docker swarm 部署 springcloud

前提已经安装好docker swarm集群

在swarm主节点上创建了网络`docker network create --driver overlay springcloud-overlay`

主要注意容器内网络，swarm集群内部可以通过services的name做dns使用

- 标准版的spring cloud项目包含4部分`eureka、gateway、provider、consumer`
- 请求流程：`client请求-->gateway-->consumer(feign)-->provider`
- 在swarm manager节点上，执行`docker stack deploy -c XXXX.yml stack_name` 部署上述四个服务

1. eureka-server配置

- application.yml配置：

```
# 容器环境变量
docker:
  env:
    eureka:
      ## 通过容器指定参数
      host: ${EUREKA_HOST}
      port: ${EUREKA_PORT}
      hostname: ${INSTANCE_NAME:localhost}
eureka:
  instance:
    hostname: ${docker.env.eureka.hostname}
    instance-id: ${eureka.instance.hostname}:${server.port}
  client:
    fetch-registry: true
    register-with-eureka: true
    service-url:
      #defaultZone: http://localhost:8888/eureka
      defaultZone: http://${docker.env.eureka.host}:${docker.env.eureka.
port}/eureka
  server:
    enable-self-preservation: false
```

- 对应EUREKA-HA相互注册的eureka的 stack脚本，springcloud-eureka-ha.yml:

```yaml
version: "3.4"

# eureka
services:
  swarm-eureka1:
    image: xdevp.xiaogj.com:8088/mall/springcloud-swarm-eureka
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
    volumes:
      - /tmp:/tmp
    networks:
      springcloud-overlay:
        aliases:
          # 该网络内取一个别名，即DNS，只要在springcloud-overlay网络内就可以通过swarm-eureka:8888可以访问注册中心
          - swarm-eureka
    ports:
      # 对外暴露端口:容器内部端口
      - "9888:8888"
    environment:
      # 注册到swarm-eureka2的注册中心
      - EUREKA_HOST=swarm-eureka2
      # 相互注册，使用容器内的8888端口并非暴露的 9888端口
      - EUREKA_PORT=8888
      # 注册到eureka的实例 ID名称，取当前服务的名称：swarm-eureka1，内网通过服务名称做dns服务发现
      - INSTANCE_NAME=swarm-eureka1
  swarm-eureka2:
    image: xdevp.xiaogj.com:8088/mall/springcloud-swarm-eureka
    volumes:
      - /tmp:/tmp
    networks:
      springcloud-overlay:
        aliases:
          - swarm-eureka
    ports:
      - "9889:8888"
    environment:
      - EUREKA_HOST=swarm-eureka1
      - EUREKA_PORT=8888
```

```
      - INSTANCE_NAME=swarm-eureka2
# 需要先创建网络：docker network create --driver overlay springcloud-overl
ay
networks:
  springcloud-overlay:
    external: true
```

执行`docker stack deploy -c springcloud-eureka-ha.yml springcloud-eureka`

2. eureka client端配置，gateway、provider、consumer端的 eureka配置都一样

```
# docker 环境变量设置
docker:
  env:
    eureka:
      host: ${EUREKA_HOST:localhost}
      port: ${EUREKA_PORT:8888}
      username: ${EUREKA_USERNAME}
      password: ${EUREKA_PASSWORD}
      hostname: ${INSTANCE_NAME:localhost}
      fetch: ${EUREKA_FETCH:true}
      register: ${EUREKA_REGISTER:true}
      # 版本号支持
      metadata:
        version: ${SERVICE_VERSION:v1}
eureka:
  client:
    register-with-eureka: ${docker.env.eureka.register}
    fetch-registry: ${docker.env.eureka.fetch}
    service-url:
      #defaultZone: http://localhost:8888/eureka
      ## 通过容器指定参数
      defaultZone: http://${docker.env.eureka.host}:${docker.env.eureka.
port}/eureka
      #defaultZone: http://${docker.env.eureka.username}:${docker.env.eu
reka.password}@${docker.eureka.host}:${docker.eureka.port}/eureka
  instance:
    # 使用docker 编排里面的service name，即定义的service名称
    hostname: ${docker.env.eureka.hostname}
    ## 元数据
    metadata-map:
      version: ${docker.env.eureka.metadata.version}
```

```
      #weight: 10
    # 使用ip注册
    #prefer-ip-address: true
    #instance-id: ${spring.cloud.client.ip-address}:${server.port}
    instance-id: ${eureka.instance.hostname}:${server.port}
    ## 心跳和续约
    lease-expiration-duration-in-seconds: 60
    lease-renewal-interval-in-seconds: 10
```

3. 部署脚本springcloud-gateway.yml

```
version: "3.4"

# gateway
services:
  springncloud-gateway:
    # 私服仓库拉取镜像，前提是集群内所有节点都需要先使用docker login登陆到私服
    image: xdevp.xiaogj.com:8088/mall/springcloud-swarm-gateway
    ports:
      - "9082:9082"
    networks:
      springcloud-overlay:
        aliases:
          - springncloud-gateway
    ## 内部eureka instance_id 使用SERVICE_NAME变量，变量的值为 service的名称，docker集群内部使用dns
    environment:
      # 使用注册中心定义的别名为注册中心的host
      - EUREKA_HOST=swarm-eureka
      - EUREKA_PORT=8888
      #  划重点：使用service name定义的名称作为instance_id，同集群网络内通过service name做dns相互调用
      - INSTANCE_NAME=springncloud-gateway
    # 该部分仅 stack deploy的时候有效
    deploy:
      replicas: 1
      # 重启策略
      restart_policy:
        condition: on-failure
      # 资源限制
      resources:
        limits:
```

```
          cpus: '1'
          memory: 1024M
        reservations:
          cpus: '0.2'
          memory: 150M
    volumes:
      - /tmp:/tmp

# 需要先创建网络：docker network create --driver overlay springcloud-overl
ay
networks:
  springcloud-overlay:
    external: true
```

执行docker stack deploy -c springcloud-gateway.yml springcloud-gateway

4. provider和consumer的stack 部署脚本springcloud-service.yml

```
version: "3.4"

# springcloud service
services:
  springcloud-consumer:
    image: xdevp.xiaogj.com:8088/mall/springcloud-swarm-consumer
    ports:
      - "8087:8087"
    networks:
      springcloud-overlay:
        aliases:
          - springcloud-consumer
    ## 内部eureka instance_id 使用SERVICE_NAME变量，变量的值为 service的名
称，docker集群内部使用dns
    environment:
      - EUREKA_HOST=swarm-eureka
      - EUREKA_PORT=8888
      - INSTANCE_NAME=springcloud-consumer
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
      resources:
        limits:
```

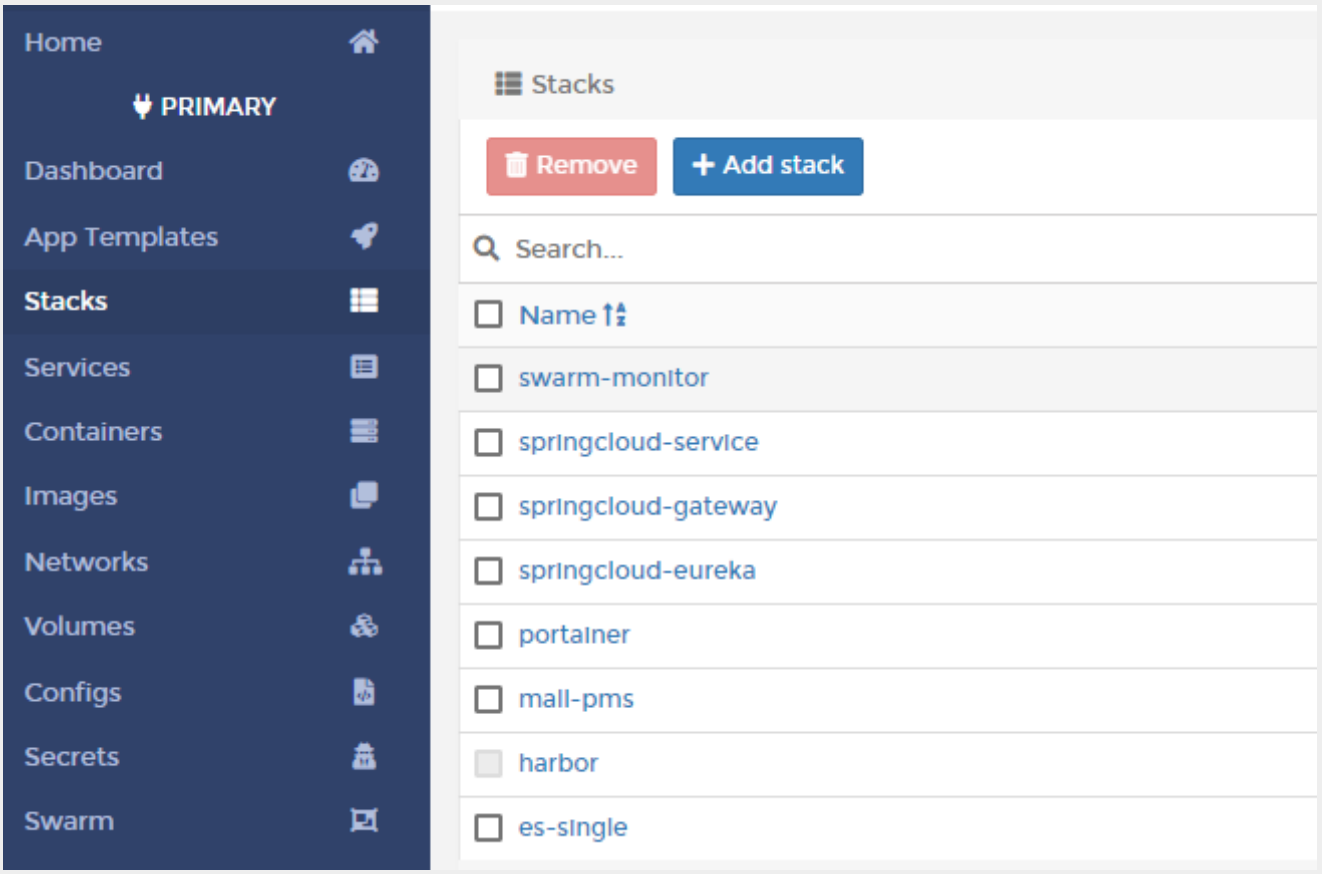
```
          cpus: '0.5'
          memory: 512M
        reservations:
          cpus: '0.2'
          memory: 200M
    volumes:
      - /tmp:/tmp

  springcloud-privoder:
    image: xdevp.xiaogj.com:8088/mall/springcloud-swarm-provider
    ports:
      - "8880:8880"
    networks:
      springcloud-overlay:
        aliases:
          - springcloud-privoder
    ## 内部eureka instance_id 使用SERVICE_NAME变量，变量的值为 service的名称，docker集群内部使用dns，
    # eureka端口使用容器内的8888端口非开放的端口
    environment:
      - EUREKA_HOST=swarm-eureka
      - EUREKA_PORT=8888
      - INSTANCE_NAME=springcloud-privoder
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: '1'
          memory: 512M
        reservations:
          cpus: '0.2'
          memory: 200M
    volumes:
      - /tmp:/tmp
networks:
  springcloud-overlay:
    external:
      name: springcloud-overlay
```

执行`docker stack deploy -c springcloud-service.yml springcloud-service`

5. 最终部署成功效果

```
[root@node1 mall]# docker stack ls
NAME                SERVICES            ORCHESTRATOR
es-single           1                   Swarm
mall-pms            1                   Swarm
portainer           2                   Swarm
springcloud-eureka  2                   Swarm
springcloud-gateway 1                   Swarm
springcloud-service 2                   Swarm
swarm-monitor       2                   Swarm
[root@node1 mall]#
```



6. 镜像制作

   - dockerfile制作镜像脚本

```
#FROM java:8
# 根据容器限制自动计算堆的值,Fabric8社区提供的基础Docker镜像,用50%的可用内存作为上限
FROM fabric8/java-jboss-openjdk8-jdk

VOLUME /tmp

# 参数设置，通过pom打包设置或执行 docker build -t image-name:tag --build-arg JAR_FILE= 设置jar路径
ARG JAR_FILE
#ARG SERVER_PORT
```

```
## 拷贝文件内容
ADD $JAR_FILE /mall-pms.jar
#RUN bash -c 'touch /mall-pms.jar'
#ENV TZ=Asia/Shanghai
#RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/t
imezone
## 开放端口
#EXPOSE ${SERVER_PORT}

# 环境变量，可在stack中通过environment指令设置
ENV JVM_OPTS="-Xmx512m -Xms512m"
ENV JVM_GC_INFO="-XX:+PrintFlagsFinal -XX:+PrintGCDetails"
ENV JVM_UNLOCK="-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLim
itForHeap"

## 容器启动后第一个命令
#ENTRYPOINT ["java","-Xmx512m -Xms512m","-Djava.security.egd=file:/de
v/./urandom","-jar","/mall-pms.jar"]
#ENTRYPOINT [ "sh", "-c", "java -server $JVM_OPTS -Djava.security.egd=fi
le:/dev/./urandom -jar /mall-pms.jar" ]
CMD java -server $JVM_UNLOCK $JVM_OPTS -jar /mall-pms.jar
```

- 使用maven构建打包镜像的pom.xml中容器相关配置，执行`mvn dockerfile:build`：

```
  <build>
    <plugins>
      <finalName>${project.artifactId}</finalName>
      <plugin>
          <groupId>com.spotify</groupId>
          <artifactId>dockerfile-maven-plugin</artifactId>
          <version>1.4.13</version>
          <!-- 绑定maven阶段自动触发dockerfile build操作 -->
          <!--        <executions>-->
          <!--          <execution>-->
          <!--            <id>default</id>-->
          <!--            <phase>package</phase>-->
          <!--            <goals>-->
          <!--              <goal>build</goal>-->
          <!--            </goals>-->
          <!--          </execution>-->
          <!--        </executions>-->
```

```
            <configuration>
              <!-- 上下文配置，设置项目跟路径，读取Dockerfile -->
              <contextDirectory>${project.basedir}</contextDirectory>
              <!-- 使用setting配置账号密码 -->
              <useMavenSettingsForAuth>true</useMavenSettingsForAuth>
              <!--上传路径/镜像构建名： Harbor地址/Harbor项目名/springboot
项目名-->
              <repository>私服镜像仓库/私服仓库项目/${project.artifactId}
</repository>
              <!--指定tag -->
              <tag>镜像tag(latest)</tag>
              <buildArgs>
                <!--指定参数jar-->
                <JAR_FILE>target/${project.build.finalName}.jar</JAR_FIL
E>
                <!-- <SERVER_PORT>8888</SERVER_PORT> -->
              </buildArgs>
            </configuration>
          </plugin>
      </plugins>
   </build>
```