**原创；微信公众号：千里行走；**

**受限图片大小限制，有些图片不是很清晰，可以到微信公众号查看；**

提供helm和yaml两种部署方式。

Helm部署步骤详见笔者git地址：

<https://github.com/hepyu/k8s-app-config/tree/master/helm/min-cluster-allinone/es-min>

yaml部署步骤详见笔者git地址：

<https://github.com/hepyu/k8s-app-config/tree/master/yaml/min-cluster-allinone/es-min>

**目录**

**(1).容器化组件**

**(2).helm容器化方式**

**1.镜像准备**

**2.部署存储卷**

**3.helm部署master node**

**4.helm部署ingest node**

**5.helm部署data node**

**(3).yaml容器化方式**

**(4).yaml配置文件重点详解**

**1.pod亲和性**

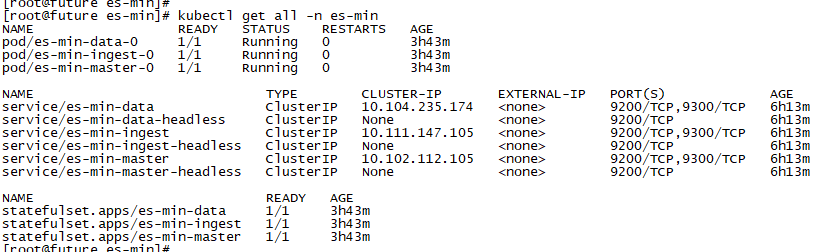
**2.pod优雅关闭**

**3.pvc保护设置**

**正文**

**(1).容器化组件**

容器化成功后的组件，因为是demo，所以replica=1，笔者的demo机器配置不高，需要节约。



**(2).helm容器化方式**

**1.镜像准备**

docker pull elasticsearch:6.4.3

重命名镜像为：docker.elastic.co/elasticsearch/elasticsearch:6.4.3

docker images |grep elasticsearch |awk '{print "docker tag ",$1":"$2,$1":"$2}' |sed -e 's#elasticsearch#docker.elastic.co/elasticsearch/elasticsearch#2' |sh -x

增加helm仓库:

Add the elastic helm charts repo:

helm repo add elastic <https://helm.elastic.co>

**2.部署存储卷**

如下地址有存储卷的yaml配置，依次执行：

<https://github.com/hepyu/k8s-app-config/tree/master/helm/min-cluster-allinone/es-min>

kubectl apply -f es-min-data0-pv-local.yaml

kubectl apply -f es-min-data-storageclass-local.yaml

kubectl apply -f es-min-ingest0-pv-local.yaml

kubectl apply -f es-min-ingest-storageclass-local.yaml

kubectl apply -f es-min-master0-pv-local.yaml

kubectl apply -f es-min-master-storageclass-local.yaml

**3.helm部署master node**

helm install --name es-min-master --namespace es-min elastic/elasticsearch --version 6.4.3 --set masterService=es-min-master,nodeGroup=master,clusterName=es-min,roles.data=false,roles.ingest=false,volumeClaimTemplate.resources.requests.storage=1Gi,volumeClaimTemplate.storageClassName=es-min-master-pv-local,volumeClaimTemplate.accessModes[0]=ReadWriteOnce,replicas=1,minimumMasterNodes=1

--name：指定安装的helm套件的名称

--namespace：要部署到容器中哪个namespace

--version 6.4.3：指定要部署的es版本

--set：指定容器化的详细参数

masterService：指定es在k8s容器内部的service负载均衡的名字；代理后端的POD。

nodeGroup：因为master生产环境是不说多个节点，所以对这些节点分组，指定这个分组名字；

clusterName：指定elasticsearch集群的名字；

roles.data：指定容器是否开启data功能；

roles.ingest：指定容器是否开启ingest功能；

volumeClaimTemplate.resources.requests.storage：指定存储空间大小；

volumeClaimTemplate.storageClassName：指定存储空间的storageClassName；

volumeClaimTemplate.accessModes[0]：指定存储空间的访问模式，ReadWriteOnce表示PV只能被一个POD读写；

replicas：标识创建几个POD。

minimumMasterNodes：标识elasticsearch集群最少需要几个存活的master；

**4.helm部署ingest node**

helm install --name es-min-ingest --namespace es-min elastic/elasticsearch --version 6.4.3 --set masterService=es-min-master,nodeGroup=ingest,clusterName=es-min,roles.data=false,roles.master=false,volumeClaimTemplate.resources.requests.storage=1Gi,volumeClaimTemplate.storageClassName=es-min-ingest-pv-local,volumeClaimTemplate.accessModes[0]=ReadWriteOnce,replicas=1,minimumMasterNodes=1

**5.helm部署data node**

helm install --name es-min-data --namespace es-min elastic/elasticsearch --version 6.4.3 --set masterService=es-min-master,nodeGroup=data,clusterName=es-min,roles.master=false,roles.ingest=false,volumeClaimTemplate.resources.requests.storage=1Gi,volumeClaimTemplate.storageClassName=es-min-data-pv-local,volumeClaimTemplate.accessModes[0]=ReadWriteOnce,replicas=1,minimumMasterNodes=1

**(3).yaml容器化方式**

笔者github提供elasticsearch的yaml配置文件：

<https://github.com/hepyu/k8s-app-config/tree/master/yaml/min-cluster-allinone/es-min>

提供了一个deploy.sh，可以直接运行sh deploy.sh完成容器化；

或者依次执行命令：

kubectl apply -f es-min-data-storageclass-local.yaml

kubectl apply -f es-min-ingest-storageclass-local.yaml

kubectl apply -f es-min-master-storageclass-local.yaml

kubectl apply -f es-min-data0-pv-local.yaml

kubectl apply -f es-min-ingest0-pv-local.yaml

kubectl apply -f es-min-master0-pv-local.yaml

kubectl apply -f es-min-data-statefulset.yaml

kubectl apply -f es-min-ingest-statefulset.yaml

kubectl apply -f es-min-master-statefulset.yaml

kubectl apply -f es-min-pvc.yaml

kubectl apply -f es-min-service.yaml

**(4).yaml配置文件重点详解**

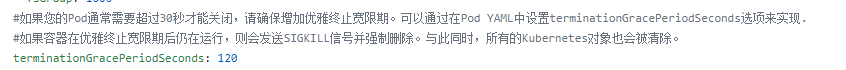
笔者的yaml配置文件中做了详细注释，可以直接进入github去阅读相关yaml配置文件，这里只罗列其中的重点：

**1.pod亲和性**

作为elasticsearch集群，data/ingest/master节点都是多个，相同类型的节点如master是不能工程拓扑到相同的workNode上的，防止一台workNode宕机后将部分数据不能访问(部分数据的primary和复本恰好都在这台宕机的workNode上)。



**2.pod优雅关闭**



**3.pvc保护设置**

防止误删除了PVC，如果发现PVC还有POD使用，即使执行了delete pvc的操作，依然不会删除，只有当pod也被杀掉后才会删除掉pvc。

