**Documentación de la implementación de Cluster en Docker Swarm**

## CONFIGURACIÓN DE LA INTERFAZ DE RED (PRIVADA)

Estos comandos fueron utilizados para la comunicación de las máquinas que conforman nuestro cluster.

1. Verificar con el comando ifconfig la interfaz de red que utilizas.

2. En nuestro caso, la interfaz de red es enp0s8. y la dirección IP 192.168.0.100, la cual dejaremos fija (estatica).

3. Configurar el archivo de interfaz de red ifcfg-enp0s8.

4. Sudo nano /etc/sysconfig/network-scripts/ifcfg-enp0s8

5. La configuración es la siguiente:

```
BOOTPROTO=static
ONBOOT=yes
IPADDR0=192.168.0.100
NETMASK=255.255.255.0
GATEWAY0=192.168.0.1
DNS1=192.168.0.1
```

6. Posteriormente hacemos ping con todos los nodos para verificar la conectividad dentro del cluster.Ping “Dirección Ip”

7. Se haría lo mismo con los otros 2 master (101, 102) y los 3 nodos (110, 111, 112).

8. Guardamos los cambios y procedemos a detener el NetworkManager que no es más que un programa que gestiona la configuración de la red.

- systemctl stop NetworkManager
- systemctl disable NetworkManager
- systemctl start NetworkManager
- systemctl enable NetworkManager
- systemctl restart network.service

## IMPLEMENTACIÓN DE ALTA DISPONIBILIDAD CON DOCKER SWARM

Para nuestro proyecto utilizamos 3 máster tomando en cuenta la documentación de docker swarm que especifica que es necesario tener más del 50% de los máster arriba para que se cumpla la alta disponibilidad por ende es necesario tener 3 máster considerando la caída de uno.

1. Preparar todos los nodos(máster, esclavos) con las dependencias requeridas
   ```
   $ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
   ```
2. Descargar el repositorio que contendrá docker Swarm
   ```
   $ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
   ```
3. Procederemos a la instalación de docker swarm en cada una de las máquinas que formarán parte de nuestro cluster
   ```
   $ sudo yum install -y docker-ce
   ```
4. Ahora habilitaremos el servidor y el cliente de docker swarm con los siguientes comandos
   ```
   $ sudo systemctl enable docker
   $ sudo systemctl start docker
   ```
5. Ahora iniciaremos el primer administrador de clúster, éste proceso se hace en el administrador para asegurarse de que toda la comunicación se realice sólo a través de una red privada.
   ```
   $ sudo docker swarm init --advertise-addr=enp0s8 --data-path-addr=enp08.
   ```
6. En éste paso enlistamos los nodos actuales y sus estados.
   ```
   sudo docker node ls
   ```

7. Ahora agregamos más nodos master, se genera el token desde el primer master que es el administrador.
   ```
   manager1$ sudo docker swarm join-token manager
   ```

8. Posteriormente desactivamos firewall en todos los nodos con el siguiente comando.

   ```
   # systemctl disable firewalld
   # systemctl stop firewalld
   ```

9. En el segundo master utilizaremos el siguiente comando para agregarlo al cluster.
   ```
   $ sudo docker swarm join --advertise-addr=eth1 --data-path-addr=eth1 --token SWMTKN-1-05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-304rmelcpj5k46baa59einuv8 192.168.0.101:2377
   ```

10. Se repite el paso 8 para agregar más nodos master.

11. Posteriormente verificamos el estado actual de cluster con el siguiente comando.
    `$ sudo docker node ls`

12. En éste paso generemos los tokens para que los nodos trabajadores puedan unirse al cluster.
    `$ sudo docker swarm join-token worker`

13. Para agregar un trabajador al enjambre tenemos que ejecutar el siguiente comando.
    `$ sudo docker swarm join --advertise-addr=enp0s8 --data-path-addr=enp0s8 --token SWMTKN-1-05r99dbfwrvg4ic31783gk9o24sq9hkkdt4ruoaybmpzs3dtor-a9ujgk4iy3f86bs7xecysta2n192.168.0.101:2377`

14. Con el comando "`$ sudo docker node ls`" enlistamos los nodos unidos al cluster.

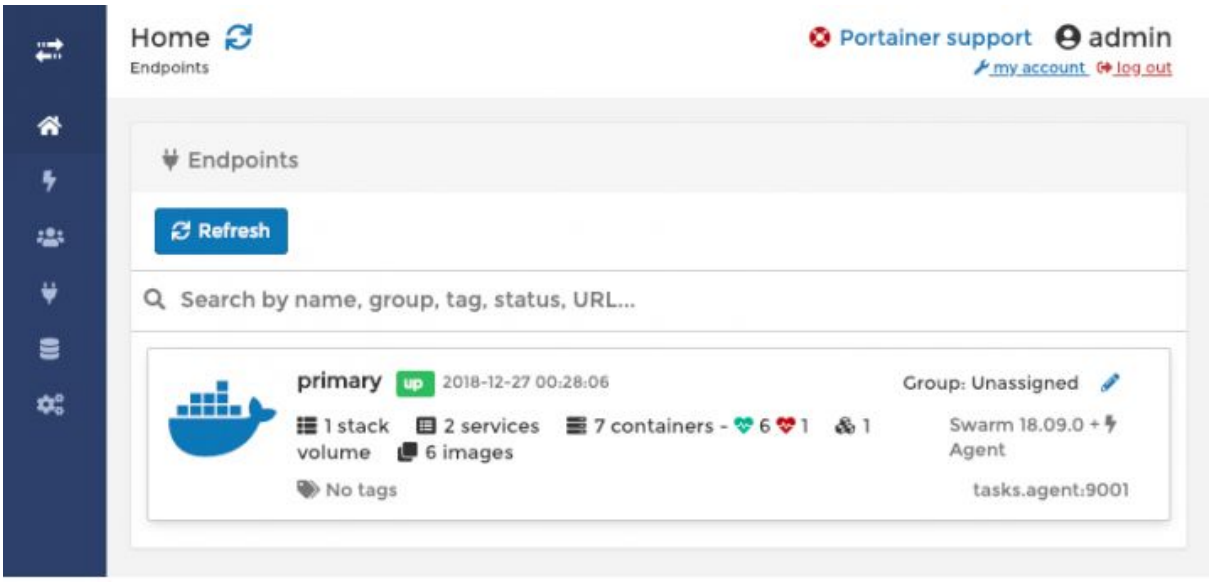
```
$ sudo docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATU
ppp9aa6itx4r3e4u4rsbb6u7g     manager1            Ready               Active              Leader
xskfh4of12jogw29jklawcy2b     manager2            Ready               Active              Reachable
f81xxoyhwbh745nurw2nur570 *   manager3            Ready               Active              Reachable
s6lwqd5nir2u4pva58uy5ryhy     worker1             Ready               Active
n3efneuhnwa57869tox6sdlhv     worker2             Ready               Active
```

15. Para tener una mejor administración de nuestro cluster, implementamos una interfaz web (Dashboard) llamado Portainer, utilizamos el siguiente comando para descargarlo en cualquiera de los nodos master.
    `$ curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o portainer-agent-stack.yml`

Portainer publicará en el puerto 9000, si desea utilizar un puerto diferente, simplemente edite el primer valor en el archivo que acabamos de descargar:

```
      - "9000:9000"
```

16. Con el siguiente comando levantamos la interfaz de Portainer para ver el Dashboard en el puerto 9000.
    `$ sudo docker stack deploy --compose-file=portainer-agent-stack.yml portainer`




## Egrafia

A continuación listamos los link de las páginas que previamente fueron investigadas para la implementación de nuestros Cluster en la herramienta Docker Swarm


1. http://dockerlabs.collabnix.com/intermediate/Implementing_High_Availability_with_Docker_Swarm.html
2. https://latinsource.wordpress.com/2017/04/17/configurar-ip-estatica-en-centos-7/
3. https://www.kubeclusters.com/docs/How-to-Deploy-a-Highly-Available-kubernetes-Cluster-with-Kubeadm-on-CentOS7
4. https://docs.docker.com/engine/swarm/