README.md

Parcial 3

Universidad ICESI
Curso: Sistemas Distribuidos
Docente: Daniel Barragán
Tema: Instalación, configuración y empleo de sistemas de ficheros distribuidos con docker swarm
Estudiantes: Carolina Zúñiga y Paula Bolaños

Objetivos

- Realizar de forma autónoma el aprovisionamiento automático de infraestructura
- Diagnosticar y ejecutar de forma autónoma las acciones necesarias para lograr infraestructuras estables
- • Emplear un sistema de ficheros distribuido como volumen persistente en un cluster de docker swarm

Prerrequisitos

- Docker
- Docker Swarm
- GlusterFs (Sistema de ficheros distribuidos)

Descripción

Desplegar un ambiente compuesto por al menos tres nodos (un nodo maestro, dos nodos de trabajo). Los tres nodos deben ser parte de un cluster de glusterfs. Desplegar un cluster de docker swarm sobre los tres nodos aprovisionados. Demostrar que es posible desplegar un contenedor de postgres sobre el nodo de trabajo uno, realizar inserciones a la base de datos, destruir el contenedor, desplegarlo en el nodo de trabajo dos y finalmente realizar una consulta que recupere la información ingresada previamente. Cada nodo de trabajo debe tener un disco extra de 5 Gigabytes para las configuraciones de glusterfs. Puede emplear un archivo de docker compose o el comando docker service para la realización de las pruebas con postgres.

Actividades

1. Desplegar un ambiente compuesto por 3 nodos: 1 manager y 2 workers, con un disco extra de 5GB para las configuraciones del glusterfs.

Para el despliegue de los nodos se utilizó un archivo Vagrantfile, en el cual se hace el aprovisionamiento automático de cada uno. Para agregar los discos extra a cada nodo, estos fueron definidos como varibales en la parte superior y después se hizo referencia a dichas variables en la configuración de virtualbox. De esta forma, el disco se creaba sólo si no había sido creado. Con el fin de que los nodos tuviesen instalado docker y glusterfs, se aprovisionaron las máquinas por medio de SHELL con los scripts de configuración externos: configurations.sh, docker.sh y glusterfs.sh.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

VAGRANTFILE_API_VERSION = "2"

firstDisk = './firstDisk.vdi'
secondDisk = './secondDisk.vdi'
thirdDisk = './thirdDisk.vdi'
```

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.ssh.insert_key = false

  #Manager
  config.vm.define :master do |master|
    master.vm.box = "ubuntu/xenial64"
    master.vm.network :private_network, ip: "192.168.56.101"
    master.vm.provider :virtualbox do |vb|
      vb.gui = true
      vb.customize ["modifyvm", :id, "--memory", "1024","--cpus", "1", "--name", "3master" ]
      unless File.exist?(firstDisk)
        vb.customize ['createhd', '--filename', firstDisk, '--variant', 'Fixed', '--size', 5 * 1024]
      end
      vb.customize ['storageattach', :id,  '--storagectl', 'SCSI', '--port', 1, '--device', 0, '--
type', 'hdd', '--medium', firstDisk]
    end
    master.vm.provision "shell", inline: <<-SHELL
      echo "node0" > /etc/hostname
      hostname node0
    SHELL
    master.vm.provision "shell", path: "install/configurations.sh"
    master.vm.provision "shell", path: "install/docker.sh"
    # k8sm.vm.provision "shell", path: "install/compose.sh"
    master.vm.provision "shell", path: "install/glusterfs.sh"
  end

  #Worker 1
  config.vm.define :node_1 do |node1|
    node1.vm.box = "ubuntu/xenial64"
    node1.vm.network :private_network, ip: "192.168.56.102"
    node1.vm.provider :virtualbox do |vb|
      vb.gui = true
      vb.customize ["modifyvm", :id, "--memory", "512","--cpus", "1", "--name", "3node_1" ]
      unless File.exist?(secondDisk)
        vb.customize ['createhd', '--filename', secondDisk, '--variant', 'Fixed', '--size', 5 * 1024]
      end
      vb.customize ['storageattach', :id,  '--storagectl', 'SCSI', '--port', 1, '--device', 0, '--
type', 'hdd', '--medium', secondDisk]
    end
    node1.vm.provision "shell", inline: <<-SHELL
      echo "node1" > /etc/hostname
      hostname node1
    SHELL
    node1.vm.provision "shell", path: "install/configurations.sh"
    node1.vm.provision "shell", path: "install/docker.sh"
    node1.vm.provision "shell", path: "install/glusterfs.sh"
  end

  #Worker 2
  config.vm.define :node_2 do |node2|
    node2.vm.box = "ubuntu/xenial64"
    node2.vm.network :private_network, ip: "192.168.56.103"
    node2.vm.provider :virtualbox do |vb|
      vb.gui = true
      vb.customize ["modifyvm", :id, "--memory", "512","--cpus", "1", "--name", "3node_2" ]
      unless File.exist?(thirdDisk)
        vb.customize ['createhd', '--filename', thirdDisk, '--variant', 'Fixed', '--size', 5 * 1024]
      end
      vb.customize ['storageattach', :id,  '--storagectl', 'SCSI', '--port', 1, '--device', 0, '--
type', 'hdd', '--medium', thirdDisk]
    end
    node2.vm.provision "shell", inline: <<-SHELL
      echo "node2" > /etc/hostname
      hostname node2
```

```
    SHELL
    node2.vm.provision "shell", path: "install/configurations.sh"
    node2.vm.provision "shell", path: "install/docker.sh"
    node2.vm.provision "shell", path: "install/glusterfs.sh"
  end
end
```

2. Instalación y configuración del sistema de archivos distribuidos, glusterfs, en los 3 nodos.

Los siguientes pasos se deben realizar en cada nodo de trabajo.

El primer paso es relacionar las IPs de los host con las etiquetas de estos en el archivo /etc/hosts. Sin embargo, esto ya se realizó con el script configurations.sh, que hace parte del aprovisionamiento con el Vagrantfile.

Posteriormente, se procede a la instalación e inicialización de docker y glusterfs, lo cual se podría hacer con estos comandos pero ya está automatizado:

```
sudo apt install -y docker.io
sudo apt install -y glusterfs-server
sudo systemctl start glusterfs-server
sudo systemctl start docker
```

Ahora, se crea el directorio para glusterfs:

```
sudo mkdir -p /gluster/data /swarm/volumes
```

Luego, se crea la unidad de almacenamiento o filesystem qye quedará compartido o replicado en todos los nodos para el uso de glusterfs, así:

```
sudo mkfs.xfs /dev/xvdb
sudo mount /dev/xvdb /gluster/data/
```

Una vez creado el filesystem, se crea el volumen como un espejo y se inicializa.

```
sudo gluster volume create swarm-vols replica 3 node1:/gluster/data node2:/gluster/data node3:/gluster/d
volume create: swarm-vols: success: please start the volume to access data
sudo gluster volume start swarm-vols
volume start: swarm-vols: success
```

Con el volumen inicializado, se montan los glusterfs en espejo de forma local en cada nodo:

```
sudo mount.glusterfs localhost:/swarm-vols /swarm/volumes
```

3. Despliegue del cluster de docker swarm

En lo que respecta a la configuración del cluster de docker swarm, el nodo manager es el que inicia el swarm y los dos nodos workers se unen a este. Después, basta con tener acceso al nodo manager por medio del comando docker-machine env para hacer el despliegue del servicio con el docker-compose, que está más adelante.

El comando que se ejecuta en el manager cuyo resultado es el token es:

```
sudo docker swarm init
```

Dicho token lo utilizan los workers para unirse por meido del comando:

```
sudo docker swarm --token (join-token worker)
```

4. Despliegue del servicio solicitado.

Teniendo en cuenta que se solicitó el servicio postgres, se decidió realizar las pruebas de este con el siguiente docker-compose.yml, que define un servicio primario con una replica.

```
version: "3.3"

services:
  primary:
    hostname: 'primary'
    image: crunchydata/crunchy-postgres:centos7-10.3-1.8.2
    environment:
    - PGHOST=/tmp
    - MAX_CONNECTIONS=10
    - MAX_WAL_SENDERS=5
    - PG_MODE=primary
    - PG_PRIMARY_USER=primaryuser
    - PG_PRIMARY_PASSWORD=password
    - PG_DATABASE=testdb
    - PG_USER=testuser
    - PG_PASSWORD=password
    - PG_ROOT_PASSWORD=password
    - PG_PRIMARY_PORT=5432
    volumes:
    - pg-primary-vol:/pgdata
    ports:
    - "5432"
    networks:
    - crunchynet
    deploy:
      placement:
        constraints:
        - node.labels.type == primary
        - node.role == worker
  replica:
    image: crunchydata/crunchy-postgres:centos7-10.3-1.8.2
    environment:
    - PGHOST=/tmp
    - MAX_CONNECTIONS=10
    - MAX_WAL_SENDERS=5
    - PG_MODE=replica
    - PG_PRIMARY_HOST=primary
    - PG_PRIMARY_PORT=5432
    - PG_PRIMARY_USER=primaryuser
    - PG_PRIMARY_PASSWORD=password
    - PG_DATABASE=testdb
    - PG_USER=testuser
    - PG_PASSWORD=password
    - PG_ROOT_PASSWORD=password
    volumes:
    - pg-replica-vol:/pgdata
    ports:
    - "5432"
    networks:
    - crunchynet
    deploy:
      placement:
        constraints:
        - node.labels.type != primary
        - node.role == worker
```

```
networks:
  crunchynet:

volumes:
  pg-primary-vol:
  pg-replica-vol:
```

El comando para desplegar el servicio es:

```
docker stack deploy -c docker-compose.yml postgres
```

5. Dificultades encontradas:

- Los controladores SATA y SCSI no estaban disponibles en OSx y no permitían desplegar el disco externo, sin embargo, cambiando el controlador por IDE fue posible realizar el despliegue.

Referencias

- https://docs.docker.com/samples/library/postgres/#-via-docker-stack-deploy-or-docker-compose
- http://info.crunchydata.com/blog/an-easy-recipe-for-creating-a-postgresql-cluster-with-docker-swarm
- https://docs.docker.com/engine/swarm/stack-deploy/
- http://embaby.com/blog/using-glusterfs-docker-swarm-cluster/
- https://everythingshouldbevirtual.com/virtualization/vagrant-adding-a-second-hard-drive/

URL Repositorio: www.github.com/