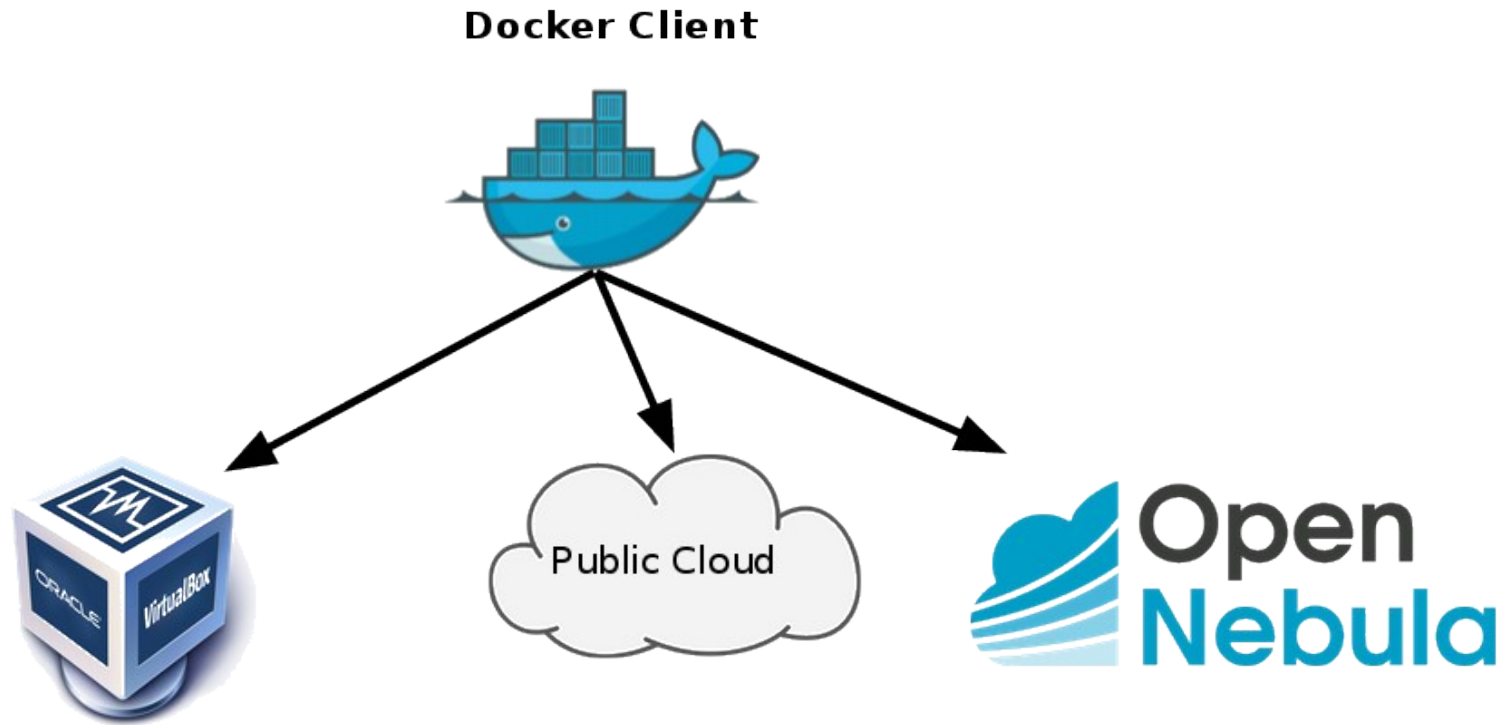


# Orquestadores que trabajan con Docker.

- OpenNebula:
  - Docker machine te permite crear Docker Hosts en nuestra computadora, en proveedores de cloud y en nuestro propio DataCenter.
  - La integración de OpenNebula y docker machine es muy simple



# Orquestadores que trabajan con Docker.



# Orquestadores que trabajan con Docker.

- OpenNebula Docker Driver y Datastore
  - **ONEDock** es un conjunto de extensiones para OpenNebula que permite usar contenedores Docker como si fueran MV ligeras.
  - Docker se configura para que actúe como un hipervisor, de manera que se comporta como KVM por ejemplo.
  - La idea subyacente es que cuando a OpenNebula se le pide una MV, desplegará un container Docker en su lugar.
  - El usuario dispondrá de una IP para gestionar el contenedor.

# Integración continua

- Definición: La **integración continua** (continuous integration en inglés) es un modelo informático propuesto inicialmente por Martin Fowler que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible [..] (compilación y ejecución de pruebas de todo un proyecto)
- El proceso suele ser: cada cierto tiempo (horas), apoyado en Git, Subversion, ejecutar pruebas y generar informes

# Integración continua

- Contraste a la CI basada en Docker
  - Aproximación relativamente nueva (surgió en el 2013)
  - Sustentada sobre conceptos de virtualización basada de contenedor, en lugar de un modelo de hipervisor, sólidamente consolidado y aprobado por la industria IT como estándar
  - Incluso el formato de contenedor es diferente de otras alternativas como Virtuozzo / openvz

# Integración continua

- Pros:
  - Docker está muy orientado a procesos de entrega de software.
  - Favorece el trabajo sostenido hacia el modelo DevOps => Despliegue Continuo.
  - Cómodo para desarrolladores y para administradores de sistema
  - Proporciona capacidades para entregas de software complejas con unas herramientas sencillas
  - Proporciona fácilmente entornos idóneos y rápidamente provisionables a los Desarrolladores... y mucho más.

# Integración continua

- Developer:
  - ¿Necesitas un entorno específico para trabajar? Busca en DockerHub y móntatelo tú mismo.
  - ¿necesitas adaptarlo? Construye tu propia imagen
  - ¿Necesitas distribuirlo y desplegarlo en otro entorno (PREPRO, PRO), y el entorno es COMPLEJO?
    - Usa un **Dockerfile** más un **fig.yml** (para FIG) o un **docker-compose.yml** (para Docker Compose).
  - ¿Quieres almacenar imágenes en un registro interno, y no en DockerHub? Instálate un registro privado.

# Integración continua

- Sysadmin
  - ¿Frustrado con entregas de software cuyos manuales contienen textos con tareas que deberían ser automatizables y reproducibles sin errores de interpretación? Pasa a modelos Dockerfile/Docker Compose u Orquestados.
  - Si Dockerfile genera problemas de despliegue, se devuelve a Desarrollo. Se puede trabajar conjuntamente con desarrollo porque ambos grupos de técnicos observan un mismo modelo para entenderse.
  - Se reducen las fricciones

# Integración continua

- Casos de éxito
  - Battlefy: Docker + Jenkins
    - Construye y sube imágenes Docker antes de desplegarlas en AWS Elastic Beanstalk
    - Comienzan con una petición en Github, lo enlazan a un ticket JIRA
    - Los resultados se envían a un equipo que lleva el código a AWS S3 donde se usan Docker containers para construir un entorno de preproducción.
    - Después de unos tests, Jenkins es capaz de automatizar la entrega a PRODUCCIÓN.



# Integración continua

- Casos de éxito (II)
  - Iron.io: creadores de IronMQ, un sistema de encolado de mensajes. Y IronWorker una herramienta de procesamiento de tareas asíncrono
    - IronWorker tiene 15 stacks de imágenes Docker en almacenamiento en bloque (cabines de fibra) con empaquetado de código de usuario en cada nuevo contenedor, que ejecuta el proceso y después destruye el contenedor.
    - Iron.io trabajan en un contexto de microservicios que no está disponible en entornos de producción al uso
    - Utilizan software de orquestación

# Integración continua

- Casos de éxito (III) SPOTIFY
  - Spotify tiene cientos de contenedores ofreciendo servicio en producción
  - Tiene un servicio monetizado, y confían en docker.
  - Han creado su propio sistema de orquestación: HELIOS. Es simple, aunque adolece de falta de límites en los contenedores (CPU, memoria...)
  - Su integración continua y de despliegue es muy intensiva.
  - A veces complementan despliegue de configuraciones con puppet.

# Integración continua

- Caso de FRACASO
  - XMLDirector: es una plataforma XML gestor de contenidos y de flujos de trabajo.
  - Andreas Jung (líder de proyecto) quiso colocar bases de datos de tipo XML en contenedores.
  - Midió que los builds normales tardaban de 5 a 10 veces más que con la shell. Algunos procesos requirieron reiniciar Docker.
  - *“Docker es un lío, pero la idea es buena”*

# Integración continua

- Caso a imitar: IIEPE (Instituto de Investigación, Innovación y Estudios de Posgrado para la Educación, México)
  - 1) Usan varios sites con Drupal, PHP y Node.js
  - 2) Todo desarrollador usa Docker para crear la aplicación
  - 3) La instancia Gitlab (que tiene una herramienta de despliegue y de integración continua) tiene configurado WebHooks, con lo que cuando se da un “push” al proyecto, ordena a Jenkins que ejecute una tarea

# Integración continua

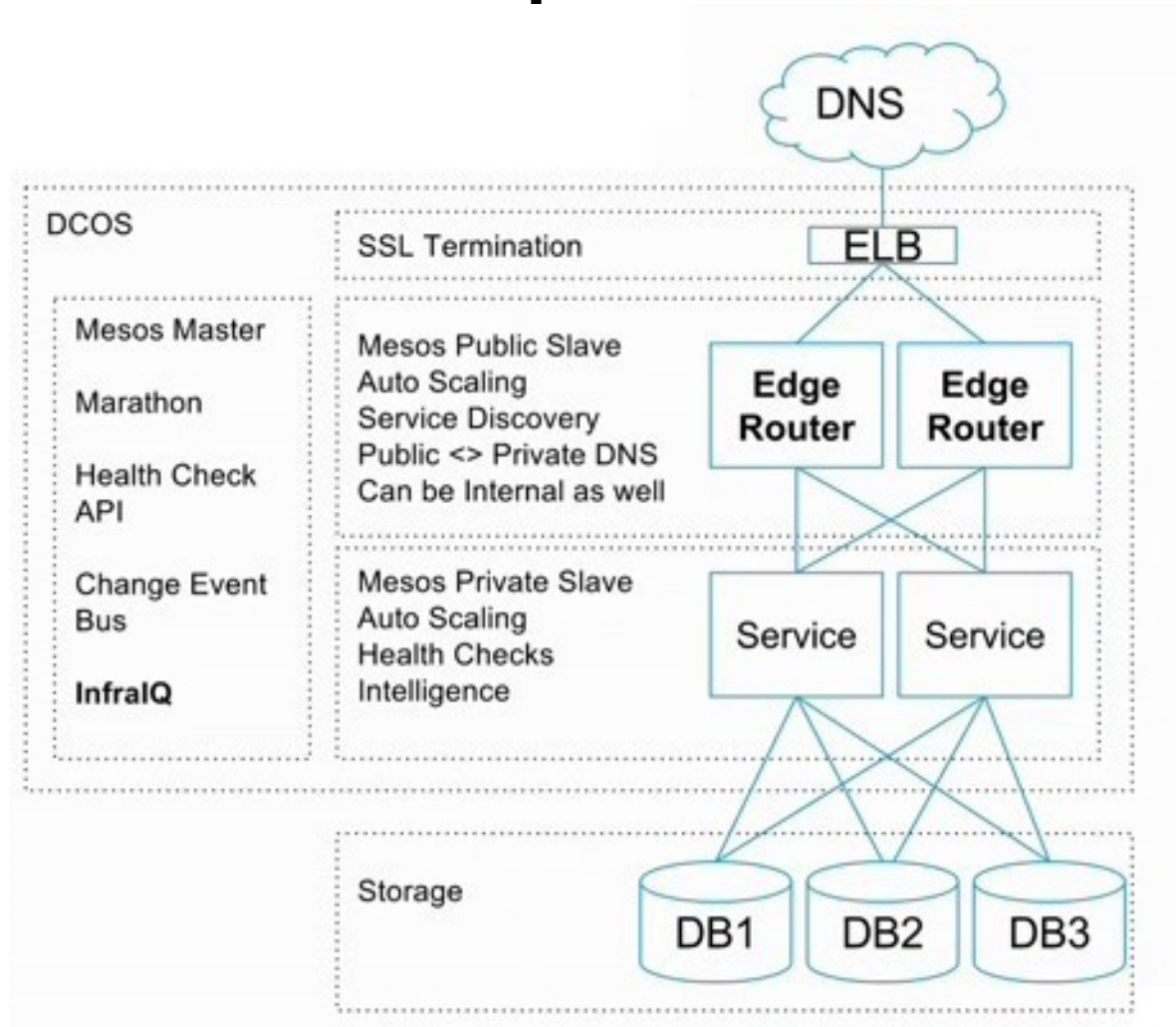
- Caso a imitar: IIEPE (II)
  - 4) Cada tarea jenkins clona el último código de gitlab, ejecuta tests, se loga en el registro privado Docker, construye una nueva imagen con la última versión y la sube al registry.
  - 5) Finalmente, Maestro-NG (software de orquestación) despliega la nueva versión de la imagen.
  - 6) El balanceador detecta el cambio y recarga la nueva configuración.

# ¿Cómo triunfar implantando Docker en producción?

- **Salesforce:**

- Empresa que ha creado un CRM con más de 100.000 clientes
- Ha trabajado intensamente buscando un modelo cuyo pilar sea docker, a través de ensayo y error.
- Modelo plenamente exitoso con el esquema siguiente:

# ¿Cómo triunfar implantando Docker en producción?



# Integración Continua

- Conclusiones
  - Docker ha llegado para quedarse.
  - Es el futuro (y presente) de la CI y la CD.
  - Muchas empresas están invirtiendo en ello.
  - El escenario que terminará por implantarse será sin duda, **híbrido**.



# Integración continua

- Esto es complicado...¿Qué escenario elijo?
  - Docker Swarm: es el clustering nativo de Docker. Bueno para empezar, junto con Consul/kubernetes para discovery.
  - Permite crear un pool de Docker Engines en un único “Virtual Host”.
  - Es habitual usar Jenkins y Git como herramientas en las que apoyarnos para la CI
  - Es un comienzo....

# LAB

- LAB En linux (compose no disponible en windows):
  - <https://github.com/sameersbn/docker-gitlab>

The quickest way to get started is using `docker-compose`.

```
wget https://raw.githubusercontent.com/sameersbn/docker-gitlab/master/docker-compose.yml
```

Generate a random string and assign to `GITLAB_SECRETS_DB_KEY_BASE` environment variable. Once set you should not change this value and ensure you backup this value.

**Tip:** You can generate a random string using `pwgen -Bsv1 64` and assign it as the value of `GITLAB_SECRETS_DB_KEY_BASE`.

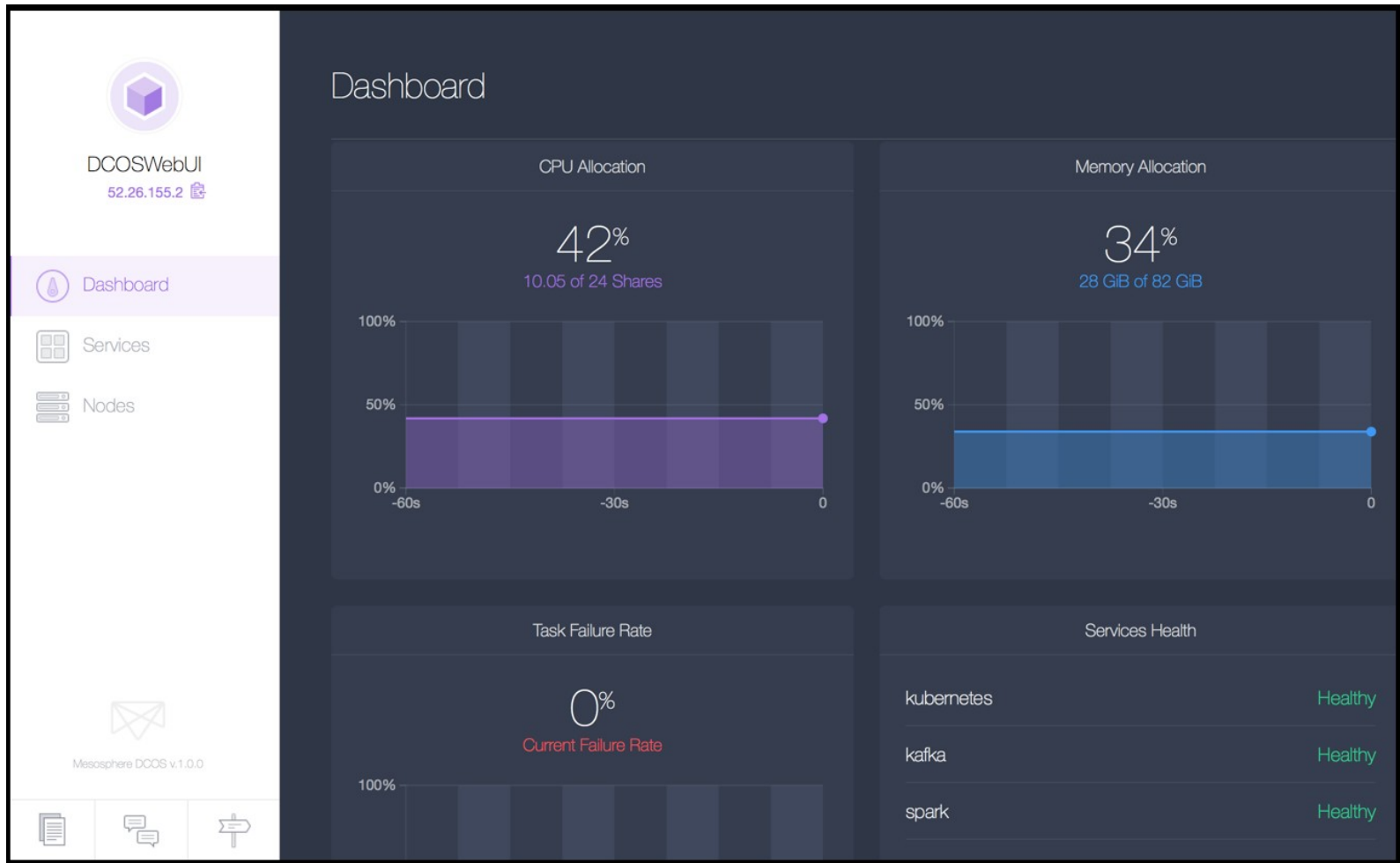
Start GitLab using:

```
docker-compose up
```


# Más allá?

- Mesosphere:
  - [https://www.youtube.com/watch?feature=player\\_embedded&v=0I6qG9RQUnY](https://www.youtube.com/watch?feature=player_embedded&v=0I6qG9RQUnY)
- PANAMAX
  - <https://www.youtube.com/watch?v=xGjBZ0IZG5E#t=87>

# Mesosphere



# Mesosphere

  
DCOSWebUI  
52.26.155.2

Dashboard

**Services**

Nodes

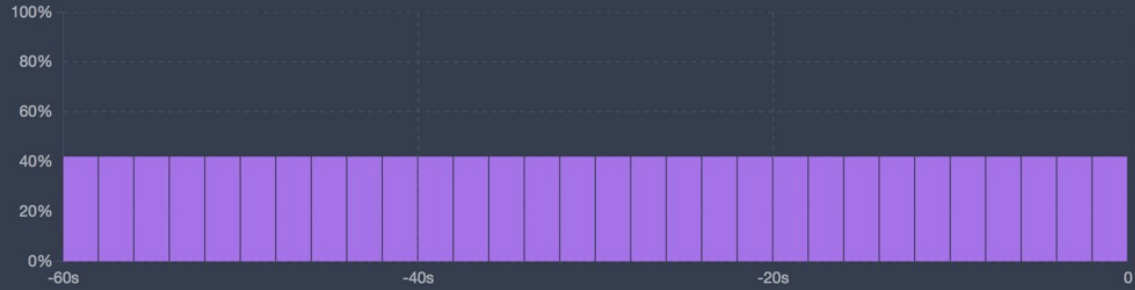
Mesosphere DCOS v.1.0.0

## Services

CPU Memory Disk

### CPU Allocation Rate

8 Total Services






Service Name	Health	Tasks	CPU	Mem	Disk
cassandra.dcos	Healthy	6	0.9	3.5 GiB	0.8 GiB
chronos	Healthy	0	0	0 B	0 B
hdfs	Healthy	9	2.65	19.2 GiB	0 B


8 Services

All 8 Healthy 7 Unhealthy 0 N/A 1

Filter

SERVICE NAME ^	HEALTH	TASKS	CPU	MEM	DISK
 cassandra.dcos	Healthy	6	0.9	3.5 GiB	0.8 GiB
 chronos	Healthy	0	0	0 B	0 B
 hdfs	Healthy	9	2.65	19.2 GiB	0 B

# Mesosphere

  
DCOSWebUI  
52.26.155.2

Dashboard  
Services  
Nodes

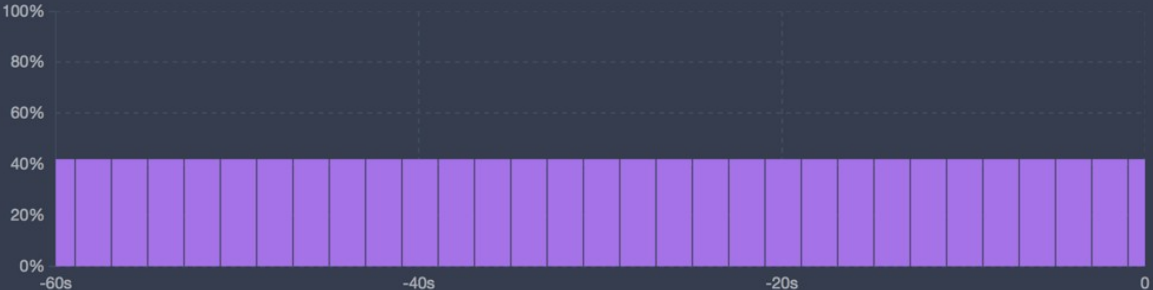
Mesosphere DCOS v1.0.0

## Nodes

CPU Memory Disk

### CPU Allocation Rate

6 Total Nodes



6 Nodes

Filter by Service

List Grid

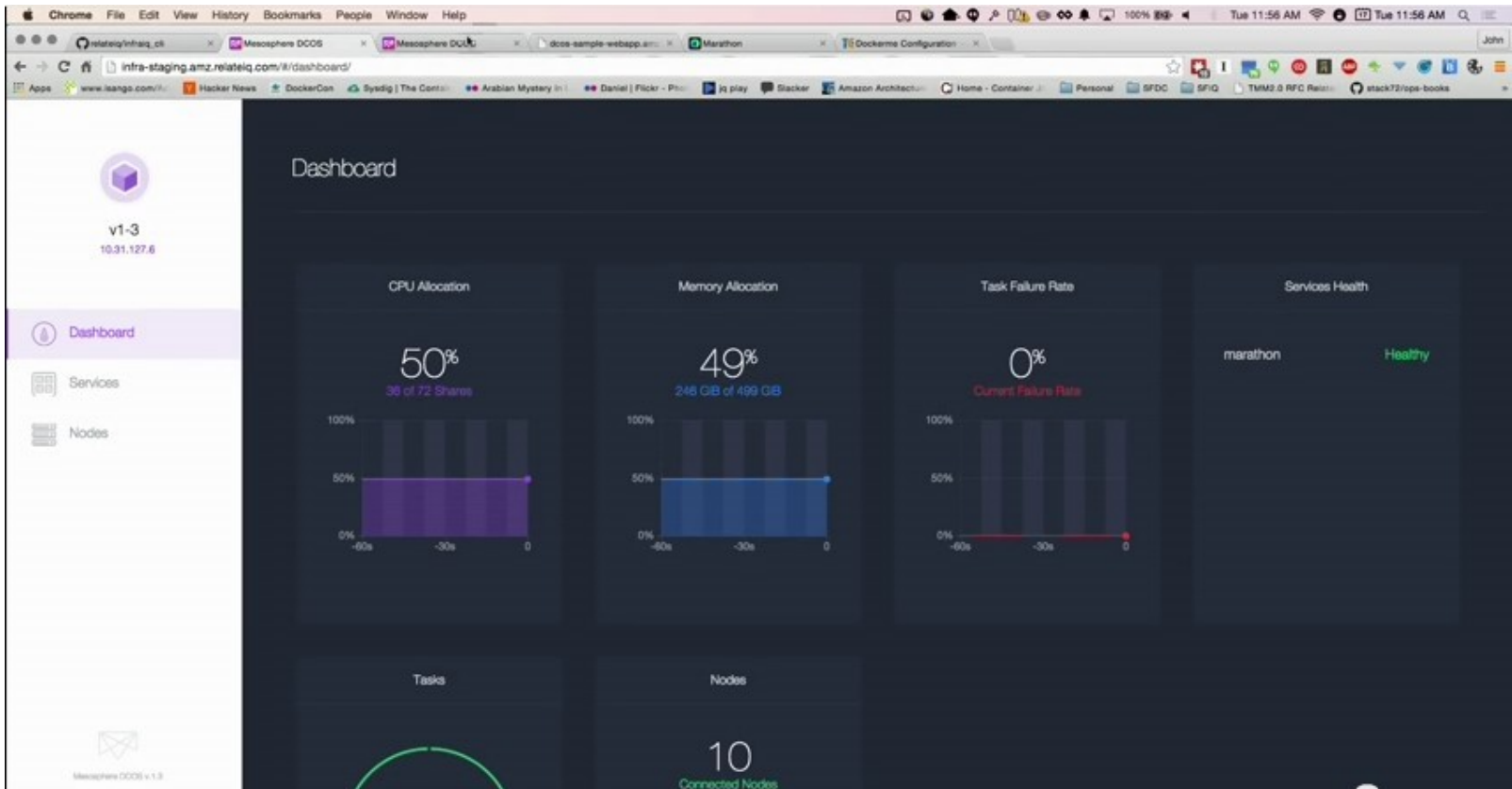
HOSTNAME	TASKS	CPU	MEM	DISK
ip-10-0-3-113.us-west-2.compute.internal	4	66%	23%	1%
ip-10-0-3-114.us-west-2.compute.internal	7	95%	74%	0%
ip-10-0-3-115.us-west-2.compute.internal	5	28%	63%	1%

# Mesosphere

The screenshot shows the Mesosphere Marathon web interface. At the top, a purple header displays 'marathon Healthy (32 tasks)'. Below this, the 'Apps' section shows a deployment for '/dcos-sample-webapp' in a 'Deploying' state. A table lists the tasks, with the most recent one in a 'Staged' state, circled in red. The table has columns for ID, Status, Version, and Updated.

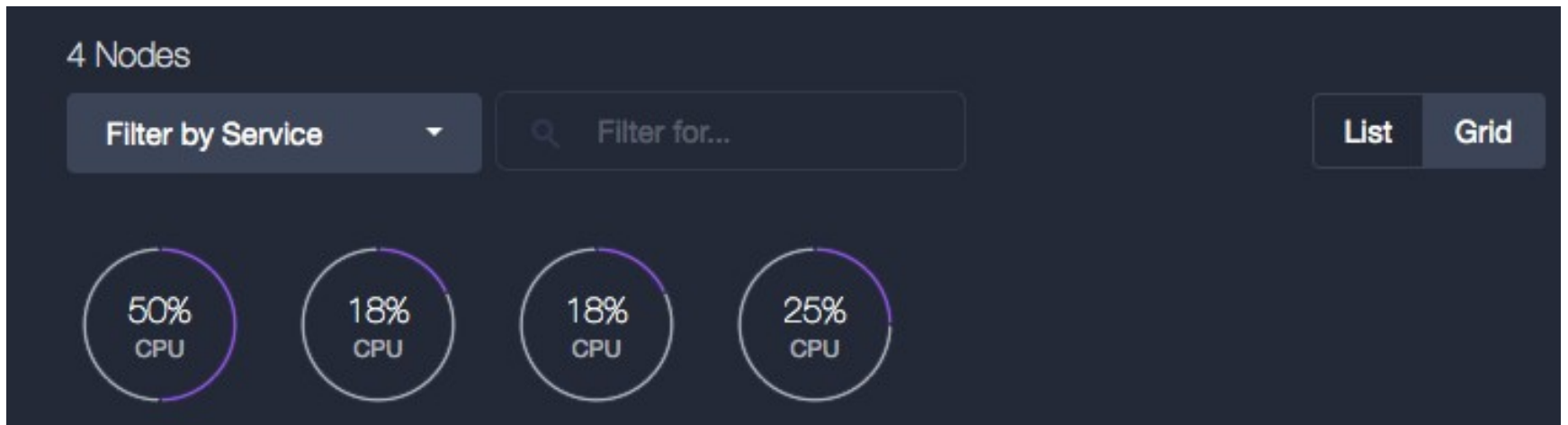
ID	Status	Version	Updated
dcos-sample-webapp.ede1469f-88cc-11e5-b11c-cae4abf417c9 ip-10-31-122-219.us-west-2.compute.internal:25259	Started	5 days ago	11/12/2015, 12:35:32 AM
dcos-sample-webapp.642dcae3-8cf4-11e5-b11c-cae4abf417c9 ip-10-31-125-194.us-west-2.compute.internal:21539	Started	a day ago	11/17/2015, 7:28:31 AM
dcos-sample-webapp.0d43ae21-8d1a-11e5-b11c-cae4abf417c9 ip-10-31-131-155.us-west-2.compute.internal:9753	Staged	a few seconds ago	11/17/2015, 11:57:53 AM

# Mesosphere





# Mesosphere



# Mesosphere

The screenshot shows the Mesosphere Marathon web interface. At the top, the browser address bar displays the URL `infra-staging.amz.relateiq.com/services/ui/marathon`. The page header includes a purple navigation bar with the text "marathon Healthy (32 tasks)" and a "Back" button. Below the header, the main content area features a dark theme with a "MARATHON" logo and a "Filter list" search box. A table lists various applications, each with columns for ID, Memory (MB), CPUs, Tasks / Instances, Health, and Status. All listed applications are in a "Running" state. A "+ New App" button is visible in the top right corner of the application list.

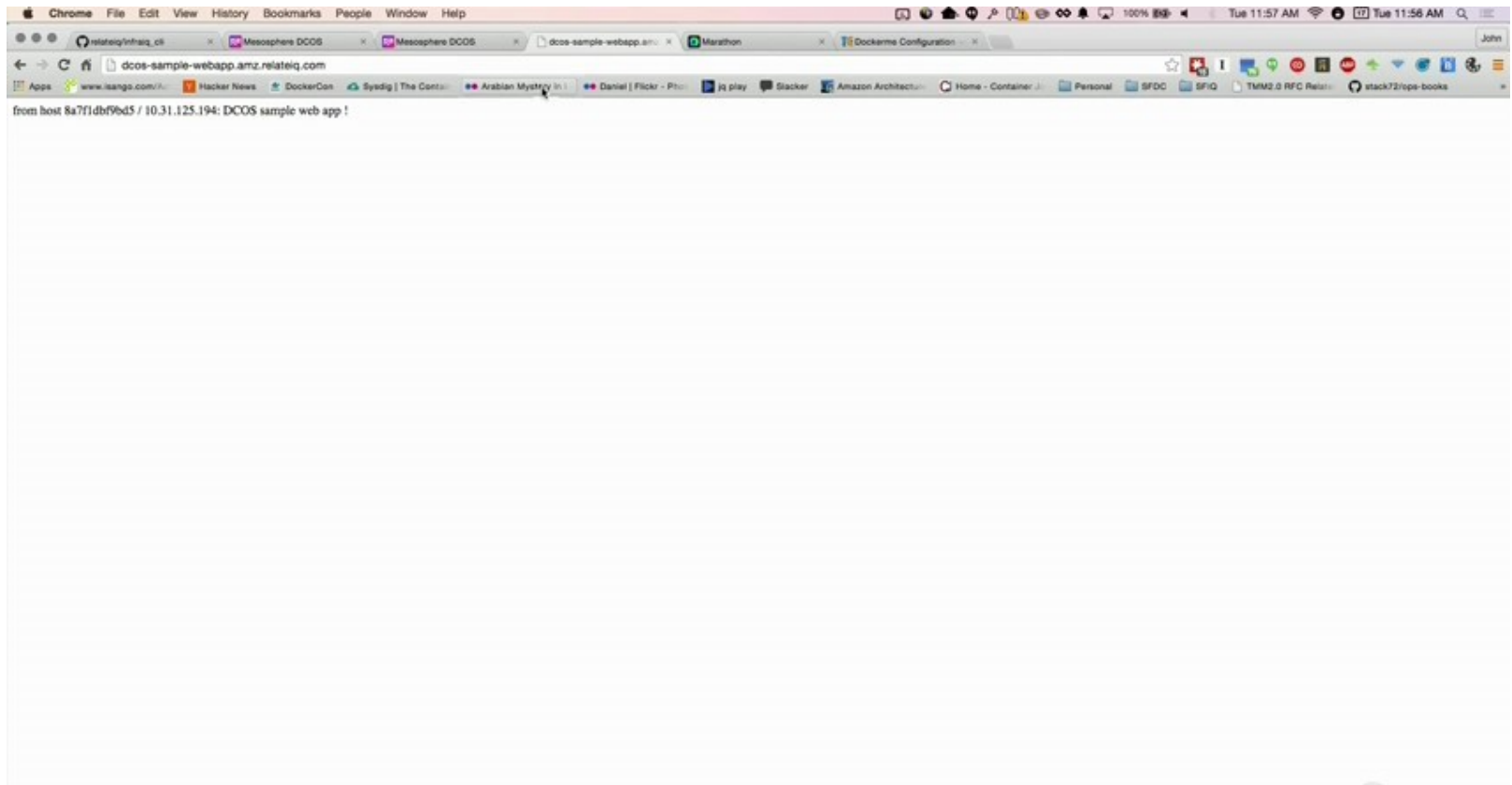
ID	Memory (MB)	CPUs	Tasks / Instances	Health	Status
/bedrock	512	0.5	1 / 1	██████████	Running
/cm-web-express	1024	1	1 / 1	██████████	Running
/dcoo-sample-webapp	1024	1	2 / 2	██████████	Running
/epiv2dm-siqas-satecloud	10000	1	1 / 1	██████████	Running
/idm-appdynamics	10000	1	1 / 1	██████████	Running
/idm-better-http-code	10000	1	1 / 1	██████████	Running
/idm-contact-prope-in-grid	10000	1	1 / 1	██████████	Running
/idm-cument-r	10000	1	1 / 1	██████████	Running
/idm-dcoo-int	10000	1	1 / 1	██████████	Running
/idm-desk-outbound	10000	1	1 / 1	██████████	Running
/idm-leq-refactor-v2	10000	1	1 / 1	██████████	Running
/idm-invitalmageemail	10000	1	1 / 1	██████████	Running
/idm-jax1	10000	1	1 / 1	██████████	Running
/idm-lispaging	10000	1	1 / 1	██████████	Running
/idm-megan-config-resource-update	10000	1	1 / 1	██████████	Running
/idm-new-schema	10000	1	1 / 1	██████████	Running
/idm-newrelic	10000	1	1 / 1	██████████	Running

# Mesosphere

The screenshot displays the Mesosphere Marathon web interface. At the top, a purple header shows the 'marathon' logo and 'Healthy (32 tasks)'. Below this, the 'MARATHON' logo is visible, along with 'App' and 'Deployments' tabs. The main content area shows the application '/dcos-sample-webapp' in a 'Running' state. There are buttons for 'Suspend', 'Scale', 'Restart App', and 'Destroy App'. A 'Tasks' tab is selected, showing a table of running tasks.

ID	Status	Version	Updated
dcos-sample-webapp.e5e1469f-85cc-11e5-b11c-cae4ab417c9 ip-10-31-122-219.us-west-2.compute.internal:25259	Started	5 days ago	11/12/2015, 12:35:32 AM
dcos-sample-webapp.642dcae3-8cf4-11e5-b11c-cae4ab417c9 ip-10-31-126-194.us-west-2.compute.internal:21539	Started	a day ago	11/17/2015, 7:28:31 AM

# Mesosphere



# Mesosphere

The screenshot shows the Mesosphere Marathon web interface. At the top, the browser address bar displays 'infra-staging.amz.relateiq.com/services/ui/marathon'. The page title is 'marathon Healthy (32 tasks)'. Below the navigation bar, the 'MARATHON' logo and 'App' tab are visible. The main content area shows the application '/dcos-sample-webapp' in a 'Running' state. There are buttons for 'Suspend', 'Scale', 'Restart App', and 'Destroy App'. A 'Tasks' tab is selected, showing a table of task details. The 'Status' column in this table is circled in red.

ID	Status	Version	Updated
dcos-sample-webapp.a5e1489f-85cc-11e5-b11c-cae4abf417c9 ip-10-31-122-219.us-west-2.compute.internal:25259	Started	5 days ago	11/12/2015, 12:35:32 AM
dcos-sample-webapp.642dccc3-8cf4-11e5-b11c-cae4abf417c9 ip-10-31-125-194.us-west-2.compute.internal:21539	Started	a day ago	11/17/2015, 7:28:31 AM

# Mesosphere

The screenshot shows the Mesosphere Marathon web interface. At the top, a purple header displays 'marathon' and 'Healthy (32 tasks)'. Below this, the 'App' page for '/dcos-sample-webapp' is visible, with a 'Scale' button circled in red. A modal dialog box is open in the center, titled 'Scale to how many instances?', with a text input field containing the number '2' and 'Cancel' and 'OK' buttons. The background shows a table of tasks with columns for ID, Status, Version, and Updated.

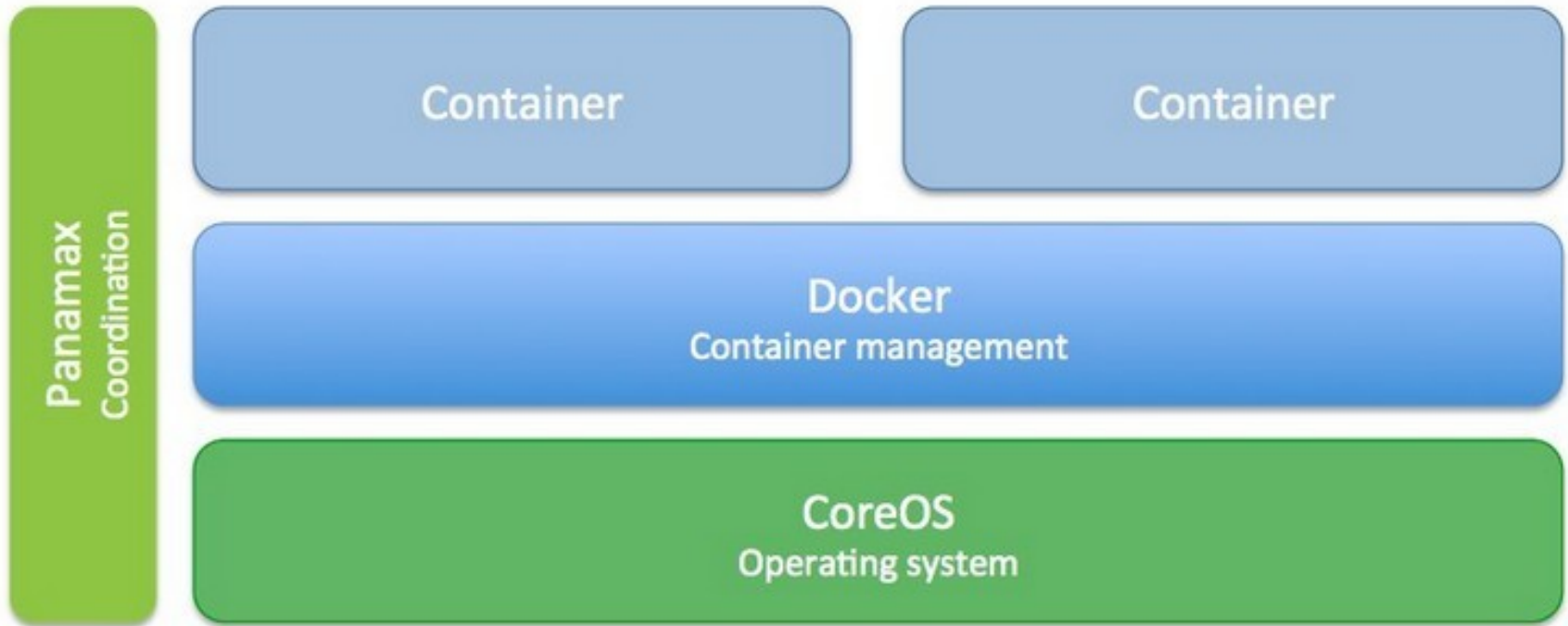
ID	Status	Version	Updated
dcos-sample-webapp_ebf1469f-8b0c-11e5-b11c-cae4ad417fd ip-10-31-122-219.us-west-2.compute.internal:25258	Started	5 days ago	11/12/2015, 12:35:32 AM
dcos-sample-webapp_8429cc3-8c44-11e5-b11c-cae4ad417fd ip-10-31-120-194.us-west-2.compute.internal:21528	Started	a day ago	11/17/2015, 7:28:31 AM

# Mesosphere

The screenshot shows the Mesosphere Marathon web interface. At the top, the browser address bar displays the URL `infra-staging.amz.relateiq.com/#!/services/ui/marathon`. The main header indicates the application is `marathon` and is `Healthy (32 tasks)`. Below this, the `App` section for `/dcos-sample-webapp` is shown, with a `Deploying` status. Action buttons for `Suspend`, `Scale`, `Restart App`, and `Destroy App` are visible. A `Tasks` tab is selected, showing a table of task instances. The table has columns for `ID`, `Status`, `Version`, and `Updated`. Three task instances are listed, with the most recent one in a `Staged` status, which is circled in red.

ID	Status	Version	Updated
<code>dcos-sample-webapp.e6e1469f-88cc-11e5-b11c-cae4abf417c9</code> <code>ip-10-31-122-219.us-west-2.compute.internal:25259</code>	Started	5 days ago	11/12/2015, 12:35:32 AM
<code>dcos-sample-webapp.642dccc3-8c44-11e5-b11c-cae4abf417c9</code> <code>ip-10-31-120-194.us-west-2.compute.internal:21339</code>	Started	a day ago	11/17/2015, 7:28:31 AM
<code>dcos-sample-webapp.0d43ae21-8d1a-11e5-b11c-cae4abf417c9</code> <code>ip-10-31-131-135.us-west-2.compute.internal:9753</code>	Staged	a few seconds ago	11/17/2015, 11:57:53 AM

# Panamax





# Panamax

**panamax** SEARCH MANAGE DOCUMENTATION

CoreOS Host: ■ CenturyLink

Manage / Dashboard / Applications /

## Socialize!

Deployed to: CoreOS Local  
[Documentation](#)  
[Access your application](#)

+ Save as Template ↻ Rebuild App ✖ Delete App

### Application Services ☰ ↻

WORKERS	API	SUPPORT
<ul style="list-style-type: none"><li>redis_latest</li><li>eti</li></ul>	<ul style="list-style-type: none"><li>postgres</li><li>api</li></ul>	<ul style="list-style-type: none"><li>erbit</li><li>mongo</li></ul>
<span>+ Add a Service</span>	<span>+ Add a Service</span>	<span>+ Add a Service</span>

UI	Add a Category
<ul style="list-style-type: none"><li>ui</li></ul>	<span>+</span>
<span>+ Add a Service</span>	

**CoreOS Journal - Application Activity Log** Show Full Activity Log

# Panamax


**פנמאקס**

CoreOS Host: ■  CenturyLink

## Search Panamax Templates & Docker Repositories

*Examples: Rails, redis, NGINX, mongoDB, you get the picture...*

### Templates



**Rails with PostgreSQL**  
Rails with PostgreSQL [More Details](#)  
*Last Updated: August 11th, 2014 18:36*

2  
Images

Did you know you can create your own custom templates to use within Panamax? [Learn more](#)

### Images

Local	rails	Tag: 4.1.1	<input type="button" value="Run Image"/>
Local	dharmamike/dc-rails	Tag: latest	<input type="button" value="Run Image"/>

# Panamax

**פנמקס** SEARCH MANAGE DOCUMENTATION

CoreOS Host: ■ CenturyLink

Manage / Dashboard / Applications / Wordpress with MySQL /

## WP

Base Image: centurylink/wordpress Tag: 3.9.1 | [View on Docker Index](#) Running  
[Documentation](#)

### Service Links

DB\_1 : DB\_1

+ Add a Linked Service

### Environment Variables

DB\_PASSWORD  
pass@word01

DB\_NAME  
wordpress

+ Add an Environment Variable

### Ports

Port Bindings	Exposed Ports
8080 : 80 /	+ Expose a Port

+ Bind a Port

### Volumes

+ Add a Volume

### Docker run command

enter run command here (optional)

Save all changes