DESARROLLO DE SOFTWARE CRÍTICO

Práctica Zookeeper/Curator

JUAN JOSÉ TIRADO ARREGUI

APACHE ZooKeeper™

Introducción

En esta práctica vamos a utilizar Curator para construir un sistema basado en Zookeeper. ZooKeeper es un servicio centralizado para mantener la información de configuración, nombrar, brindar sincronización distribuida y brindar servicios grupales.

Construcción básica del sistema

Vamos a tener N nodos que realizan mediciones (con valores aleatorios) cada 2 segundos. Arrancaremos la misma aplicación N veces, representando cada ejecución a uno de los nodos.

Gracias a Zookeeper uno de estos nodos actuará como líder. Cada nodo comprobará si lo es con el método getImLeading(). Si es el líder, entonces recogerá todas las medidas consultando los hijos de la ruta /mediciones y realizará la media. Esta media la pasará al método enviarMedia(double media) que realizará una petición GET a la url correspondiente de la práctica anterior.

Desarrollo de stack Docker para la solución propuesta

En este apartado debe desarrollarse un stack incluyendo a Zookeeper y la aplicación desarrollada de manera que baste con desplegar dicho stack para poder enviar las mediciones al servicio externo. Por simplicidad, en este stack vamos a incorporar también la API diseñada en la práctica anterior, así como a redis.

docker-compose.yml:

```
version: "3.1"

services:

  web:

    image: jjtirado/practica2:ej2 //incluimos el contenedor de la p2

    deploy:

      replicas: 5

      restart_policy:

        condition: on-failure

    environment:

      - REDIS_HOST=redis
```

```
  ports:
    - "4000:4567"
  networks:
    - webnet
visualizer:
  image: dockersamples/visualizer:stable //visualizador
  ports:
    - "8080:8080"
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock"
  deploy:
    placement:
      constraints: [node.role == manager]
  networks:
    - webnet
redis:
  image: redis //redis
  ports:
    - "6378:6378"
  deploy:
    placement:
      constraints: [node.role == manager]
  networks:
    - webnet
zookeeper:
```

```
  image: zookeeper     // zookeeper
  hostname: zookeeper
  environment:
    ZOO_MY_ID: 1
    ZOO_PORT: 2181
    ZOO_SERVERS: server.1=zookeeper:2888:3888
  ports:
      - "2181:2181"
// A continuación despliego 3 nodos con la imagen de mi practica de
   zookeeper
practicazookeeper1:
  image: jjtirado/practicazookeper:def
  deploy:
    restart_policy:
      condition: on-failure
  environment:
    - NODO=node1
    - ZOOKEEPER_HOST:"zookeeper:2181"
  depends_on:
    - zookeeper
  networks:
    - webnet
practicazookeeper2:
  image: jjtirado/practicazookeper:def
  deploy:
```

```
      restart_policy:
        condition: on-failure
    environment:
      - NODO=node2
      - ZOOKEEPER_HOST:"zookeeper:2181"
    depends_on:
      - zookeeper
    networks:
      - webnet
  practicazookeeper3:
    image: jjtirado/practicazookeper:def
    deploy:
      restart_policy:
        condition: on-failure
    environment:
      - NODO=node3
      - ZOOKEEPER_HOST:"zookeeper:2181"
    depends_on:
      - zookeeper
    networks:
      - webnet
networks:
  webnet:
```