Universidad Mariano Gálvez de Guatemala

Ingeniería en Sistemas

Sistemas operativos 2 sección A

Proyecto – Entrega final

Integrantes:

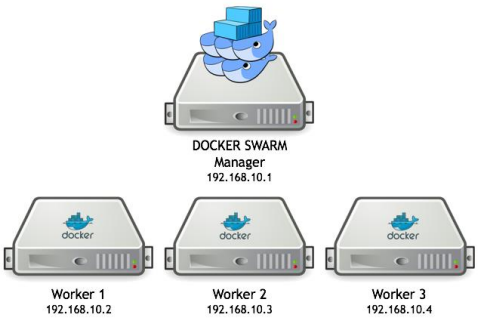
Jonatán Rodolfo Suruy Figueroa 3190-15-2425

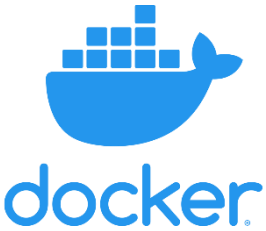
José Antonio Rojas Gómez 3190-17-2307

Jorge Eduardo Miguel Aguilar Oliva 3190 – 17 -11155

Santos Yonatan Zurdo Sunun 3190-14-6019

Francisco Alejandro Barrios Meléndez 3190-15-2711





Introducción

El siguiente documento comprende una guía para levantar la entrega final del proyecto de sistemas operativos 2. El alcance fue el siguiente: 3 nodos manager, 3 nodos worker, corriendo una imagen de Nginx con alta disponibilidad (si falla un manager otro toma el liderato). Lo anterior fue realizado en Docker Swarm.

El ambiente fue completamente virtualizado. Se utilizaron 6 docker machines para el clúster dividas de la forma explicada anteriormente. Las especificaciones de las Docker machines son las siguientes: 2 GB de RAM, 2 GHZ de procesador. El sistema del host utilizado fue Ubuntu 18.04 x64.

Guía

Requisitos y ambiente para el cluster

Docker 1.13 o una versión más actualizada

Opción Docker Machine habilitada

Sistema operativo Ubuntu 18.04 (opcional)

Cluster

Para iniciar el cluster se crearán las Docker machines que funcionarán como los nodos a utilizarse. Primero se crean 3 nodos manager. Los mismos se reflejan como leaders al hacer la consulta de nodos.

docker-machine create --driver virtualbox manager-1

docker-machine create --driver virtualbox manager-2

docker-machine create --driver virtualbox manager-3

Luego se crean los worker

docker-machine create --driver virtualbox worker-1

docker-machine create --driver virtualbox worker-2

docker-machine create --driver virtualbox worker-3

Podemos listas los nodos con el siguiente comando, veremos el estado de los mismo y su dirección Ip.

docker-machine ls

Para continuar se abren 6 terminales y nos conectamos a cada nodo por medio del siguiente comando.

docker-machine ssh "Nombre del nodo"

Por lo tanto, se escribirá:

docker-machine ssh manager-1

docker-machine ssh manager-2

docker-machine ssh manager-3

docker-machine ssh worker-1

docker-machine ssh worker-2

docker-machine ssh worker-3

Luego de obtener la conexión a cada nodo iniciamos a armar el cluster. Primero levantamos el primer Manager. Necesitamos las ip de los nodos las cuales podemos ver na lista que generamos "docker-machine ls".

docker swarm init --advertise-addr 192.168.99.100

Luego, adentro de nuestro manager hacemos una consulta de los nodos conectados. Solo aparecerá nuestro nuevo nodo leader. Solo los nodos leader pueden ejecutar este comando.

docker node ls

Al generar nuestro primer leader obtenemos dos opciones importantes que nos facilitaran el resto de la creación del cluster.

docker swarm join-token worker

Con este comando podemos obtener el token para conectar nodos worker al leader.

Con el siguiente comando solicitamos un token para agregar mas leaders al cluster. Inicialmente solo un nodo es leader, mientras el resto de managers esperan en modo "reachable", lo que significa que están disponibles en caso de que el leader falle, si eso sucede uno de los disponibles toma el puesto.

docker swarm join-token worker

Para terminar la creación solo queda ingresar el token correspondiente en cada terminal. Token worker en las terminales donde estemos conectados a nodos worker y los token manager en las terminales en las que estemos conectados a nodos que serán leader.

Por ulitmo podemos ver todos los nodos conectados al cluster al ejecutar "docker node ls".

Alta Disponibilidad

Para comprobar la alta disponibilidad podemos detener el funcionamiento de un nodo leader. Para esto regresamos a la terminal que tenemos en el host y detenemos el leader principal con el siguiente comando (el nombre dependerá de cual sea el líder en el momento).

docker stop manager-1

Esperamos a que se confirme que el servicio está detenido y luego podemos volver a la terminal donde teníamos la conexión con ese nodo. De inmediato veremos que la conexión fue interrumpida, por lo que buscamos la otra terminal en la que esté conectada un leader, luego listamos de nuevo los nodos por medio de "docker node ls" y podemos observar que otro nodo manager a tomado el status de Leader mientras que el anterior no se refleja como disponible.

Desplegar imagen en cluster

Para continuar veremos cómo se despliegan imágenes en el cluster. Podemos tomar la imagen tanto del docker hub así como también de un repositorio privado utilizando la siguiente línea de comando en la que especificamos el número de imágenes a desplegar "numero de nodos" que incluye leaders, también definimos el nombre que le daremos al servicio y por último, el nombre de la imagen a descargar junto con su versión (si no escribimos el número de versión automáticamente se descarga la versión más actualizada).

docker service create --replicas "numero imagenes" -p 80:80 --name "nombre servicio" "nombre imagen"

Para la demostración se utilizó una imagen de Nginx (Latest), con 6 réplicas y nombre "servicio"

docker service create --replicas 3 -p 80:80 --name serviceName nginx

Podemos listar el servicio funcionando en los nodos por medio de:

docker service ls

Para el manejo de imágenes corriendo podemos aumentar la escala y añadir más imágenes a un nodo. Por ejemplo, podríamos hacer que el nodo principal (manager-1) este corriendo dos imágenes en vez de una por medio de la siguiente línea de código (este código solo se puede ejecutar en un nodo leader).

docker service scale Nginx=2

Para opciones más avanzadas podemos inspeccionar un nodo especifico o realizar una consulta sobre el mismo con las siguientes líneas de código.

docker node inspect "nombre nodo"

Línea para inspecciona un nodo desde el mismo.

docker node inspect self

Desde un manager también podemos apagar nodos, los cuales al ser apagados se reflejan como "Drain".

docker node update --availability drain worker-1

También podemos actualizar imágenes desde un manager con la siguiente línea en la que especificamos el nombre de la imagen y su nueva versión.

docker service update --image "nombre imagen":"versión"

Con la siguiente línea también podemos eliminar un servicio de nuestro cluster.

docker service rm serviceName

Tambien podemos hacer que un nodo salga del cluster (necesitaría de nuevo un token para ingresar), ejecutando la siguiente línea de código en la terminal del nodo que deseamos dejar fuera del cluster (es posible que sea necesario añadir "--force" al final para completar la acción).

docker swarm leave

Link Video:

https://drive.google.com/file/d/11qdOhYeI5w8pyADpDR54GEKkPFGiNRBJ/view