**Spark部署**

spark版本：3.2.3 [下载网址](http://mirrors.gigenet.com/apache/spark/spark-3.2.3/)

**spark镜像搭建**

两个配置文件保存在/root/spark/images-building目录下，保存为Dockerfile.executor和Dockerfile.driver，执行命令：

|  |
| --- |
| Bash docker build -f Dockerfile.executor -t awayee/spark-k8s-base-new:v3.2.3 . docker push awayee/spark-k8s-base-new:v3.2.3 # push到docker仓库，需提前执行docker login docker build -f Dockerfile.driver -t awayee/spark-k8s-driver-new:v3.2.3 . docker push awayee/spark-k8s-driver-new:v3.2.3 |

**Spark UI & History Server搭建:(存在不显示job&task的问题)**

两者的相关配置文件都保存在/root/spark/UI&HS目录下，执行命令：

|  |
| --- |
| Bash kubectl create -f namespace-spark-cluster.yaml  #新建名为spark-cluster的namespace  kubectl create -f spark-master-controller.yaml —namespace=spark-cluster # 暴露master的7077和8088端口给集群使用（相关设置写于.yaml文件中）  kubectl create -f spark-master-service.yaml —namespace=spark-cluster # master主体构建  kubectl create -f spark-worker-controller.yaml —namespace=spark-cluster # work主体构建  kubectl create -f spark-ui-proxy-controller.yaml —namespace=spark-cluster # 部署代理，方便访问集群的master和worker(因为集群内部master和worker随机分配， # 这样可以形成黑盒，使用者不用关心具体角色)，并暴露代理的80端口 |

以上是spark UI的相关部署。

|  |
| --- |
| Bash kubectl create -f spark-history-server.yaml # 外部访问端口：32088，相关记录文件存在file:///tmp（存疑，网上大都用的hdfs://） |

以上是History Server的相关部署

**Spark用户权限设置**

相关配置文件存在/root/spark/user-auth目录下

|  |
| --- |
| Bash kubectl create serviceaccount spark -n spark-cluster # 创建名为spark的sa，放在spark-cluster命名空间下  kubectl apply -f create\_clusterrole.yaml # 定义一种权限，可对.yaml所设定的resources进行verbs下包含的操作，此处为读写权限  kubectl apply -f create\_ClusterRoleBinding.yaml # 赋予权限给spark用户 |

注意：若使用/opt/spark/bin/spark-submit指令，应在指令后添加：

|  |
| --- |
| Bash ##以下在spark-submit中添加 --conf spark.kubernetes.namespace=spark-cluster \ --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark |

**Pi计算样例测试**

测试前进入到master的pod中，指令如下：

|  |
| --- |
| Bash kubectl exec -it <pod-name> -n <name-space> /bin/bash  # kubectl get pods --all-namespaces 查看所有命名空间所属的pods # kubectl get pods -n spark-cluster 指定命名空间，不写-n会进入default命名空间 |

添加全局变量（暂未设置——可直接spark-shell或spark-submit，而不用/opt/spark/bin/spark-submit）

|  |
| --- |
| Bash cd /etc/profile  export SPARK\_HOME=/opt/spark export PATH=$SPARK\_HOME/bin:$PATH #在/etc/profile中添加上述两行  source profile |

本地测试（即在一台机器上跑，不放入集群中运行）

|  |
| --- |
| Bash /opt/spark/bin/spark-submit \ --class org.apache.spark.examples.SparkPi \ --master local /opt/spark/examples/jars/spark-examples\_2.12-3.2.3.jar |

Cluster mode测试

|  |
| --- |
| Bash /opt/spark/bin/spark-submit \  --master k8s://https://10.176.62.221:6443 \  # --master后内容可由kubectl cluster-info获取  --deploy-mode cluster \  --name spark-pi \  --class org.apache.spark.examples.SparkPi \  --conf spark.kubernetes.namespace=spark-cluster \  --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \  --conf spark.executor.instances=2 \  # instance:设定每个job的executor运行个数  --conf spark.kubernetes.container.image=awayee/spark-k8s-driver-new:v3.2.3 \  --conf spark.eventLog.enabled=true \  --conf spark.eventLog.dir=file:///tmp \  local:///opt/spark/examples/jars/spark-examples\_2.12-3.2.3.jar |

*注：cluster-mode下查看结果需使用 kubectl logs -f <long-name> -n spark-cluster查看对应日志*

**Wordcount相关测试**

以下测试python I/O的程序，以验证pvc下共享文件夹的读写

（本机是/nfsdata1，pod内由spark-submit --conf控制，具体见下）

|  |
| --- |
| Python # from \_\_future\_\_ import print\_function  # import sys # from operator import add  # from pyspark.sql import SparkSession  if \_\_name\_\_ == "\_\_main\_\_":  # if len(sys.argv) != 2:  # print("Usage: wordcount <file>", file=sys.stderr)  # sys.exit(-1)   # spark = SparkSession\  # .builder\  # .appName("PythonWordCount")\  # .getOrCreate()  # # print("————————",sys.argv[1])  fin = open("/mydata/index.txt","r")  print(fin.readline())  # lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])  # counts = lines.flatMap(lambda x: x.split(' ')) \  # .map(lambda x: (x, 1)) \  # .reduceByKey(add)  # output = counts.collect()  # for (word, count) in output:  # # print("%s: %i" % (word, count))  # fo.write("%s: %i" % (word, count))  fo = open("/mydata/wcout.txt","a")  fo.write(" (word, count)")  fo.close()  fin.close()  # spark.stop() |

|  |
| --- |
| Bash /opt/spark/bin/spark-submit \  --master k8s://https://10.176.62.221:6443 \  --deploy-mode cluster \  --conf spark.executor.instances=3 \  --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \  --conf spark.kubernetes.namespace=spark-cluster \  --conf spark.kubernetes.driver.pod.name=spark-wctest-1 \  --conf spark.kubernetes.container.image=awayee/spark-k8s-driver-new:v3.2.3 \  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.path=/mydata \  # 上句设定新的pod的共享目录  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.readOnly=false \  # 上句取消只读设定  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.options.claimName=mypvc2 \  # 上句绑定pvc  --name spark\_wc\_1 \ local:///mydata/wc.py |

下面是最新能跑通的wc的测试：

|  |
| --- |
| Python from \_\_future\_\_ import print\_function  import sys from operator import add  from pyspark.sql import SparkSession  if \_\_name\_\_ == "\_\_main\_\_":  if len(sys.argv) != 2:  print("Usage: wordcount <file>", file=sys.stderr)  sys.exit(-1)   spark = SparkSession\  .builder\  .appName("PythonWordCount")\  .getOrCreate()    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])  counts = lines.flatMap(lambda x: x.split(' ')) \  .map(lambda x: (x, 1)) \  .reduceByKey(add)  output = counts.collect()  fo = open("/mydata/wcout.txt","a")  for (word, count) in output:  fo.write("%s: %i" % (word, count))  fo.close()  spark.stop() |

|  |
| --- |
| Bash /opt/spark/bin/spark-submit \  --master k8s://https://10.176.62.221:6443 \  --deploy-mode cluster \  --conf spark.executor.instances=5 \  --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \  --conf spark.kubernetes.namespace=spark-cluster \  --conf spark.kubernetes.driver.pod.name=spark-wctest-1 \  --conf spark.kubernetes.container.image=awayee/spark-k8s-driver-new:v3.2.3 \  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.path=/mydata \  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.mount.readOnly=false \  --conf spark.kubernetes.driver.volumes.persistentVolumeClaim.data.options.claimName=mypvc2 \  ###### add ↓  --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.path=/mydata \  --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.mount.readOnly=false \  --conf spark.kubernetes.executor.volumes.persistentVolumeClaim.data.options.claimName=mypvc2 \  ###### add ↑  --name spark\_wc\_1 \ local:///mydata/wc.py \ /mydata/test.txt |

新的spark-submit指令添加了executor指令，这样就可以跑通了，原因是第19行是在任务的executor角色的pod中找路径，所以也要给executor挂载。

再次解释，18行的local是绑定该任务的driver角色的pod路径

此外第19行使用local://会报错，系统会表示它无法识别local

**相关指令合集**

便于查看QwQ

|  |
| --- |
| Bash docker images # 查看本地镜像  docker rm -v $(docker ps -a -q) # 删除所有已停止的容器  kubectl get pods --all-namespaces  #获取所有命名空间的pods  kubectl delete po --all -n custom-namespace  #删除命名空间中的所有pod，但保留命名空间  kubectl delete ns custom-namespace  #通过删除整个命名空间来删除pod  kubectl delete po ` kubectl get po | grep ContainerStatusUnknown | awk '{print $1}' ` # 自定义批量删除 kubectl get pods -n kube-sample | grep Evicted | awk '{print$1}'| xargs kubectl delete -n kube-sample pods # 自定义批量删除  kubectl describe pods -n spark-cluster # 查看pod详细日志  kubectl exec -it <pod-name> -n <name-space> /bin/bash  #进入pod  # 以下两天cp命令在本机（pod外）运行 kubectl cp /root/spark-3.0.0-bin-hadoop3.2/examples/src/main/python/pi.py \  -n spark-cluster spark-master-controller-f2r6f:/root/cptest/pi\_cp\_test.py # 将本机文件拷贝到pod # 将/root/spark-3.0.0-bin-hadoop3.2/examples/src/main/python/pi.py拷贝到spark-cluster命名空间下 # spark-master-controller-f2r6f这一pod的/root/cptest路径下，并将新文件命名为pi\_cp\_test.py  kubectl cp -n spark-cluster spark-master-controller-f2r6f:/root/cptest/pi\_cp\_test.py /root/spark/pi\_cp\_test.py # 将pod文件拷贝到本机，大致相同，注意顺序即可 |

**一些主要参考**

[镜像搭建参考](https://www.oak-tree.tech/blog/spark-kubernetes-primer)

[Spark UI部署参考](https://blog.csdn.net/lixinkuan328/article/details/104528182/)

[Spark History Server部署参考](https://blog.csdn.net/lhyandlwl/article/details/122025937)

[K8S Pod Pending详解](https://blog.csdn.net/xcbeyond/article/details/124580730)

[Cluster-mode v.s Client mode](https://www.freesion.com/article/1927105653/)

[Spark on k8s原理](https://blog.csdn.net/zfw_666666/article/details/127512994)

[k8s授权机制相关（系列）](https://blog.csdn.net/weixin_40228200/article/details/124486725)

**一些次要参考**

[docker仓库](https://hub.docker.com/)

[Dockerfile相关](https://www.runoob.com/docker/docker-dockerfile.html)（菜鸟）

[spark-submit相关拼接](https://blog.csdn.net/qq_39698985/article/details/127824962)

[spark-shell&spark-submit](https://blog.csdn.net/weixin_45666566/article/details/112724328)

[pending解决方案](https://blog.csdn.net/weixin_44480167/article/details/121417368)

[k8s+spark+submit-1](https://www.cnblogs.com/moonlight-lin/p/13296909.html)