



# PLATAFORMAS DE GESTIÓN DE CONTENEDORES

Despliegue de API REST en Clúster Docker Swarm

Universidad Politécnica de Valencia  
Mater Universitario en computación paralela y distribuida

DORIAN ADOLFO ORDOÑEZ OSORTO  
dooros@posgrado.upv.es

## Contenido

Introducción .....	2
Esquema General .....	3
Objetivos .....	3
Desarrollo .....	4
Requisitos previos .....	4
Creación de contenedor personalizado .....	4
Script de configuración .....	6
Ejecución .....	9
Creación de clúster.....	9
Pruebas de Funcionamiento. ....	11
Conclusiones .....	13
Referencias.....	14

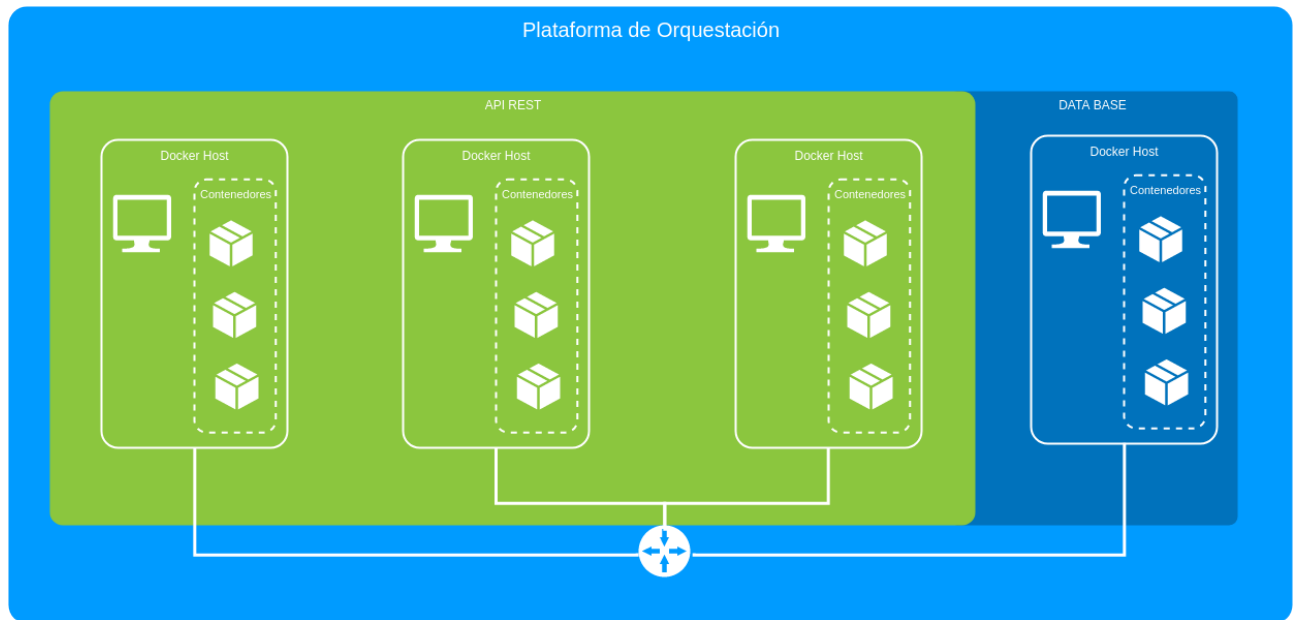
## Introducción

El uso de tecnologías orientadas a microservicios cada vez está siendo más usada y tecnologías como lo es Docker, LXC Linux, Kubernetes, Apache Mesos y entre otros aportan cada vez más facilidad al manejo de infraestructuras orientadas a los microservicios. En este documento se presenta una forma de realizar el despliegue de un Clúster con Docker Swarm mediante el uso de Scripts de Linux, de esta forma crearemos nuestro Clúster y lanzaremos sobre él una serie de servicios que nos darán acceso a una aplicación de tipo API REST con un servicio de base datos en el que se agrega un volumen para que nuestros datos tengan persistencia. También haremos uso de la plataforma Docker Hub para crear nuestro contenedor personalizado con el software de desarrollo necesario para que nuestra aplicación se ejecute.

## Esquema General

### Objetivos

- Creación de Contenedor con NodeJS y MongoDB en Docker Hub
- Creación de Docker Compose para levantar los servicios.
- Uso de volumen para persistencia de datos de base datos.
- Uso de volumen para montar nuestra aplicación.
- Uso de plataforma de orquestación Swarm.
- Analizar usabilidad, facilidad, versatilidad.



Nuestra infraestructura consta de un Clúster de Docker Swarm en el que tendremos nuestros servicios de base de datos y de aplicación corriendo. Para este ejemplo aremos uso del software de virtualización VirtualBox para crear cada uno de los nodos.

## Desarrollo

### Requisitos previos

Para realizar de forma automatizada el despliegue de nuestra API REST haremos tendremos como sistema operativo una distribución de Linux, en este caso se realizó con la distribución Linux Mint y previamente instalado VirtualBox

### Creación de contenedor personalizado

Creación de contenedor personalizado en Docker Hub

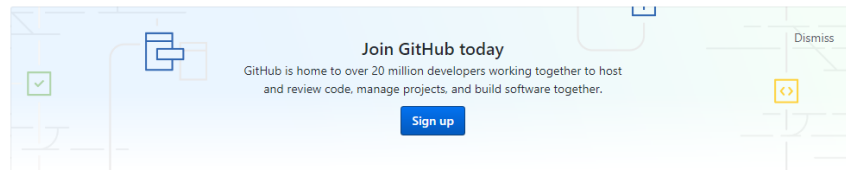
#### ***Dockerfile***

```
FROM ubuntu:14.04
MAINTAINER dooros@posgrado.upv.es
VOLUME ["/data/db"]
# install SSH
RUN apt-get update && apt-get -y install openssh-server supervisor
RUN mkdir /var/run/sshd
RUN echo 'root:root' |chpasswd
RUN sed -ri 's/^PermitRootLogin\s+.*\/PermitRootLogin yes/'
/etc/ssh/sshd_config
RUN sed -ri 's/UsePAM yes/#UsePAM yes/g' /etc/ssh/sshd_config
# Install MongoDB
RUN \
    sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
EA312927 && \
    sudo echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-
org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-
3.2.list && \
    sudo apt-get update && \
    sudo apt-get install -y mongodb-org
RUN apt-get update
# Install Node.js
RUN apt-get install --yes curl
RUN curl --silent --location https://deb.nodesource.com/setup_8.x | sudo
-E bash -
RUN apt-get install --yes nodejs
COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
# Open the ports
EXPOSE 22 3000 27017
# run services
CMD ["/usr/bin/supervisord"]
```

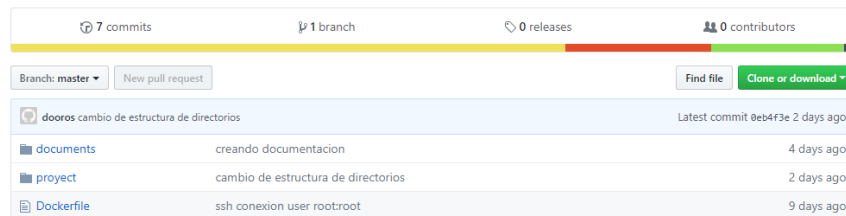
En anterior archivo define:

1. La configuración de un servidor SSH
2. Instalación de MongoDB
3. Instalación de NodeJs

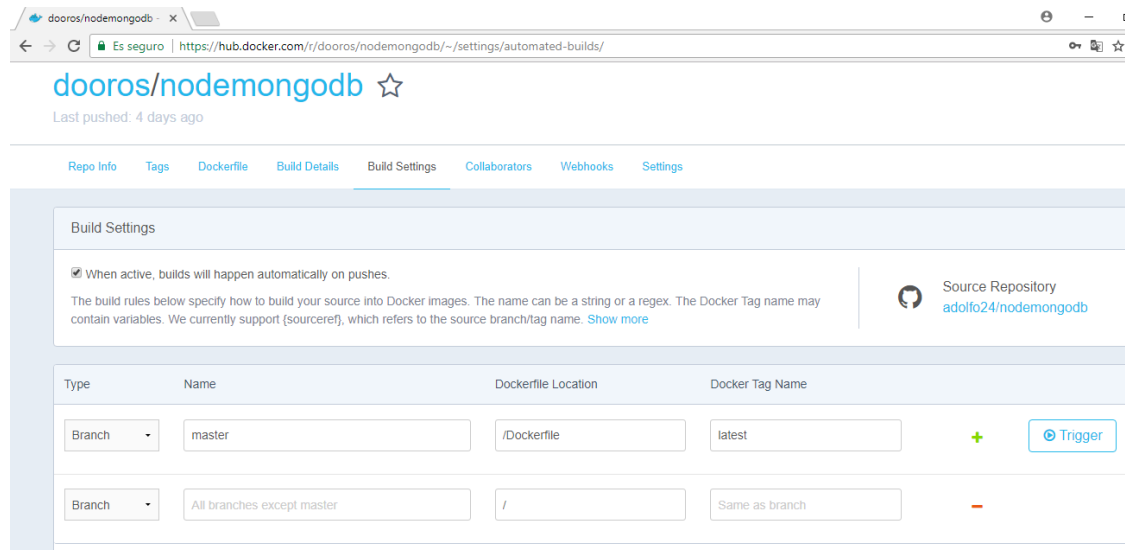
Para crear nuestro contenedor de forma automática a partir de un repositorio de GitHub tendremos que subir nuestro archivo Dockerfile a nuestro repositorio.



container with mongodb and nodejs



Una vez este nuestro archivo en el repositorio tendremos que enlazar la creación de nuestro contenedor con el repositorio de GitHub en la plataforma de Docker Hub.



De esta forma cada vez que se genere una modificación en nuestro repositorio se creara automáticamente el contenedor personalizado.

## Script de configuración

Creación de clúster Docker Swarm y servicios

### **cluster.sh**

```
#!/usr/bin/env bash
for i in "1" "2" "3"; do
    docker-machine create -d virtualbox nodo-$i
done
MANAGER_IP=$(docker-machine ip nodo-1)
eval $(docker-machine env nodo-1)
docker swarm init --advertise-addr $MANAGER_IP
MANAGER_TOKEN=$(docker swarm join-token -q manager)
WORKER_TOKEN=$(docker swarm join-token -q worker)
for i in "2" "3";do
    WORKER_IP=$(docker-machine ip nodo-$i)
    eval $(docker-machine env nodo-$i)
    docker swarm join --token $WORKER_TOKEN --advertise-addr $WORKER_IP
$MANAGER_IP:2377
done
for i in "1" "2" "3"; do
    docker-machine ssh nodo-$i "sudo mkdir -p /home/"
    docker-machine scp -r `pwd` nodo-$i:/home/
    docker-machine ssh nodo-$i "sudo mkdir -p /data/db/"
    eval $(docker-machine env nodo-$i)
    docker pull dooros/nodemongodb
done
eval $(docker-machine env nodo-1)
docker pull dockersamples/visualizer
docker stack deploy --compose-file docker-compose.yml project
docker service ls
docker-machine ls
```

Los pasos que sigue el Script anterior son:

1. Creación de las máquinas virtuales en VirtualBox.
2. Configuración del clúster Docker Swarm.
3. Copiamos la aplicación a los directorios locales de cada una de las máquinas virtuales.
4. Creamos nuestros servicios a partir del fichero docker-compose.yml
5. Listamos nuestros servicios y máquinas virtuales

***docker-compose.yml***

```
version: "3.3"
services:
  api:
    image: dooros/nodemongodb
    links:
      - db
    ports:
      - "80:3000"
    volumes:
      - aplicacion:/home
    command: bash -c "node /home/app.js"
    deploy:
      replicas: 4
  db:
    image: dooros/nodemongodb
    ports:
      - "27017:27017"
    volumes:
      - data-db:/data/db
    deploy:
      placement:
        constraints: [node.role == manager]
  visualizer:
    image: dockersamples/visualizer
    ports:
      - "8080:8080"
    stop_grace_period: 1m30s
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    deploy:
      placement:
        constraints: [node.role == manager]
volumes:
  aplicacion:
    driver_opts:
      o: bind
      device: /home/proyecto/
  data-db:
    driver_opts:
      device: /data/db/
      o: bind
```



En el archivo anterior se crean los siguientes servicios:

1. El servicio API que monta un volumen en el que se encuentra nuestra aplicación de API REST
2. El Servicio DB que monta un volumen data-db en el que van estar almacenado los datos de la base de datos.
3. El servicio visualizer en el que podremos observar nuestro clúster Swarm y los servicios que hay en cada uno de los nodos.

## Ejecución

Para crear nuestro clúster de Docker Swarm tendremos que descargar el siguiente repositorio

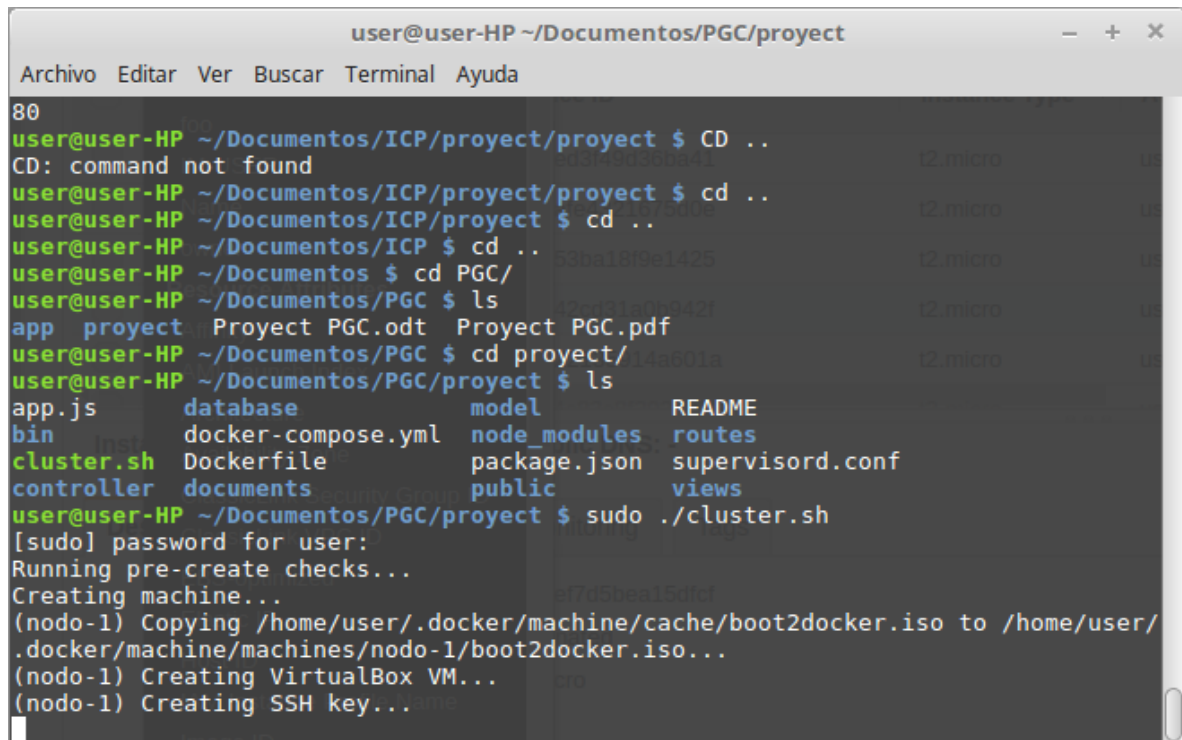
<https://github.com/adolfo24/nodemongodb>

```
cd nodemongodb/proyect
```

Creación de clúster

```
sudo chmod +x cluster.sh  
./cluster.sh
```

Con esto se creará automáticamente nuestro clúster y nuestros servicios.



```
user@user-HP ~/Documentos/PGC/proyect
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
80
user@user-HP ~/Documentos/ICP/proyect/proyect $ CD ..
CD: command not found
user@user-HP ~/Documentos/ICP/proyect/proyect $ cd ..
user@user-HP ~/Documentos/ICP/proyect $ cd ..
user@user-HP ~/Documentos/ICP $ cd ..
user@user-HP ~/Documentos $ cd PGC/
user@user-HP ~/Documentos/PGC $ ls
app  proyect  Project PGC.odt  Project PGC.pdf
user@user-HP ~/Documentos/PGC $ cd proyect/
user@user-HP ~/Documentos/PGC/proyect $ ls
app.js      database      model          README
bin          docker-compose.yml  node_modules  routes
cluster.sh  Dockerfile     package.json  supervisord.conf
controller  documents     public        views
user@user-HP ~/Documentos/PGC/proyect $ sudo ./cluster.sh
[sudo] password for user:
Running pre-create checks...
Creating machine...
(nodo-1) Copying /home/user/.docker/machine/cache/boot2docker.iso to /home/user/.docker/machine/machines/nodo-1/boot2docker.iso...
(nodo-1) Creating VirtualBox VM...
(nodo-1) Creating SSH key...
```

```
user@user-HP ~/Documentos/PGC/proyecto
Archivo Editar Ver Buscar Terminal Ayuda
master      IM-USER      100% 958      0.9KB/s  00:00
master      Name            ed3149d36 100% 876      0.9KB/s  00:00
master      Name            42cd31a0b942f 100% 41        0.0KB/s  00:00
master      Name            22135914a601a 100% 41        0.0KB/s  00:00
package.json 432167      100% 335      0.3KB/s  00:00
Using default tag: latest
latest: Pulling from dooros/nodemongodb
c954d15f947c: Pull complete
c3688624ef2b: Pull complete
848fe4263b3b: Pull complete
23b4459d3b04: Pull complete
36ab3b56c8f1: Pull complete
5c327b69da8d: Pull complete
86b47eebf4f3: Pull complete
11acfb5fd425: Pull complete
d4cb10bcc803: Pull complete
cfc938cb09ce: Pull complete
1ce9672bc952: Downloading 62.65MB/92.01MB
68e262f4fe37: Download complete
f32e7a956cea: Download complete
0142a00f3ce2: Downloading 675.4kB/21.85MB
f8c11335ff43: Downloading 804.2kB/19.34MB
5dad816db764: Waiting
```

```
Creating network project_default
Creating service project_api
Creating service project_db
Creating service project_visualizer
ID          NAME          MODE          REPLICAS          IMAGE          PORTS
qy87amden2pm project_api    replicated:1x 4/4                dooros/nodemongodb:latest *:80->3000/tcp
xenkjov283rg project_db     replicated:1x 1/1                dooros/nodemongodb:latest *:27017->27017/tcp
f0q5buhgu1me project_visualizer replicated:0x 0/1                dockersamples/visualizer:latest *:8080->8080/tcp

user@user-HP ~/Documentos/PGC/proyecto $ docker-machine ls
NAME        ACTIVE   DRIVER        STATE     URL                         SWARM   DOCKER        ERRORS
nodo-1     *        virtualbox    Running  tcp://192.168.99.100:2376   v18.02.0-ce
nodo-2     -        virtualbox    Running  tcp://192.168.99.101:2376   v18.02.0-ce
nodo-3     -        virtualbox    Running  tcp://192.168.99.102:2376   v18.02.0-ce

user@user-HP ~/Documentos/PGC/proyecto $
```

Una vez haya terminado el Script veremos en pantalla los servicios disponibles y la lista de los nodos del clúster.

## Pruebas de Funcionamiento.

Cuando ya haya terminado podremos ver el servicio de API REST en cualquiera de las direcciones IPs de los nodos del clúster.



Procederemos a registrar un nuevo usuario



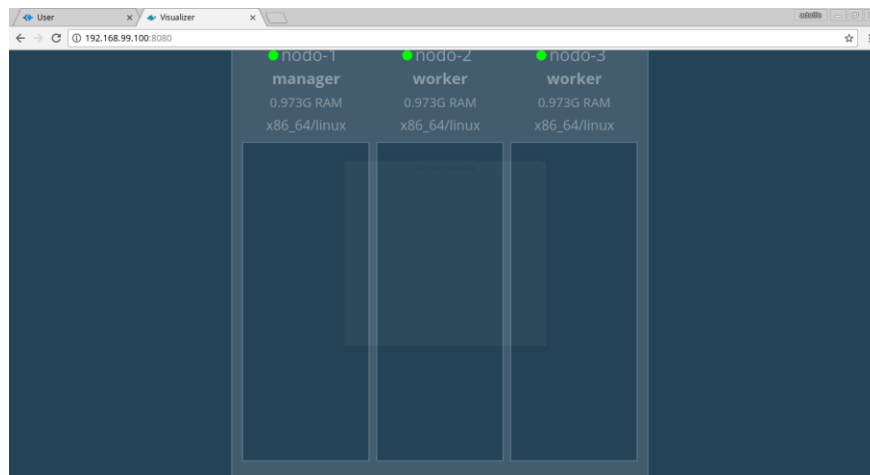
Y por último listaremos el usuario desde la aplicación:



Ahora eliminaremos todos los servicios con el comando

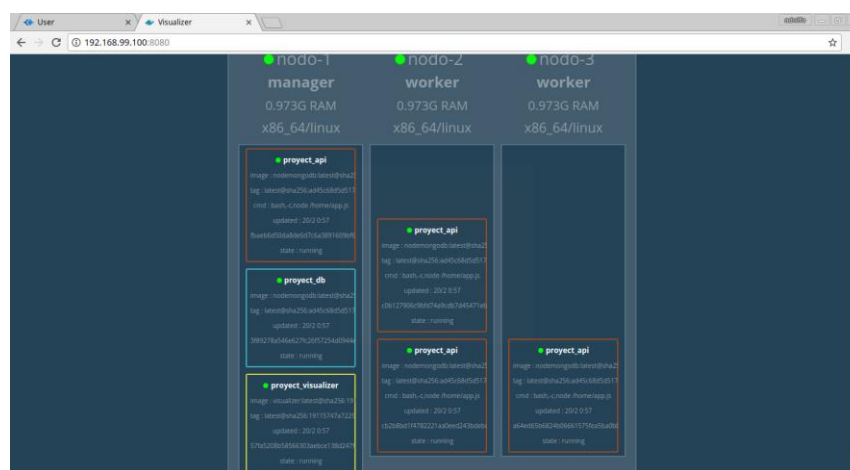
```
docker stack rm project
```

y veremos en nuestro visualizer en el puerto 8080 que ya no existen servicios



Crearemos los servicios nuevamente con el comando

```
docker stack deploy --compose-file docker-compose.yml project
```



Si visualizamos nuevamente los usuarios veremos que no se han perdido los datos y siguen en nuestra aplicación.



## Conclusiones

1. Docker Swarm es una plataforma de orquestación de contenedores de fácil uso para la creación de clúster. Con ella podemos montarnos un servicio de alta disponibilidad para nuestras aplicaciones haciendo uso de la replicación.
2. Docker Hub es una plataforma en la que podemos crear nuestros contenedores personalizados y actualizar de forma automatizada por medio de la herramienta GitHub.
3. En cuanto al uso del Docker Swarm se puede decir que solo basta tener conocimientos generales de Docker para poder poner ejecución los servicios que requiramos. Existe una gran facilidad de uso debido a que es una herramienta propia de Docker y no necesitamos aprender otro tipo de tecnología para ponerlo en práctica.

## Referencias

<https://docs.docker.com/>

<https://docs.docker.com/engine/swarm/>