Definición de una infraestructura cloud de alta disponibilidad en un entorno distribuido para el despliegue de una plataforma IoT
Manual de instalación

JOSE DAVID ROJAS AGUILAR

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2019

Definición de una infraestructura cloud de alta disponibilidad en un entorno distribuido para el despliegue de una plataforma IoT
Manual de instalación

JOSE DAVID ROJAS AGUILAR

Trabajo de grado para optar al título de Ingeniero de Sistemas

Director
GABRIEL RODRIGO PEDRAZA FERREIRA
PhD en Ciencias de la computación

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2019

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

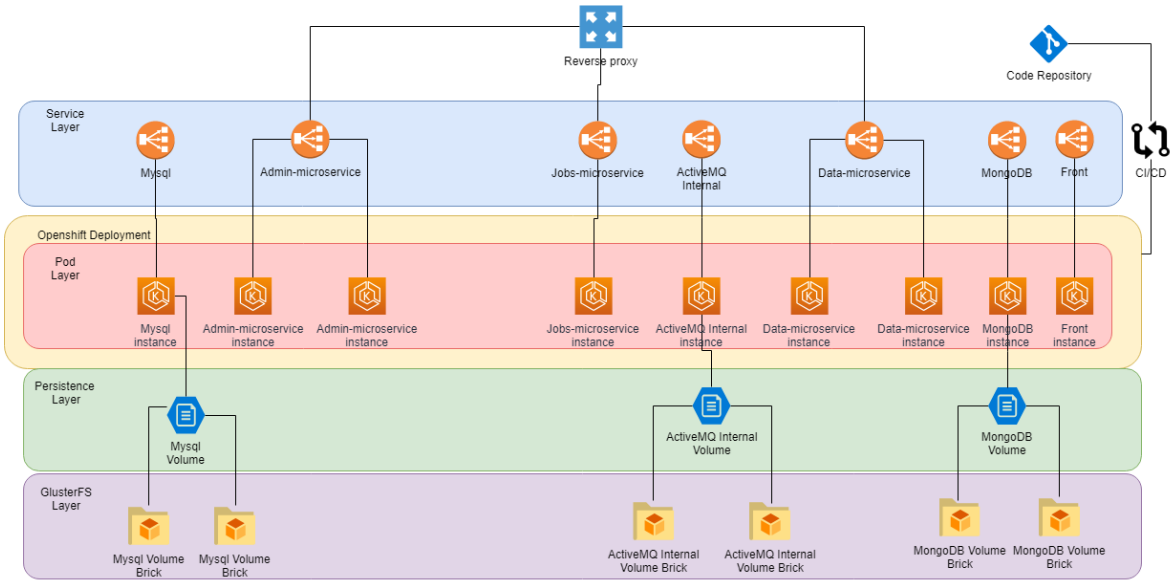
LISTA DE TABLAS

pág.

Presentación

En el presente documento se presenta como desplegar la arquitectura de la figura 1 en una infraestructura distribuida.

Figura 1: Arquitectura diseñada durante el proyecto



Los archivos de configuración a los que se hace referencia en el presente documento se encuentra en el siguiente repositorio de GitHub: https://github.com/JoseDRojasA/thesis-files.

1. Hardware

Los servidores se clasifican en los siguientes roles:

- **DNS:** Es el servidor encargado de traducir nombres de dominio a direcciones ip.
- **Maestro:** Es el servidor encargado de recibir las
- **Esclavo:** Es el servidor encargado de desplegar las instancias de la arquitectura.
- **Persistencia:** Es el servidor encargado de persistir datos. Suele poseer unidades de almacenamiento de alta velocidad.

La infraestructura mínima de alta disponibilidad se compone de:

- 1 servidor DNS
- 1 servidor maestro
- 2 servidores esclavo y persistencia.

Los requisitos mínimos y recomendados para los servidores se encuentran en la documentación oficial de Openshift[1].

Se recomienda hacer uso de servidores alojados en Amazon Web Services, Google Cloud Platform ó Azure.

En caso de tener una infraestructura local, se recomienda hacer uso de una infraestructura de red de alta velocidad como cables 10 Gigabit Ethernet.

[1] https://docs.openshift.com/container-platform/3.11/install/prerequisites.html

2. Software

Se recomienda hacer uso de CentOS[1] como sistema operativo para los diferentes servidores por su estabilidad, velocidad y comunidad.

2.1. CONFIGURACIÓN BÁSICA DE NODOS

Se actualizan los paquetes del sistema operativo.

```
yum update
yum upgrade
```

Se genera una clave RSA la cual se usará para la comunicación entre los servidores.

```
ssh-keygen
```

Realizamos el siguiente comando entre los diferentes servidores de tal forma que puedan comunicarse entre si por medio de la clave RSA.

```
ssh-copy-id -i ~/.ssh/id_rsa.pub <ip_hosts>
```

Posteriormente, Se configura la interfaz conectada a la red con dirección ip estatica. Para ello, se ubica dentro de la carpeta /etc/sysconfig/network-scripts la configuración de la interfaz conectada en la red[2] y se ingresan las variables de la siguiente forma:

- NM_CONTROLLED=no
- BOOTPROTO=dhcp

[1] Página oficial de CentOS
[2] Configuración básica de nodos/ifcfg-ens3

- IPADDR=<ip_nodo>
- NETMASK=<mascara_red>
- GATEWAY=<gateway_red>
- DNS1=<ip_servidor_dns>

Se añade el servidor DNS al archivo /etc/sysconfig/network-scripts/resolv.conf[3] y se reinicia la interfaz de red.

```
service network restart
```

2.2. INSTALACIÓN DEL SERVIDOR DNS

Se instalan los paquetes de bind9.

```
yum install bind bind-utils -y
```

Se editan los siguientes archivos de configuración: /etc/named.conf[4], /var/named/-forward.local.cluster[5] y /var/named/reverse.local.cluster[6] en los cuales configuramos la resolución de nombres de dominio.

Se habilita el servicio de dns.

```
systemctl enable named
systemctl start named
```

Se añade el puerto 53 como excepción al cortafuegos y se reinicia para que los cambios se vean aplicados.

[3]Configuración básica de nodos/resolv.conf
[4]Configuración de servidor DNS/named.conf
[5]Configuración de servidor DNS/forward.local.cluster
[6]Configuración de servidor DNS/reverse.local.cluster

```
firewall-cmd --permanent --add-port=53/tcp
firewall-cmd --permanent --add-port=53/udp
firewall-cmd --reload
```

Se modifican los permisos, grupos y SELinux de named.

```
chgrp named -R /var/named
chown -v root:named /etc/named.conf
restorecon -rv /var/named
restorecon /etc/named.conf
```

2.3. CONFIGURACIÓN GLUSTERFS

El siguiente proceso se realiza en cada uno de los servidores de persistencia de la arquitectura.

2.3.1. instalación de paquetes de glusterfs

Se instalan los repositorios de GlusterFS y se instala el paquete correspondiente.

```
wget http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release
   ↪ -6-8.noarch.rpm
rpm -ivh epel-release-6-8.noarch.rpm
wget -P /etc/yum.repos.d https://download.gluster.org/pub/gluster/
   ↪ glusterfs/6/LATEST/CentOS/glusterfs-rhel8.repo
yum install glusterfs-server
```

Se inicia el servicio de glusterFS.

```
service glusterd start
```

Se configura SELinux en el archivo /etc/sysconfig/selinux[7] para deshabilitar SELinux.

Se realiza flush de iptables para dar acceso entre los servidores.

```
iptables -F
```

2.3.2. configuración de un volumen glusterfs Se crea una carpeta donde se almacenara el brick de glusterfs en cada uno de los servidores de persistencia.

```
mkdir /export/gluster/mysql
```

En uno de los servidores se crea e inicia el volumen GlusterFS.

```
gluster volume create mysql replica 2 persistence1.local.cluster:/
    ↪ export/gluster/mysql persistence2.local.cluster:/export/gluster/
    ↪ mysql
gluster volume start mysql
```

Es importante notar que en el ejemplo dado se está creando un volumen llamado 'mysql' con 2 bricks: Uno ubicado en el servidor persistence1.local.cluster en la ruta /export/gluster/mysql y otro en el servidor persistence2.local.cluster en la ruta /export/gluster/mysql.

Para añadir un nuevo brick para escalar horizontalmente la arquitectura, se realiza el proceso de instalación de los paquetes de GlusterFS descrito en la subsección 2.3.1.

Posteriormente se configura el brick adicional.

```
gluster volume add-brick mysql replica 3 persistence3.local.cluster:/
    ↪ export/gluster/mysql
```

[7] Configuración de GlusterFS/selinux

En el ejemplo se añade un tercer brick ubicado en el servidor persistence3.local.cluster en la ruta /export/gluster/mysql.

Para el despliegue de la arquitectura, se realiza el proceso descrito en esta sección para cada uno de los siguientes volúmenes.

Tabla 1: Configuración de bricks GlusterFS

Volumen	Usuario Id	Grupo Id	Permisos
activemq	0	0	755
mongo	184	184	777
mysql	27	0	777

2.4. INSTALACIÓN DE OPENSHIFT ORIGIN

En un computador externo a la arquitectura se configura el servidor DNS y el acceso por medio de clave RSA.

Seguidamente, se descarga el repositorio de Openshift Origin.

```
git clone https://github.com/openshift/origin.git
```

El proceso de instalación hace uso de Ansible, por lo que es necesario generar un archivo de inventory[8]. El proceso para generar el archivo inventory se encuentra en la documentación oficial de Openshift Origin[9].

Una vez se genera el archivo Inventory, se ejecutan 2 playbooks del repositorio de Openshift Origin:

1. **prerequisites.yaml** Instala y actualiza los paquetes software necesarios para

[8]Instalación de Openshift Origin/inventory
[9]Example Inventory Files

la instalación del cluster.

2. **deploy_cluster.yaml** Instala e inicia el cluster dentro de la infraestructura.

```
ansible-playbook -i inventory openshift-ansible/playbooks/prerequisites
    ↪ .yml
ansible-playbook -i inventory openshift-ansible/playbooks/
    ↪ deploy_cluster.yml
```

La instalación puede ser verificada ingresando al sistema desde la terminal del servidor maestro.

```
oc login
```

3. Despliegue de artefactos

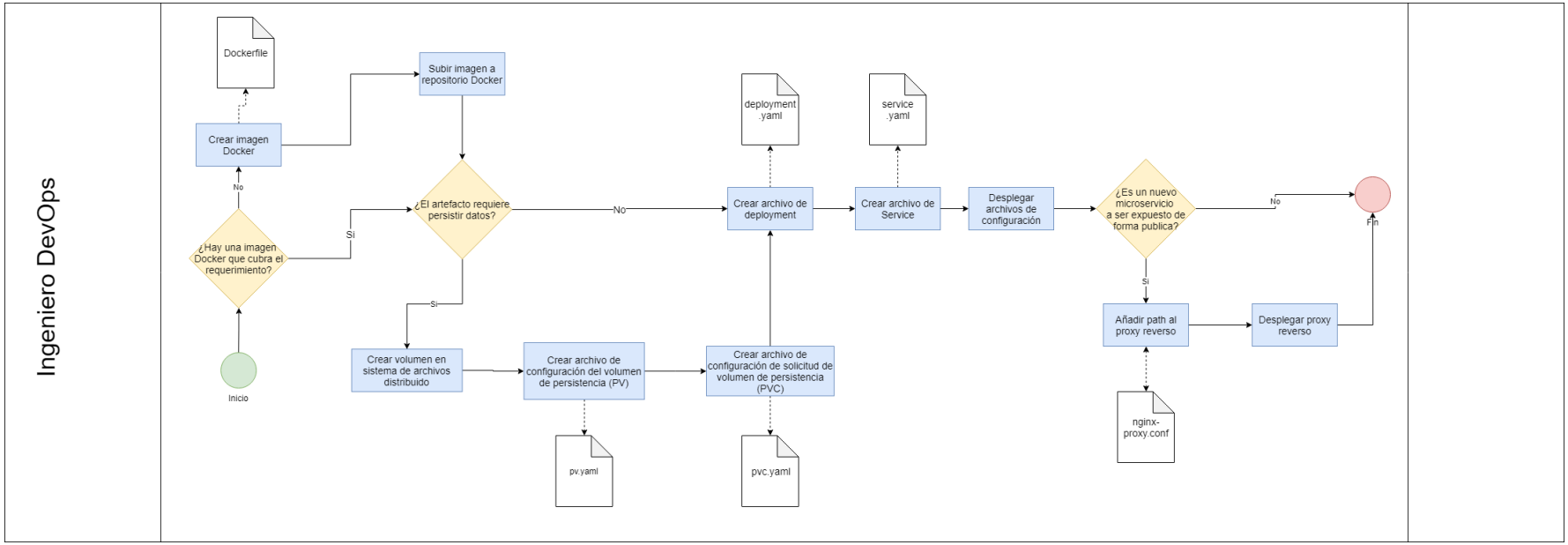
Se recomienda crear 2 proyectos para separar lo relacionado con la capa backend de la capa front.

```
oc create project backend
oc create project iot-front
```

3.1. METODOLOGÍA DE DESPLIEGUE

Durante el desarrollo del proyecto se desarrollo la siguiente metodología con el fin de generalizar el proceso de despliegue de los diferentes componentes de la arquitectura de tal forma que sea sencillo integrar nuevos componentes a la arquitectura usando la metodología propuesta en la figura 2.

Figura 2: Metodología de despliegue en BPMN



3.2. DETALLES TÉCNICOS DE LAS ACTIVIDADES DE LA METODOLOGÍA

- **Crear una imagen docker:** Se diseña un archivo Dockerfile[1], el cual se usara para crear la imagen de docker.

```
docker build . -t josedrojasa/thesis-front
```

- **Subir imagen Docker a Docker Hub:** Una vez se construye la imagen de docker, se sube a un repositorio de Docker. Para efectos prácticos, se sugiere usar Dockerhub[2], pero en producción se recomienda tener un repositorio privado de Docker[3].

```
docker push josedrojasa/thesis-front
```

- **Crear volumen en sistema de archivos distribuido:** Este proceso se encuentra en la sección 2.3.

- **Crear archivo de configuración del volumen de persistencia:** Según el sistema de archivos distribuido, se diseña un archivo de configuración de un volumen de persistencia como lo indica la documentación de Openshift Origin[4].

- **Crear archivo de configuración de solicitud de volumen de persistencia (PVC):** Se diseña el archivo de configuración de solicitud de volumen de persistencia como lo indica la documentación de Openshift Origin[5].

- **Crear archivo de deployment:** Se diseña un archivo de configuración de deployment como lo indica la documentación de Kubernetes[6].

- **Crear archivo de Service:** Se diseña un archivo de configuración de service como lo indica la documentación de Kubernetes[7].

[1] Dockerfile reference
[2] DockerHub
[3] Deploy a registry server
[4] Persistent volumes
[5] Persistent volume claims
[6] Deployments
[7] Services

- **Desplegar archivos de configuración:** Se despliegan los archivos generados en las actividades anteriores usando el comando .ºc create -f <nombre_archivo>".

```
oc create -f pv.yaml
oc create -f pvc.yaml
oc create -f deployment.yaml
oc create -f service.yaml
```

- **Añadir path al proxy reverso:** Modificar el archivo nginx-proxy.conf[8] para añadir las rutas de los endpoints del nuevo microservicio.

- **Desplegar proxy reverso:** Ejecutar la metodología de despliegue para el artefacto de proxy reverso.

3.3. CONSIDERACIONES ESPECIALES

- La comunicación entre los microservicios se realiza por medio de la capa service por lo cual debe usarse el nombre de dominio del service para comunicar los diferentes microservicios. Bajo ninguna circunstancia se debe hacer la comunicación directamente desde la capa pod ya que esta es efímera mientras que la capa service es persistente.

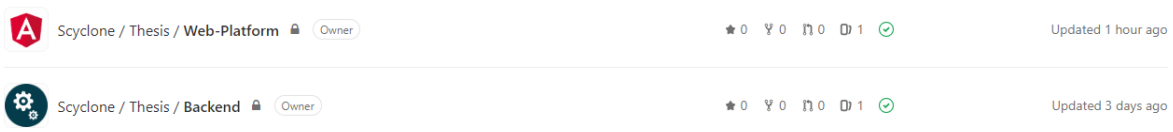
- Dentro de la arquitectura existe un microservicio que está hecho para tener una sola instancia: Job-service, ya que este microservicio tiene aquellas tareas que deben realizarse una sola vez en ciertas horas especificas.

[8] Despliegue de artefactos/proxy-reverso/nginx-proxy.conf

4. Integración continua y despliegue continuo

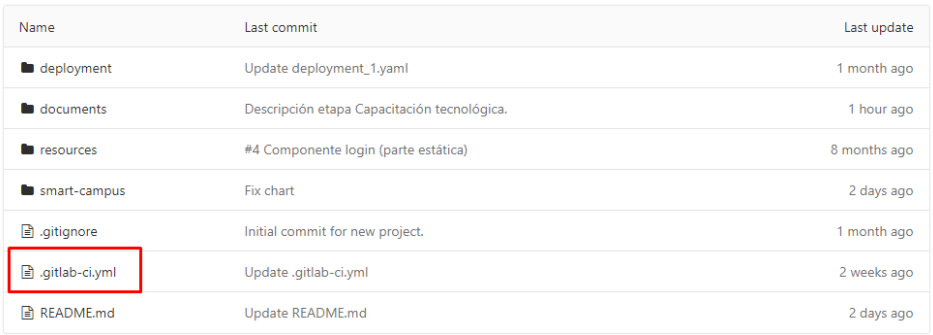
Para la integración continua y el despliegue continuo se utilizó la plataforma: Gitlab. Se crea 1 repositorio para el front y, se recomienda tener tantos repositorios como microservicios para tener un despliegue continuo mas controlado, sin embargo, por efectos prácticos del prototipo, todos los microservicios se ubicaron el mismo repositorio.

Figura 3: Proyectos en GitLab



Dentro de cada proyecto, se crea un archivo .gitlab-ci.yml[1] el cual se ejecutara cada vez que se realice un commit en la rama master del repositorio. Se recomienda fuertemente el uso de ramas para aquellos commits parciales durante el desarrollo del proyecto con el fin de reducir la cantidad de despliegues automáticos innecesarios.

Figura 4: Repositorio con archivo .gitlab-ci.yml



En el proyecto se uso un archivo para el front[2] y otro para el backend[3].

[1] Getting started with GitLab CI/CD
[2] Configuración de CD/front/.gitlab-ci.yml
[3] Configuración de CD/backend/.gitlab-ci.yml