Author：191820133 陆昱宽

迭代三在CICD方面没有改变，因此Jenkins之类我也不介绍了。 为了提升集群的可用性，我们组用了Docker Swarm作为容器编排工具，搭建了Swarm集群， 集群内采用Overlay2网络，基于三层网络来构建虚拟的二层网络，使容器可以在虚拟的链路层进行通信

< 本来我想做k8s，奈何时间不够了，只能搭一个Swarm来凑数QAQ >

**Swarm介绍**

Docker Swarm是一个Docker容器集群编排方案。 和K8S一样，可以实现服务编排，服务扩缩容，负载均衡，服务发现，服务自愈等等功能。

Docker 提供了多种容器网络，其中被Docker Swarm默认采用的是Ovrlay2，它可以实现在单个容器网络内包含多个物理主机，各个处于不同物理网络的容器可以用**内网地址或者容器名来**来和其他容器通信

**Swarm服务**

Swarm使用Master-Slave模式，Master管理服务，Worker运行服务。

Swarm中的“服务”，其实就是逻辑上作为一整个实体的一个或一组容器，Master根据容器镜像来构建服务，则服务就会自动部署到所有的Worker节点

**服务发布**

Swarm支持两种服务发布模式：Host和Ingress，默认采用Ingress

简而言之，Ingress模式就和K8S中的NodePort一样， 其核心概念是： 服务可以向集群外界发布端口，供外接界访问，**外部流量对集群的任意节点（哪怕该节点没有运行该服务）上的该端口的访问都会被路由到运行服务副本的节点，**这就实现了服务发现和负载均衡

**Swarm集群**

**配置**

|  |  |  |
| --- | --- | --- |
| 主机名 | 主机ip | 角色 |
| lyk阿里云服务器 | 123.56.20.222 | master，CICD工作节点 |
| lyk华为云服务器 | 121.36.247.134 | master |
| lyk腾讯云服务器 | 124.222.135.47 | master |

3台Master， 3台Worker（ Master也同时作为Worker，因此实际上只有三台主机 ）

由于我们组的前端之前运行在一个1C2G的服务器，访问起来非常慢，因此首先是对前端进行扩容，我们的三个节点都运行了前端服务/容器:

**准备**

前置准备：所有节点必须打开：

* UDP/4789： 绑定到VTE
* TCP/2377： Swarm的集群管理默认使用2377端口
* TCP/7946, UDP/7946: Swarm的节点发现使用7946端口

**步骤**

1. 先在阿里云主机上创建第一个master节点：

docker swarm init --advertise-addr 123.56.20.222:2377

1. 生成节点作为master加入集群的令牌：

docker swarm join-token manager

1. 在其他节点上使用上述令牌，使其作为master加入该swarm集群。 成功后执行下述命令，查看集群中的节点：

docker node ls  ✔  1m 10s  lyk@lyk-Ali

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

awxgef7wqwl0uty7q5olo5btl k8s-master Ready Active Reachable 20.10.12

rhm2u9w27d8z1n4f7y99t1mwr k8s-node-lyk-tx Ready Active Reachable 20.10.5

ap2gvi9h3u4h7544y2c8b35vj \* lyk-Ali Ready Active Leader 20.10.7

1. 在任意master上创建overlay网络，名为`volatile`:

docker network create -d overlay volatile

1. 在master上基于镜像创建新服务`volatile\_frontend\_svc`，并使用网络`volatile`:

docker service create --name volatile\_frontend\_svc --network volatile -p 80:80 --replicas 3 lyklove/volatile\_frontend:latest

* 这里设置服务实例数为3
* 我们将集群的80端口映射到了容器的80端口。 因此访问集群的任意节点的80端口的流量最终都会被抓发到运行了该服务副本的节点

1. （后续）滚动更新:

docker service update --image lyklove/volatile\_frontend:new --update-parallelism 2 --update-delay 1s volatile\_frontend\_svc

* 基于新镜像`lyklove/volatile\_frontend:new `更新服务，并在其他节点上也进行服务更新
* 该命令可以在任意拥有该新镜像的master节点上执行

**集成CICD**

可以发现，CICD只需要将服务打包成镜像，然后利用该镜像滚动更新就行了

Jenkins脚本`jenkinsfile.groovy`:

// ...

// 根据代码构建新镜像

stage("update service by built image"){

sh "docker service update --image ${IMAGE\_TO\_RUN} --update-parallelism 2 --update-delay 2s ${SERVICE\_NAME}"

}

**集群使用**

通过`[host-ip]:80`访问前端

其中`host-ip`可以是集群中任意节点的ip