Gestión de recursos con Apache Mesos

Francisco Guillermo García
Administración de sistemas informáticos en red

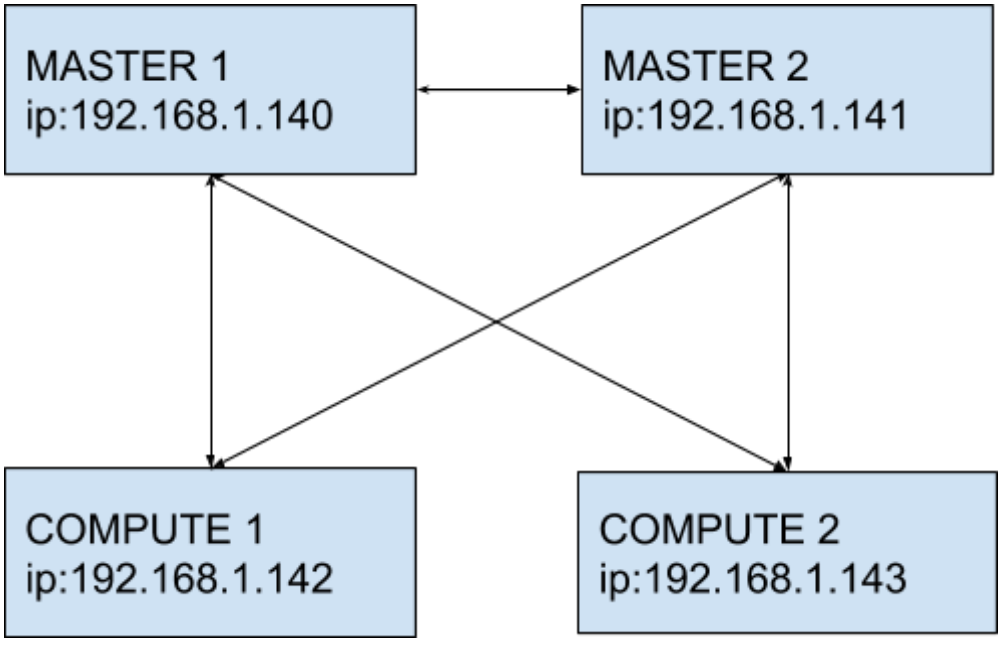
Índice

1.Introducción

El objetivo de este proyecto es realizar la instalación de un cluster de apache mesos sobre máquinas virtuales locales. probar las características que apache mesos nos puede ofrecer y las funcionalidades que nos pueden dar alguno de sus frameworks.

1.1.Escenario

El escenario donde se va a realizar el proyecto son 4 máquinas virtuales con sistema operativo tipo linux, en mi caso, Ubuntu. en la cual 2 de estas máquinas van a ejercer como nodos master y otras dos máquinas como nodos de computación.



En las máquinas master se va a instalar el master mesos junto zookeeper y los frameworks y en los nodos compute se va a instalar el agente mesos y docker que lo necesitaremos para uno de los frameworks que se va a instalar.

2.¿Qué es Apache Mesos?

Apache Mesos es un administrador de Cluster, este puede instalarse en cualquier plataforma. Apache Mesos proporciona aplicaciones para el manejo de recursos y la planificación de estos mismos.

Apache mesos obtiene los recursos de los servidores donde se van a correr nuestras aplicaciones o nuestros procesos, para que podamos gestionarlos de una manera más cómoda.

Un Cluster de Apache Mesos se divide en nodos master, nodos slaves o compute y frameworks:

Los nodos master son los que se encargan de pasar las tareas a los nodos slaves o compute. De estos nodos masters solo uno de los que tengamos va a estar ejecutándose como nodo activo ya que los demás nodos master que tengamos van a estar en forma pasiva esperando a que el nodo master que este activo falle, para asumir el rol de master activo. Esto se realiza a través de zookeeper, que es quien gestiona la unión de los nodos masters

Los nodos slaves o compute son los que se encargan de realizar las tareas de los frameworks. Estos nodos comparten su estado con el nodo master activo, para que podamos saber los recursos libres que tenemos.

Los frameworks se instalan en los nodos master, los frameworks son los que trabajan realmente con los recursos que los nodos slave o compute ponen a la disposición del cluster. Los frameworks se pueden desarrollar para crear uno propio o usar los ya existentes para realizar las acciones que necesitemos.

2.1. Características de Apache Mesos

Las características más reseñables de Apache Mesos son:

- Escalabilidad: Un cluster mesos se puede escalar de forma lineal hasta cientos de nodos
- Alta disponibilidad: Tiene tolerancia a fallos tanto de masters como de agentes.
- Contenedores: Apache Mesos soporta de forma nativa el uso de contenedores Dockers
- Web UI: Interfaz web para el usuario, la cual permite ver el estado del cluster y navegar por los contenedores
- Multiplataforma: Se puede correr en varios sistemas operativos.

3. Instalación de Apache mesos

La instalación de apache mesos se puede realizar de diferentes formas, se puede instalar mediante repositorio, mediante compilación del código o realizar una instalación sobre docker ya que existe una imagen docker de apache mesos. En este caso vamos a realizar una instalación mediante repositorios.

Lo primero que vamos a hacer es añadir el repositorio del que vamos a instalar mesos a nuestros nodos, esto lo vamos a hacer tanto en los nodos master como en los nodos slave o compute.

Añadimos la clave del servidor y dos variables que guardarán nuestro sistema y la versión:

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv E56151BF
DISTRO=$(lsb_release -is | tr '[:upper:]' '[:lower:]')
CODENAME=$(lsb_release -cs)
```

Añadimos el repositorio:

```
echo "deb http://repos.mesosphere.com/${DISTRO} ${CODENAME} main"
> /etc/apt/sources.list.d/mesosphere.list

apt-get update
```

3.1 Instalación nodos master

```
apt-get install mesos zookeeper
```

Una vez termine la instalación vamos a proceder a la configuración de mesos y zookeeper.

Lo primero que vamos a configurar va a ser la conexión con zookeeper. Para ello vamos a editar el fichero **/etc/mesos/zk**

```
zk://192.168.1.140:2181,192.168.1.141:2181/mesos
```

En este fichero debemos poner todas las ips de nuestros nodos master. Si tenemos 9 nodos master por ejemplo deberíamos poner las ip de los 9 nodos master.

Configuramos el ID de los servidores master, esta configuración se realiza en el fichero **/etc/zookeeper/conf/myid**

Configuración Master 1:

```
1
```

Configuración Master 2:

```
2
```

Esta configuración se realiza para que cada nodo master tenga un ID único, de esta manera nos aseguramos que se puede resolver correctamente cada host mediante el sistema de IDs que se usa. Este sistema se configura en el fichero **/etc/zookeeper/conf/zoo.cfg** de cada nodo master

```
server.1=192.168.1.140:2888:3888
server.2=192.168.1.141:2888:3888
```

El primer puesto es el puerto usado para conectar con el nodo master que funcionará como líder y el segundo puesto es el puesto que se usa para la selección del líder.

Una vez configurado zookeeper procedemos a configurar mesos.

Lo primero que debemos configurar es el quorum de los nodos master, esto nos sirve para saber el número de hosts que necesitamos para que el cluster sea funcional.

El quorum determina el número de fallos que podemos tener dentro de nuestro cluster en este caso como tenemos 2 nodos master el quorum debe ser establecido a 1 ya que solo podremos tener el fallo de uno de los dos nodos.

El quorum lo configuramos en el fichero **/etc/mesos-master/quorum**

```
1
```

Configuramos las ips de los servidores.

En los fichero **/etc/mesos-master/ip y /etc/mesos-master/hostname:**

Master 1:

```
192.168.1.140
```

Master 2:

```
192.168.1.141
```

Esto se pone en los dos ficheros antes mencionados de cada nodo master

Para darle un nombre al cluster y aparezca en el panel web, solo tenemos que crear el siguiente fichero: **/etc/mesos-master/cluster** y en ese fichero poner el nombre que queremos que aparezca

```
MesosPaco
```

Desactivamos el servicio de mesos agent

```
systemctl stop mesos-slave.service
systemctl disable mesos-slave.service
```

Habilitamos los servicios zookeeper y mesos-master.

```
systemctl enable mesos-master.service
systemctl enable zookeeper.service
```

Iniciamos los servicios

```
systemctl start zookeeper.service
systemctl start mesos-master.service
```

3.2.Instalación nodos slave o compute

```
apt-get install mesos
```

Paramos los servicios zookeeper y mesos-master ya que en estos nodos solo necesitamos el servicio mesos-slave

```
systemctl stop zookeeper.service
systemctl stop mesos-master.service
systemctl disable zookeeper.service
systemctl disable mesos-master.service
```

Configuramos las ips de los nodos slave o compute en el fichero **/etc/mesos-slave/ip y /etc/mesos-slave/hostname**

Compute 1:

```
192.168.1.142
```

Compute 2:

```
192.168.1.143
```

Ahora configuramos para que identifique como nodos master los que hemos configurado anteriormente, para ello vamos al siguiente fichero: **/etc/mesos/zk**

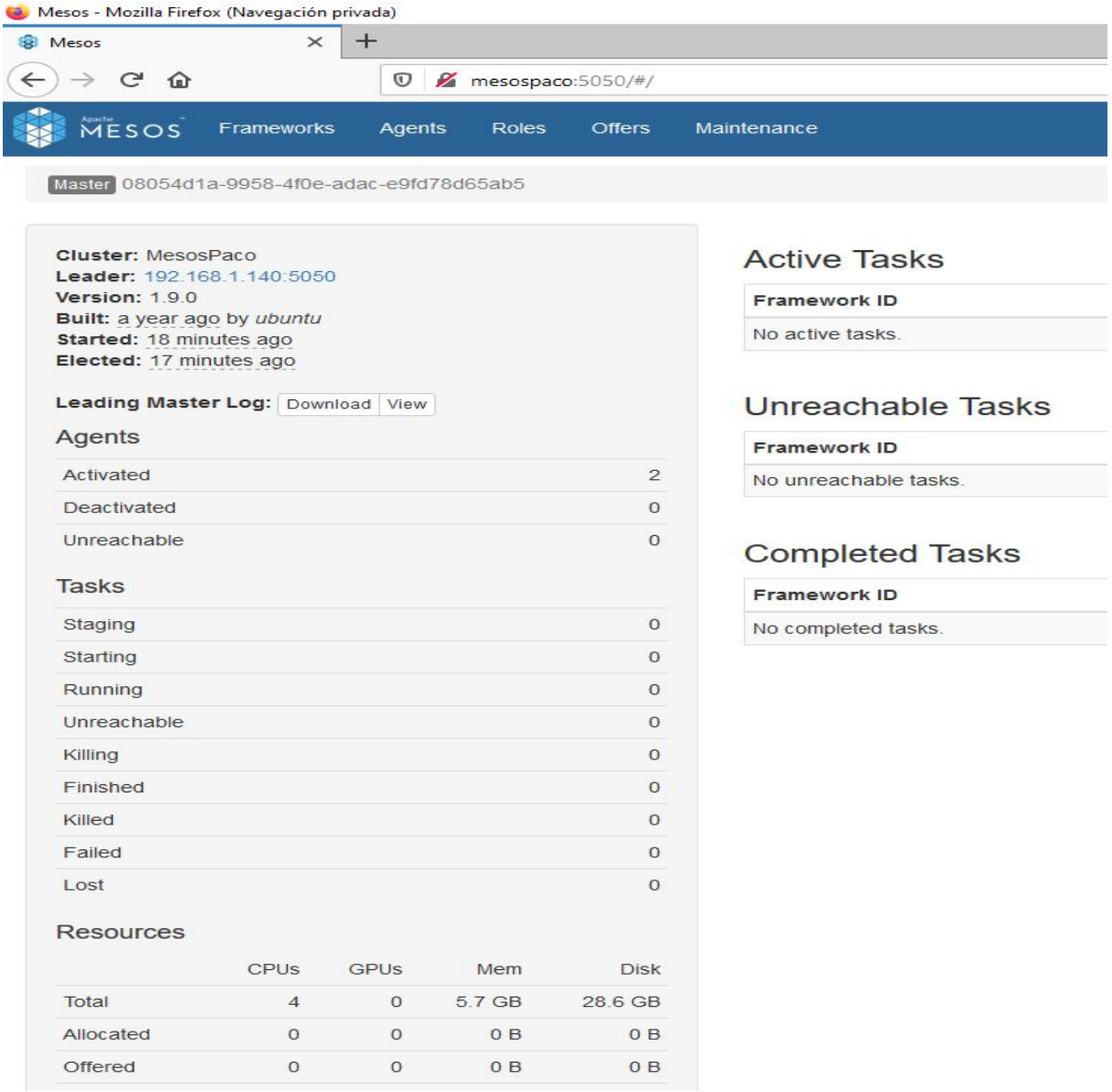
```
zk://192.168.1.140:2181,192.168.1.141:2181/mesos
```

Reiniciamos el servicio en los dos nodos slave o compute

```
systemctl restart mesos-slave
```

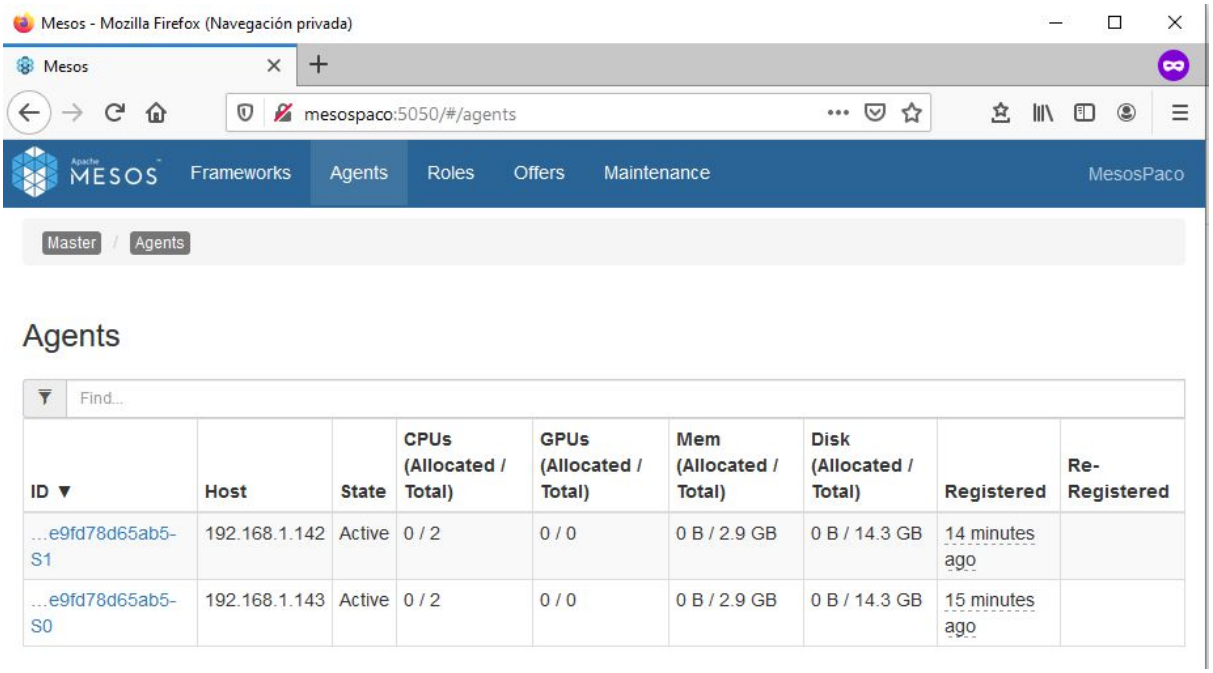
Ya tendríamos configurado nuestro cluster mesos, para acceder a su panel web tendríamos que acceder a la ip del nodo master en ese momento se está ejecutando como líder, en mi caso yo he configurado mi fichero **/etc/hosts** para tener una resolución de nombres.

El acceso sería desde un navegador a la url: **http://ipmaster:5050**



Como vemos nos dice que tenemos dos agentes activados y que tenemos 4 cpus, 5.7GB de memoria y 28.6GB de disco disponible para usar en nuestro frameworks instalados en nuestro cluster.

Si accedemos al apartado Agents, podemos ver cuales son los agentes que tenemos en el cluster y cuantos recursos tienen cada uno disponibles y cuantos de estos recursos están en uso.



Mesos les pone un identificador a los agente pero podemos ver por la dirección ip de cado uno, cual es cada uno.

4. Frameworks

4.1. Marathon

Marathon es un framework que permite lanzar aplicaciones que van a estar ejecutándose durante un largo periodo de tiempo, como puede ser una página web, una aplicación python. Este framework te permite lanzar tus aplicaciones en contenedores docker, en este proyecto vamos a desplegar una aplicación con Marathon, el cual está corriendo junto Apache Mesos, en contenedores docker.

4.2. Chronos

Chronos es un planificador de tareas con tolerancia a fallos, para explicarlo de una manera más amena, es como “cron” de los sistemas linux, este framework nos permite programar tareas cada cierto tiempo, las cuales si fallan, volverán a intentar realizarse sin fallo.

5. Instalación de frameworks

5.1. Instalación Marathon

Para la instalación de Marathon la vamos a realizar sobre los nodos master del cluster. Pero cómo se van a ejecutar las aplicaciones que lancemos en los nodos slave o compute, también tenemos que realizar acciones en los nodos slave o compute.

En los nodos master:

```
apt-get install marathon
```

Creamos el directorio de configuración de marathon

```
mkdir -p /etc/marathon/conf
```

Creamos los siguiente ficheros dentro de ese directorio: **/etc/marathon/conf/hostname, /etc/marathon/conf/master y /etc/marathon/zk**

Fichero /etc/marathon/conf/hostname de Master 1

```
192.168.1.140
```

Fichero /etc/marathon/conf/hostname de Master 2

```
192.168.1.141
```

Los otros dos fichero son iguales para ambos nodos master

Fichero /etc/marathon/conf/master

```
zk://192.168.1.140:2181,192.168.1.141:2181/mesos
```

Fichero /etc/marathon/conf/zk

```
zk://192.168.1.140:2181,192.168.1.141:2181/marathon
```

Ahora vamos al fichero **/etc/default/marathon**

```
MARATHON_MASTER=zk://192.168.1.140:2181,192.168.1.141:2181/mesos
MARATHON_ZK=zk://192.168.1.140:2181,192.168.1.141:2181/marathon
```

Reiniciamos el servicio en ambos nodos

```
systemctl restart marathon.service
```

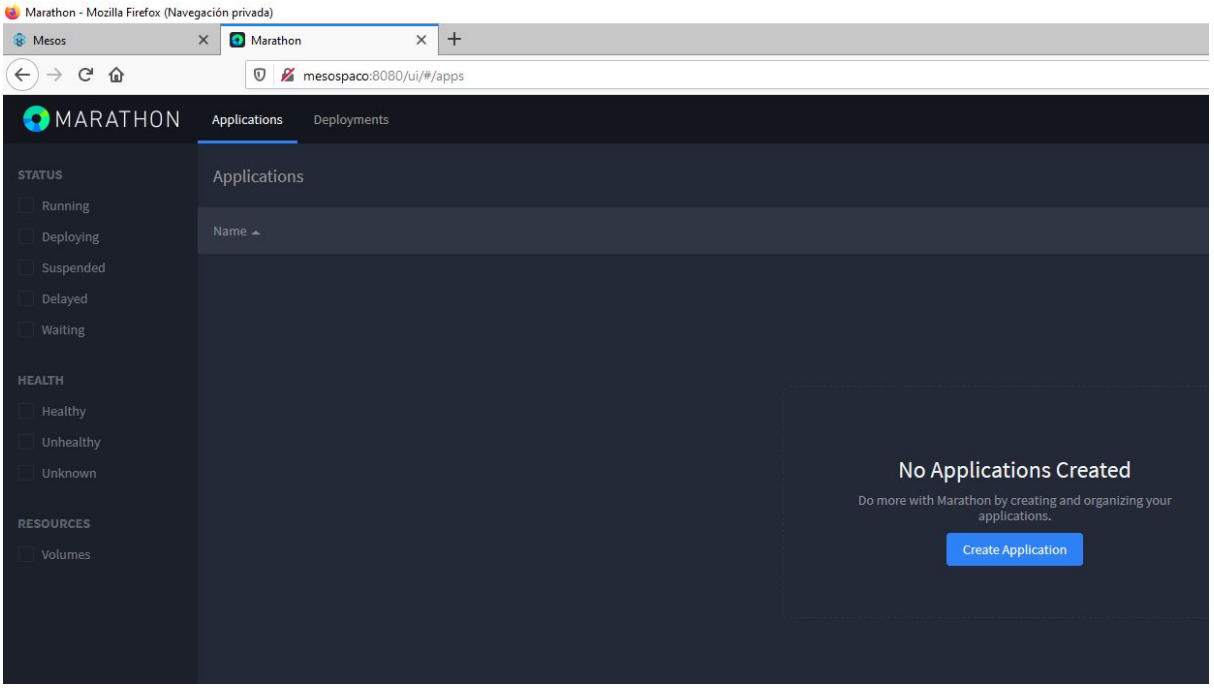
En los nodos slave o compute tenemos que instalar docker para poder lanzar nuestras aplicaciones sobre contenedores docker

```
apt-get install docker.io
```

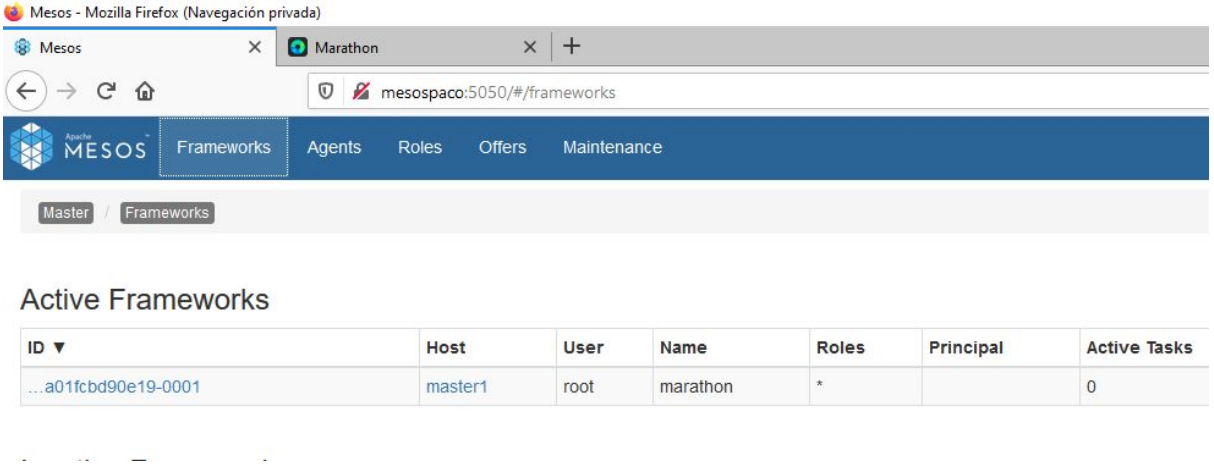
También tenemos que configurar el servicio mesos-slave para el uso de contenedores docker. Para ello vamos a crear realizar lo siguiente:

```
echo 'docker,mesos' > /etc/mesos-slave/containerizers
echo '5mins' > /etc/mesos-slave/executor_registration_timeout
```

Marathon cuenta con una interfaz web a través de la cual podemos lanzar nuestras aplicaciones, para acceder a esta interfaz tenemos que acceder a http://ipmaster:8080 , en mi caso yo he configurado el fichero **/etc/hosts** para acceder mediante resolución estática.



De esta forma ya hemos accedido a nuestro framework, para comprobar que está correctamente corriendo junto con nuestro apache mesos, vamos al panel web de apache mesos y en la pestaña frameworks debería de salirnos el framework.

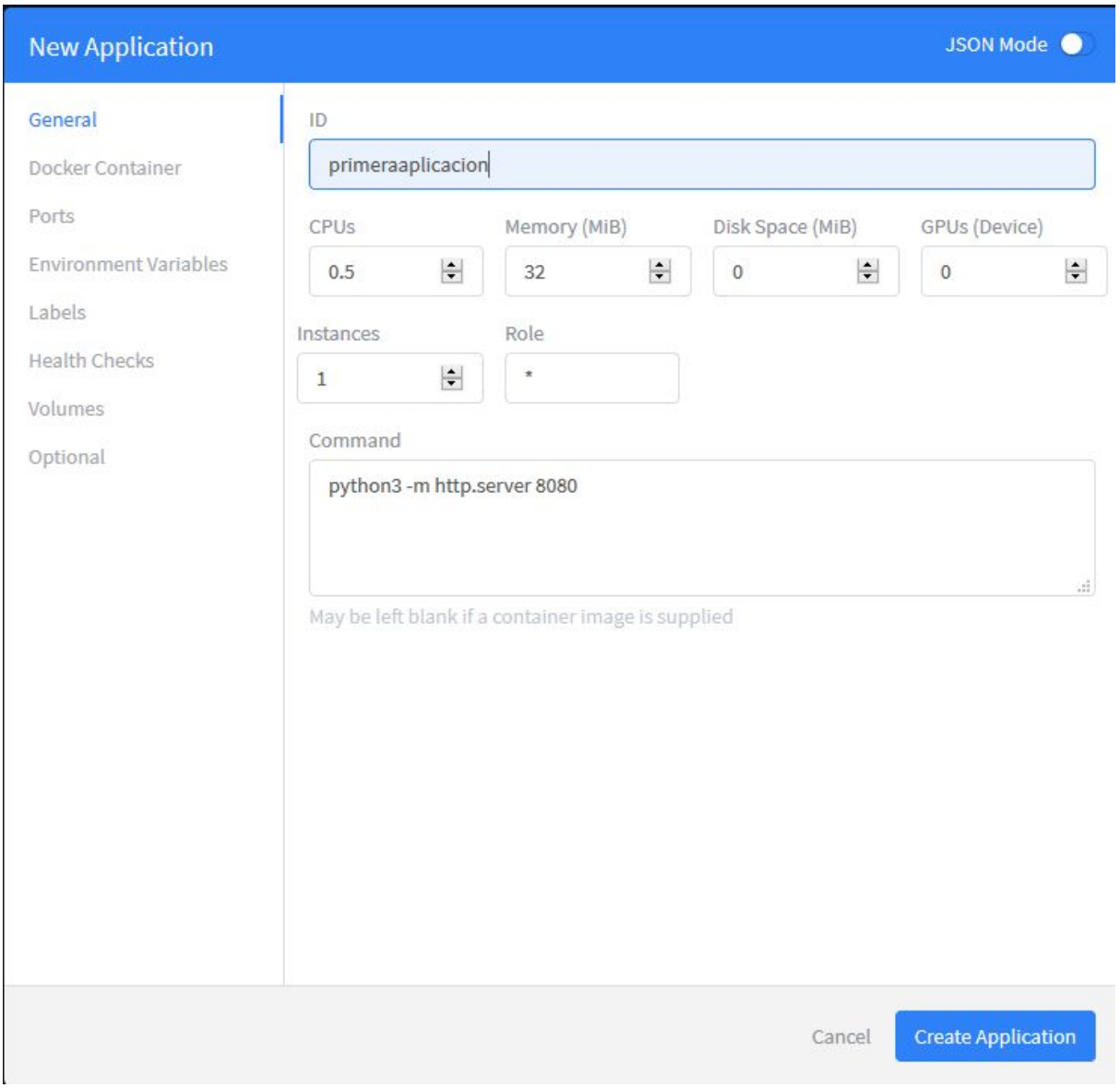


Como vemos en la imagen el framework está corriendo en nuestro cluster de apache mesos.

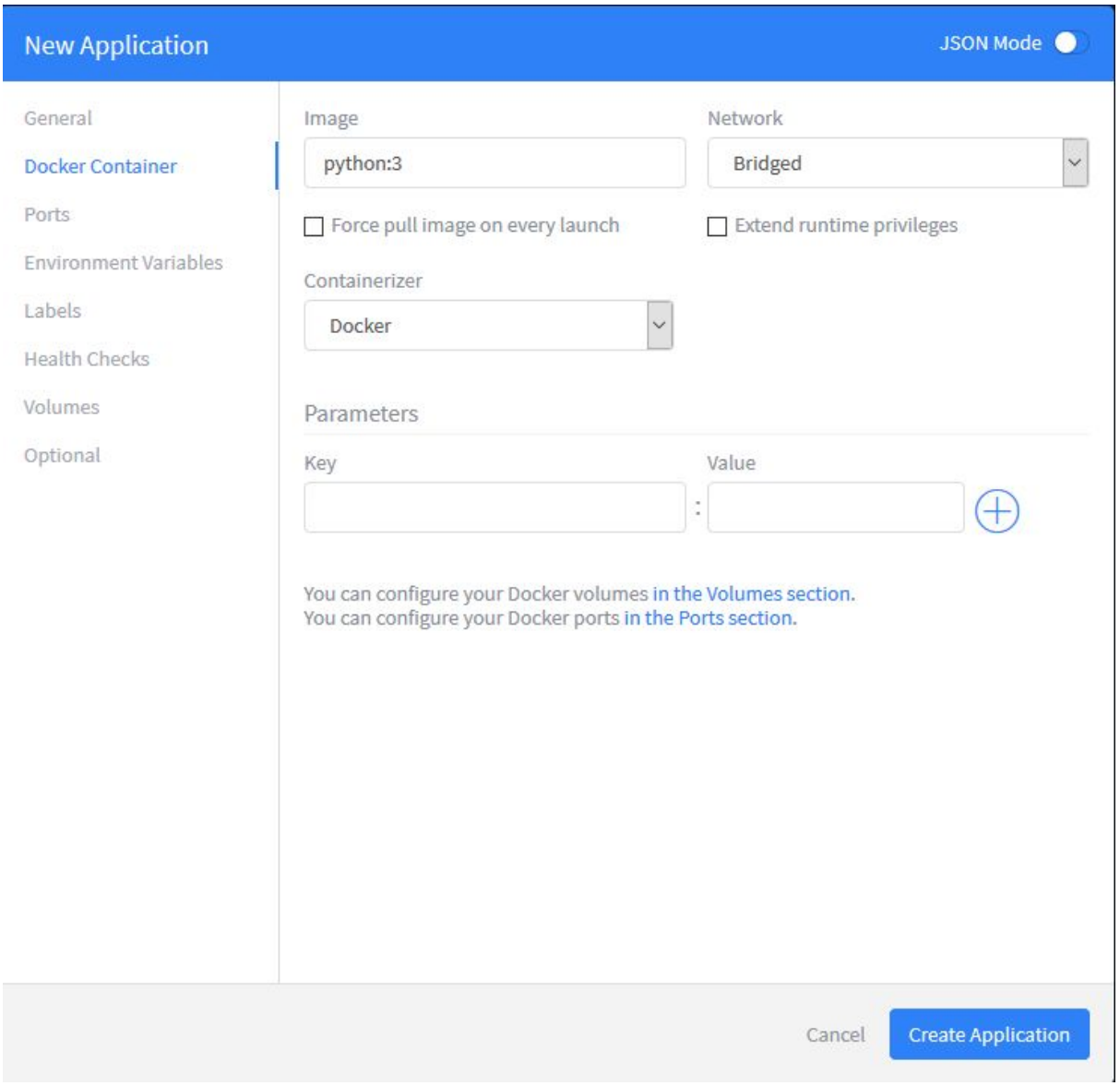
Ahora vamos a desplegar un par de aplicaciones para comprobar que se pueden desplegar correctamente con docker

Primero vamos a desplegar una aplicación muy sencilla de python en el que lo único que nos va a mostrar son los directorios que hay en el contenedor que se va a crear. Para ello, nos vamos a ir a nuestro panel web de marathon y vamos a hacer click en “Create Application”, nos saldrá una ventana emergente la cual debemos rellenar los campos.

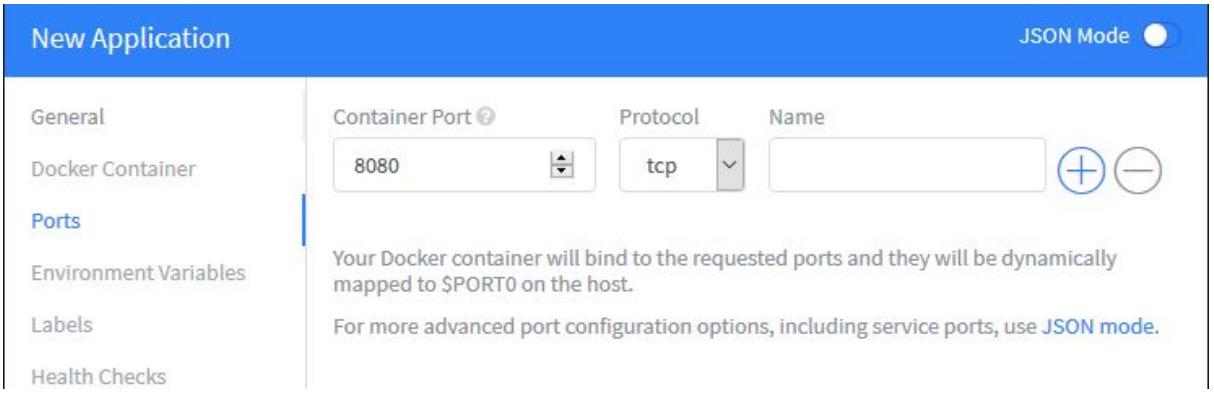
En la pestaña general vamos a decirle el nombre que va a tener la aplicación, los recursos que se le van a dar y que comando se va a ejecutar en el contenedor.



En la pestaña docker container vamos a que imagen de docker va usar y el tipo de red que queremos.

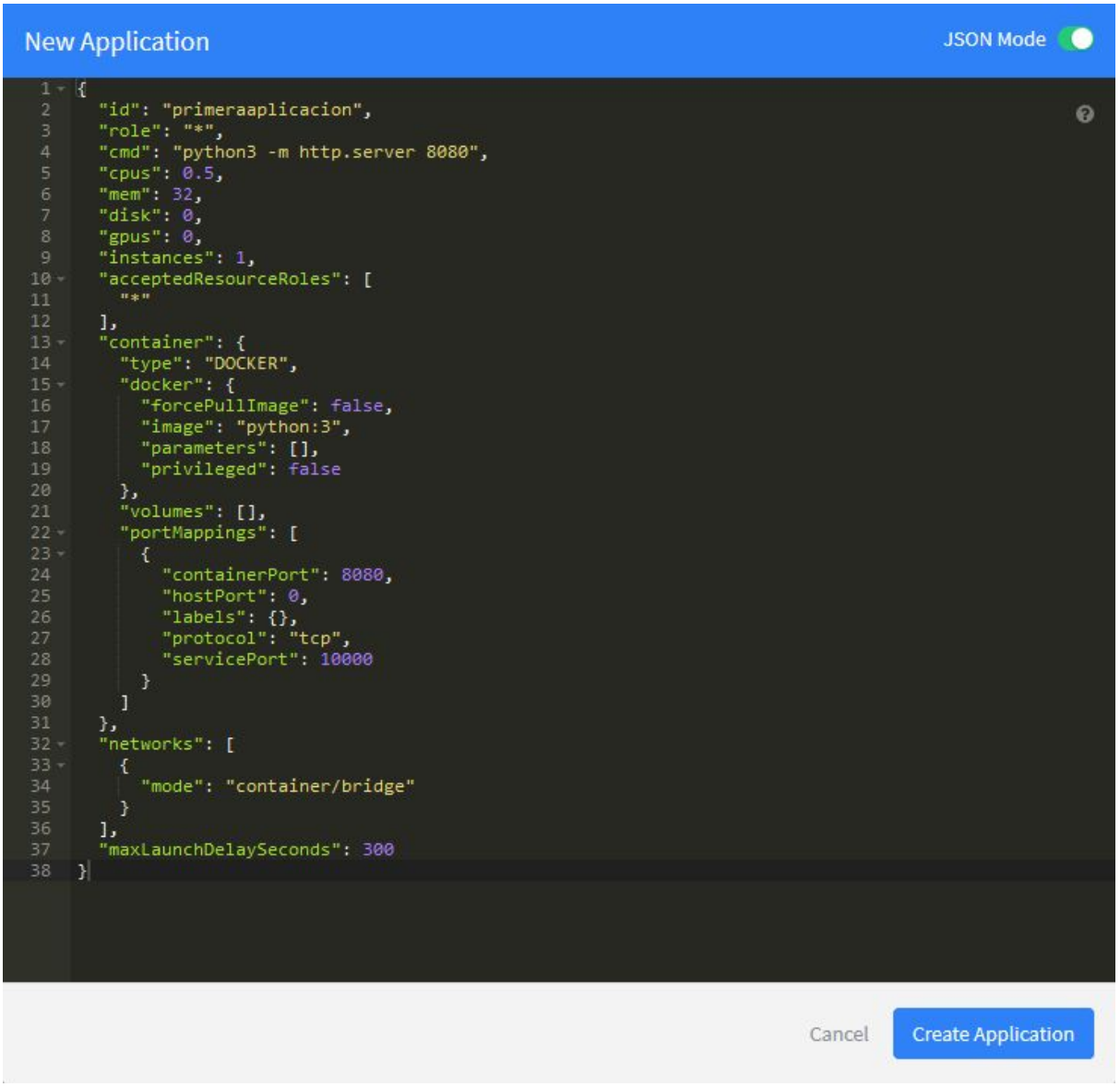


En la pestaña Ports vamos a decirle el puesto que va a redirigir, como en la pestaña general le hemos puesto en el comando que va a ejecutar el puerto 8080, tenemos que redirigir dicho puerto.



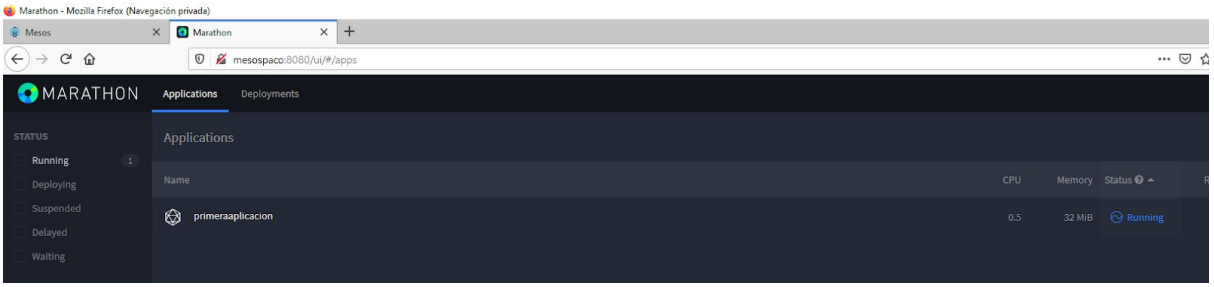
En las demás pestañas no es necesario cambiar nada, lo dejamos por defecto.

Si queremos en la esquina superior derecha de esta ventana emergente podemos ver el json que se crea al poner estos parámetros.



Una vez terminado vamos a darle a create application.

Se nos creará una aplicación en marathon:

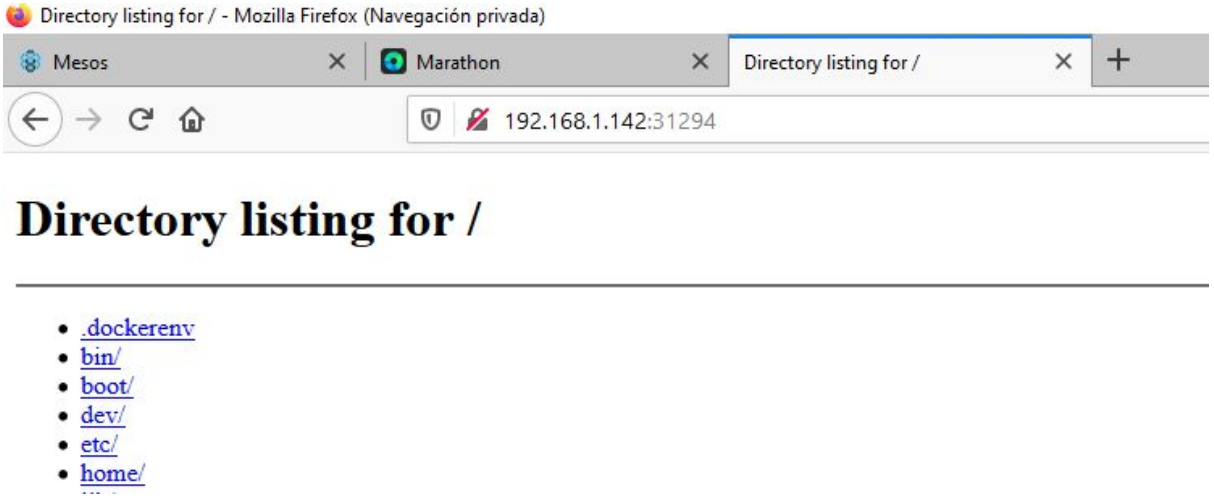


Nos vamos al panel web de apache mesos y vamos si la tarea se esta ejecutando.

Como vemos en la imagen el contenedor se ha creado en el nodo slave o compute que tiene por ip 192.168.1.142, si nos vamos a esta máquina y vemos los contenedores docker que se están ejecutando:

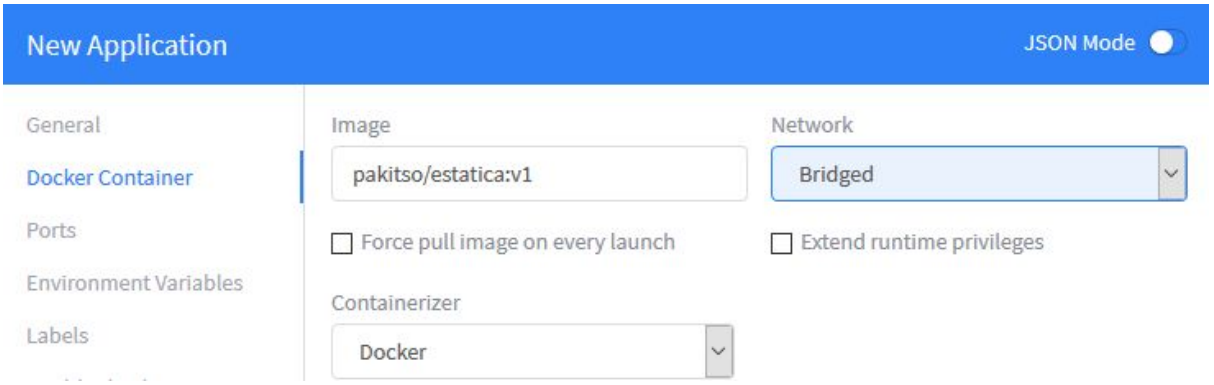
```
root@compute1:~# docker ps
CONTAINER ID    IMAGE              COMMAND                  CREATED
        STATUS              PORTS                NAMES
a2f043bd5231    python:3           "/bin/sh -c 'python3…"   5 minutes ago
        Up 5 minutes        0.0.0.0:31294->8080/tcp
mesos-0084d61a-5847-437e-80d9-ba5b4aba8c83
```

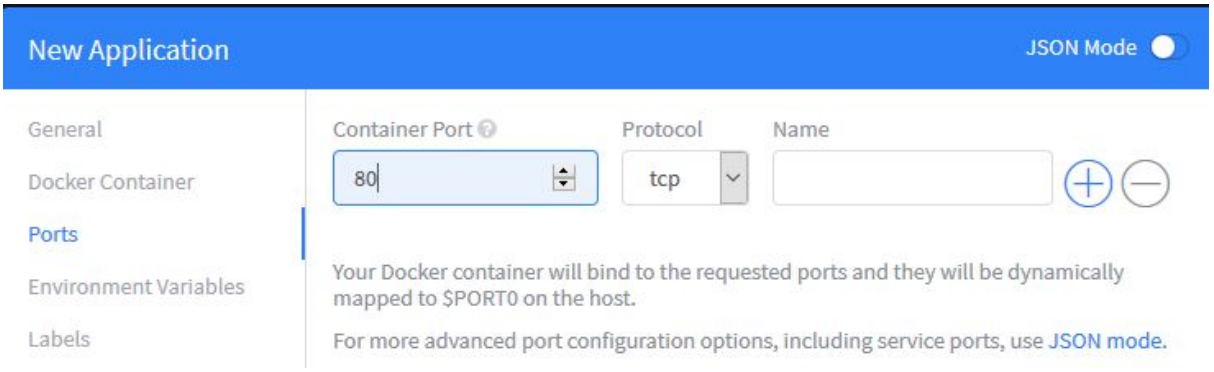
Vemos que se está ejecutando un contenedor docker. Si accedemos a la ip del nodo slave o compute que esta ejecutando este contenedor docker y al puerto de esa máquina que se ha redireccionado al puerto 8080 del contenedor:



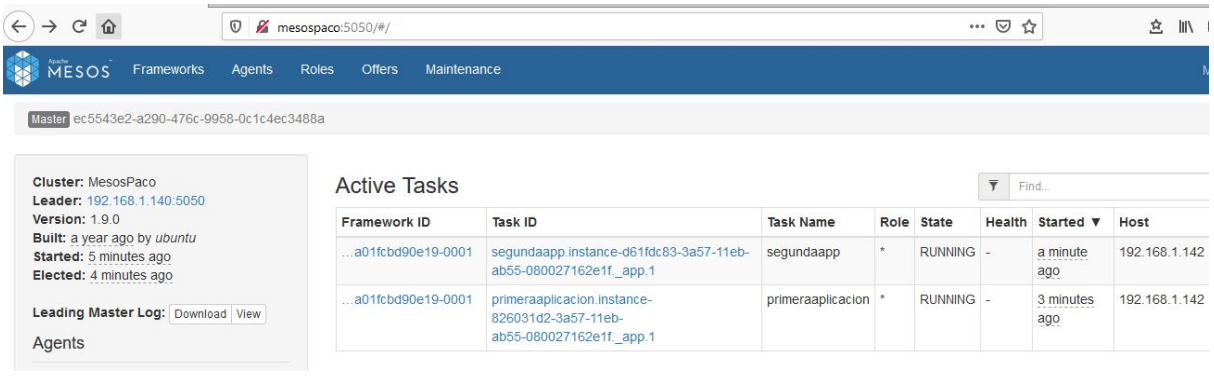
Ahora vamos a desplegar una página web estática que tengo en una imagen docker que realizamos durante el curso.

Para ello lo único que vamos a cambiar de la anterior es el puerto que se va a redireccionar y la imagen.





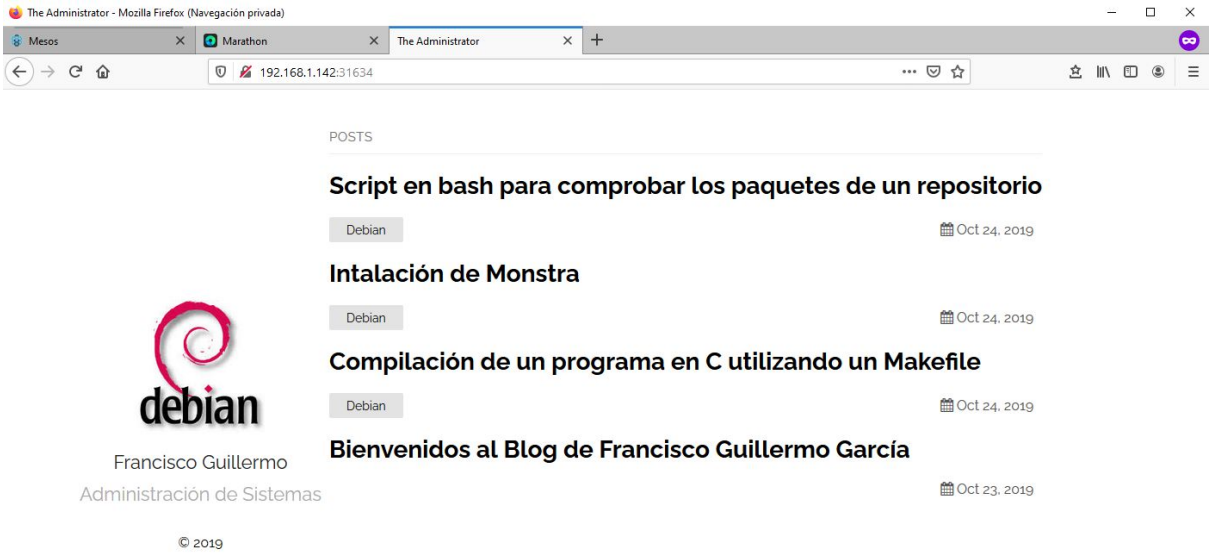
Esperamos a que se cree la aplicación y vamos al panel de mesos y comprobamos que se está corriendo la aplicación.



Como vemos se está corriendo en el nodo slave o compute que tiene como ip 192.168.1.142, por lo que vamos a comprobar los contenedores docker que se están ejecutando

```
root@compute1:~# docker ps
CONTAINER ID    IMAGE                COMMAND                  CREATED
        STATUS              PORTS               NAMES
123353cd662d      pakitso/estatica:v1   "/usr/sbin/apache2ct…"   9 seconds
ago    Up 8 seconds        0.0.0.0:31634->80/tcp
mesos-ef6935fb-3d46-41a2-be1e-e281c3a9c402
```

Si accedemos a la ip de la máquina que tiene el docker junto con el puesto que se ha redireccionado podremos ver la aplicación o en este caso página estática.

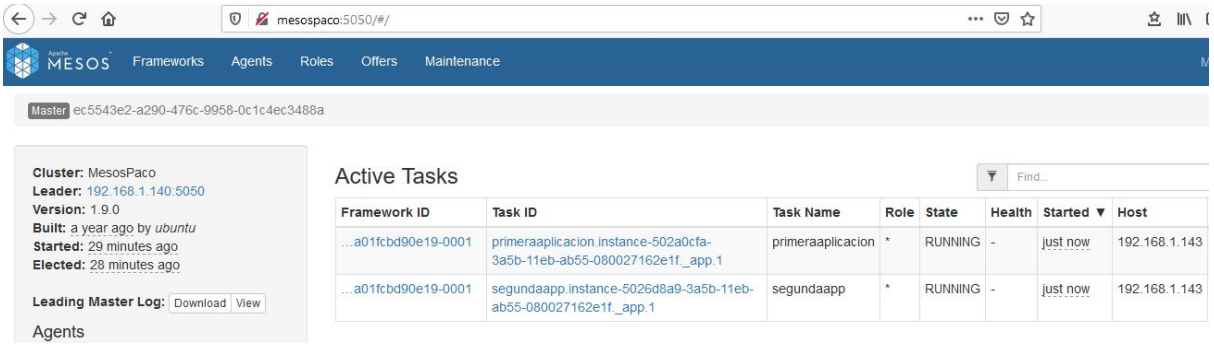


Ahora vamos a probar la prueba contra fallos de los nodos slave o compute, es decir, vamos a parar el servicio del nodo que tiene como ip 192.168.1.142 y vamos a esperar 2 minutos que es el tiempo que le he configurado para que si no responde el nodo slave o compute se creen los contenedores con las aplicaciones en el otro nodo.

Para ello paramos el servicio en el nodo compute1

```
root@compute1:~# systemctl stop mesos-slave
```

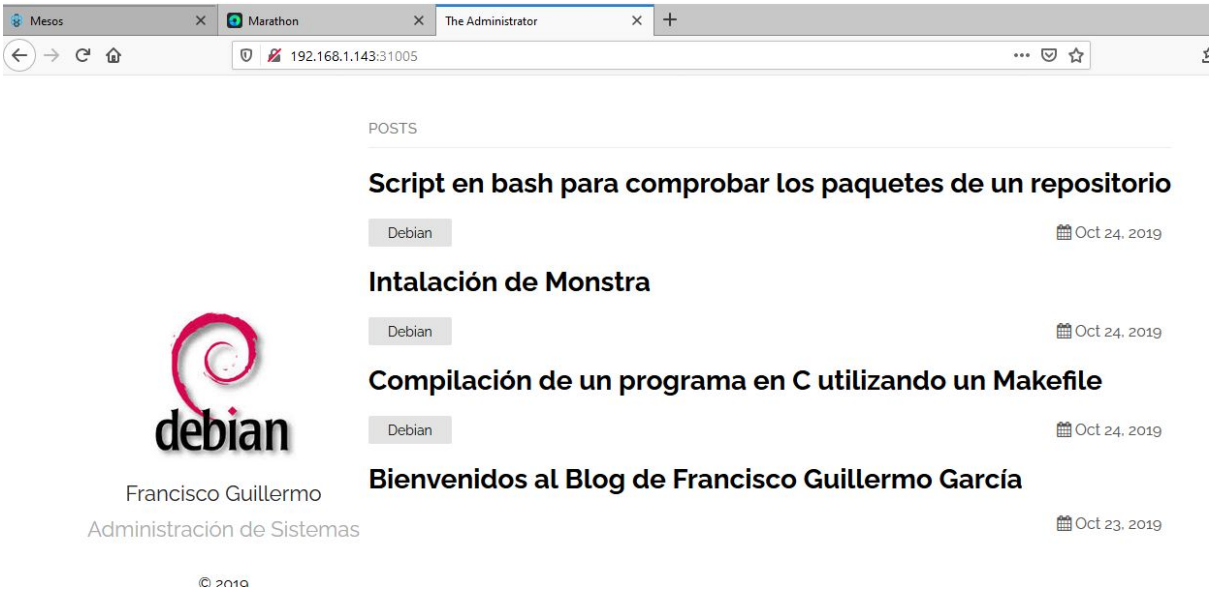
Ahora vamos a esperar dos minutos y se empezaran a crear en el otro nodo, por lo que en el panel web de mesos, podemos ver que ahora la tarea se está ejecutando en el nodo 192.168.1.143



Vemos los contenedores que se están ejecutando en ese nodo

```
root@compute2:~# docker ps
CONTAINER ID    IMAGE                COMMAND                  CREATED
        STATUS              PORTS               NAMES
94ec0fb76789        python:3             "/bin/sh -c 'python3…"   About a
minute ago   Up About a minute   0.0.0.0:31388->8080/tcp
mesos-5e565962-e716-4a01-ac05-e044f67e5da6
169d2edd4180        pakitso/estatica:v1   "/usr/sbin/apache2ct…"   About a
minute ago   Up About a minute   0.0.0.0:31005->80/tcp
mesos-33df09c3-2237-4386-84cc-0ab5e8b821d7
```

Ahora para acceder a nuestra aplicación tenemos que acceder a la ip del nuevo nodo compute donde se están ejecutando y el nuevo puerto que docker ha redireccionado



5.2. Instalación Chronos

La instalación de Chronos la vamos a realizar en los nodos master de nuestro cluster, en chronos vamos a ejecutar tareas que queramos que se ejecuten cada cierto tiempo, estas tareas se van a ejecutar en los nodos slave.

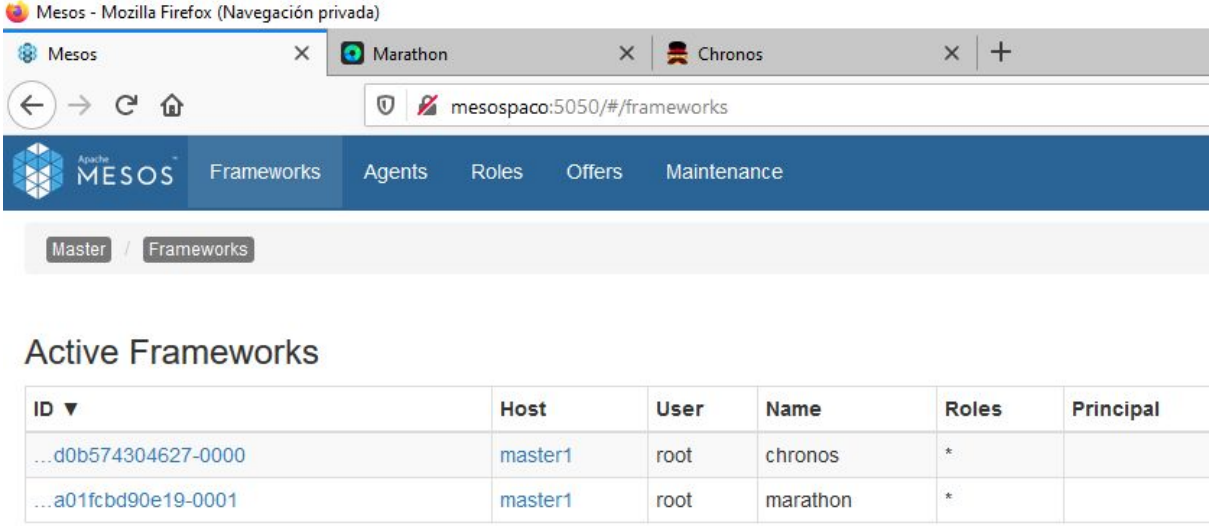
Para instalar Chronos nos vamos a nuestros nodos slave o compute y vamos a ejecutar el siguiente comando

```
apt-get install chronos
```

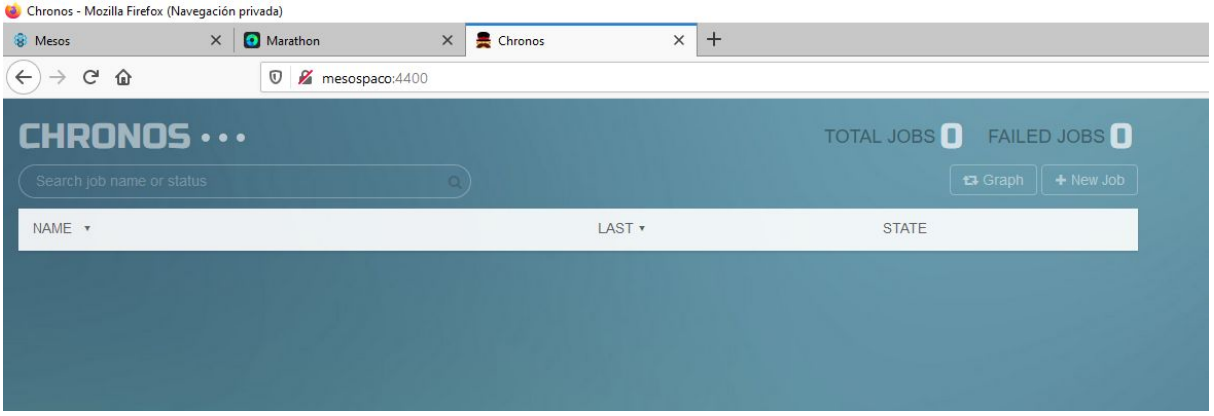
Al ser la instalación mediante paquetería y tener un cluster mesos ya corriendo sobre las máquinas, Chronos directamente se configura para funcionar sobre este cluster que tenemos ejecutándose en nuestras máquinas y lo único que debemos hacer es iniciar el servicio de Chronos.

```
systemctl start chronos.service
```

Una vez iniciado el servicio nos vamos a ir a nuestro panel web de mesos y vamos al apartado “Frameworks” para ver si el framework se esta ejecutando en conjunto nuestro cluster de apache mesos.

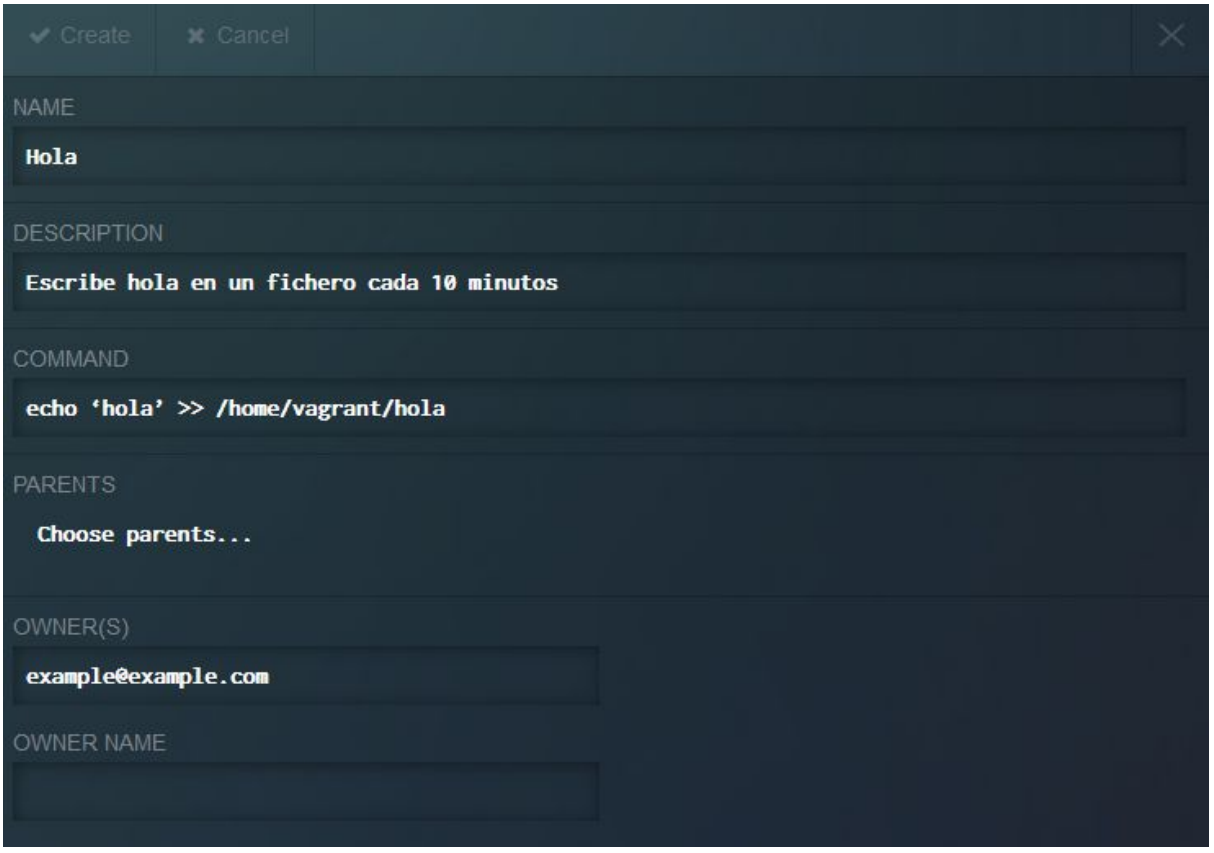


Como vemos en la imagen, Mesos ha detectado Chronos como framework del cluster, por lo que ya podemos usarlo, Chronos cuenta con una interfaz web en la que podemos crear las tareas que queremos que se ejecuten. Para acceder a este panel tenemos que acceder a la ip del master que se encuentra ejecutándose como líder al puesto 4400, este puerto se puede configurar en el fichero **/etc/chronos/conf/http_port** , en mi caso como yo he configurado el fichero hosts para tener una resolución estática, voy a acceder a la url http://mesospaco:4400

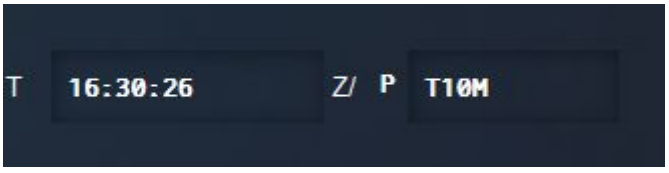


Ahora ya podemos crear nuestras tareas que queremos que se ejecuten periódicamente cada cierto tiempo. Yo voy a ejecutar una tarea muy sencilla como puede ser que una vez cada 10 minutos se ejecute el comando “echo ‘hola’ >> /home/vagrant/hola” en ambos nodos slave.

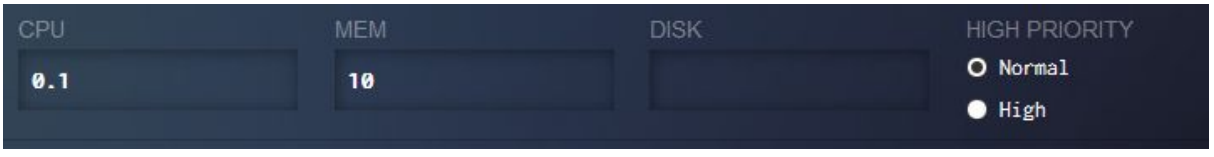
En el panel web de Chronos vamos a hacer click en “New Job”, se nos mostrará una ventana emergente donde podemos configurar nuestro nueva tarea.



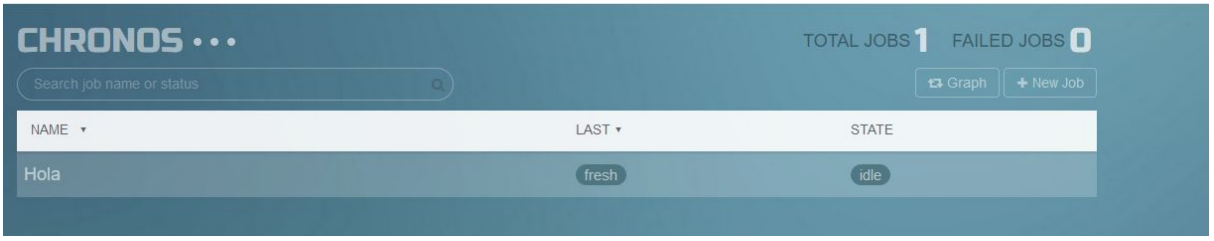
Abajo de esta ventana emergente tenemos un apartado “other settings” en el que podemos configurar a que hora queremos que se inicie la tarea y cada cuanto tiempo, en mi caso se va a ejecutar cada 10 minutos.



También vamos a seleccionar los recursos que le queremos dar a la tarea.



Hacemos click en crear y en el panel web de chronos nos aparecerá nuestra nueva tarea.



Esperamos a que sea la hora que hemos seleccionado para que se ejecute la tarea y comprobamos en el panel web de mesos que se ha ejecutado la tarea.

Vemos que la tarea se ha ejecutado en el nodo 192.168.1.142, por lo que si nos vamos a ese nodo y miramos que hay dentro del directorio /home/vagrant vemos que se ha creado el fichero “hola” y el contenido es “hola”

```
root@compute1:~# ls /home/vagrant/
hola
root@compute1:~# cat /home/vagrant/hola
hola
```

Esperamos 10 minutos a que se vuelva a ejecutar la tarea y volvemos al panel web de mesos la tarea ejecutada.

Esta vez se a ejecutado en el nodo con ip 192.168.1.143. si nos vamos a ese nodo y comprobamos que se ha creado el fichero

```
root@compute2:~# ls /home/vagrant/
hola
root@compute2:~# cat /home/vagrant/hola
hola
```

6. Conclusión

En conclusión, como hemos visto en el desarrollo de esta memoria, apache mesos, abstrae los recursos del cluster para ofrecerles a los frameworks los recursos que necesitan para la ejecución de las tareas que van a realizar estos frameworks, ya sean tareas diarias con el framework chronos, como el despliegue de aplicaciones con el framework Marathon.

En mi opinión, es una herramienta que una vez consigues el conocimiento para usarla, tiene bastante potencia pero al no tener un gran soporte de una comunidad como puede ser el soporte que tiene kubernetes, lo hace una herramienta el cual menos gente usa para la realización de estas tareas.

7. Trabajos futuros

Después del desarrollo de este proyecto, se pueden llevar a acabo acciones para seguir el desarrollo de este proyecto, estas acciones podrían ser la instalación y aprendizaje de utilización de otros frameworks que tiene apache mesos o la implementación de aplicaciones web más grandes con el framework Marathon, como pueden ser aplicaciones que necesiten una base de datos ejecutándose en conjunto para que la aplicación pueda tener un funcionamiento correcto.

8. Referencias

http://mesos.apache.org/

https://mesosphere.github.io/marathon/

https://mesos.github.io/chronos/