

期末实验一：基于华为云的大数据实时数据分析综合实践

注：本实验的实验环境需以实验一的Hadoop环境和实验三的Zookeeper环境为基础进行搭建，如遇到相关问题可以检查一下实验一和实验三的环境配置是否正确

一、local模式部署安装

1、实验介绍

local模式下的Flink部署安装只需要使用单台机器，仅用本地线程来模拟其程序运行，不需要启动任何进程，适用于软件测试等情况。这种模式下，机器不用更改任何配置，只需要安装JDK 8的运行环境即可。

2、实验目的

- 2.1 实现Flink的安装；
- 2.2 学会Flink的脚本启动；
- 2.3 使用Flink自带的单词统计程序进行测试。

3、实验步骤

3.1 上传安装包并解压

下载flink1.13.6并上传

下载地址：

<https://archive.apache.org/dist/flink/flink-1.13.6/>

由于flink在1.8版本后就不支持hadoop了，所以需要下载额外的包，下载路径如下：

<https://mvnrepository.com/artifact/org.apache.flink/flink-shaded-hadoop-3-uber/3.1.1.7.2.1.0-327-9.0>

<https://mvnrepository.com/artifact/commons-cli/commons-cli/1.4>

上传压缩包到服务器中,进行解压并将文件名改为flink

```
[root@zj-2023140696 ~]# tar -zxvf flink-1.13.6-bin-scala_2.11.tgz
flink-1.13.6/
flink-1.13.6/lib/
flink-1.13.6/lib/flink-shaded-zookeeper-3.4.14.jar
flink-1.13.6/lib/flink-json-1.13.6.jar
flink-1.13.6/lib/log4j-1.2-api-2.17.1.jar
flink-1.13.6/lib/flink-table-blink_2.11-1.13.6.jar
flink-1.13.6/lib/flink-csv-1.13.6.jar
flink-1.13.6/lib/flink-dist_2.11-1.13.6.jar
flink-1.13.6/lib/log4j-slf4j-impl-2.17.1.jar
flink-1.13.6/lib/flink-table_2.11-1.13.6.jar
flink-1.13.6/lib/log4j-core-2.17.1.jar
flink-1.13.6/lib/log4j-api-2.17.1.jar
flink-1.13.6/README.txt
flink-1.13.6/licenses/
flink-1.13.6/licenses/LICENSE.grizzled-slf4j
flink-1.13.6/licenses/LICENSE.javax.activation
flink-1.13.6/licenses/LICENSE.reflections
```

apache-zookeeper-3.9.2-bin.tar.gz	docker336_exp3.zip	flink-1.19.0-bin-scala_2.12.tgz	kafka_2.12-3.5.0.tgz
A.txt	Dockerfile	flink-shaded-hadoop-3-uber-3.1.1.7.2.1.0-327-9.0.jar	MyWordCount.jar
B.txt	Dockerfile-old	flink-shaded-hadoop-uber-3.2.3.jar	OpenJDK8U-jdk_aarch64_linux_hotspot_8u252b09.tar
commons-cli-1.4.jar	flink	hadoop.config	scrips
commons-cli-1.7.0.jar	flink1	hbase-2.5.8-bin.tar.gz	yarn-site.xml
config	flink-1.12.5-bin-scala_2.12.tgz	hbase.config	zookeeper_config
config.yaml	flink-1.13.6-bin-scala_2.11.tgz	hdfs-site.xml	
core-site.xml	flink-1.19.0	kafka_2.12-3.5.0	

3.2 配置全局环境变量

在 .bashrc 配置文件中添加flink路径如下。

```
1 root@master:~# vim .bashrc
2
3
4 #FLINK
5 export FLINK_HOME=/root/flink
   PATH=$FLINK_HOME/bin:$PATH
```

配置完.bashrc后再执行 `source .bashrc`，让其生效

3.3 脚本启动Flink进程

配置好环境变量后就可以全局使用Flink的启动命令，直接以下命令。

注：在实验一中将node01节点hosts文件中127.0.0.1地址注释掉的同学在这一部分需要取消node01节点中hosts文件的注释，保证localhost能被node01找到。

```
1 root@master:~# start-cluster.sh
```

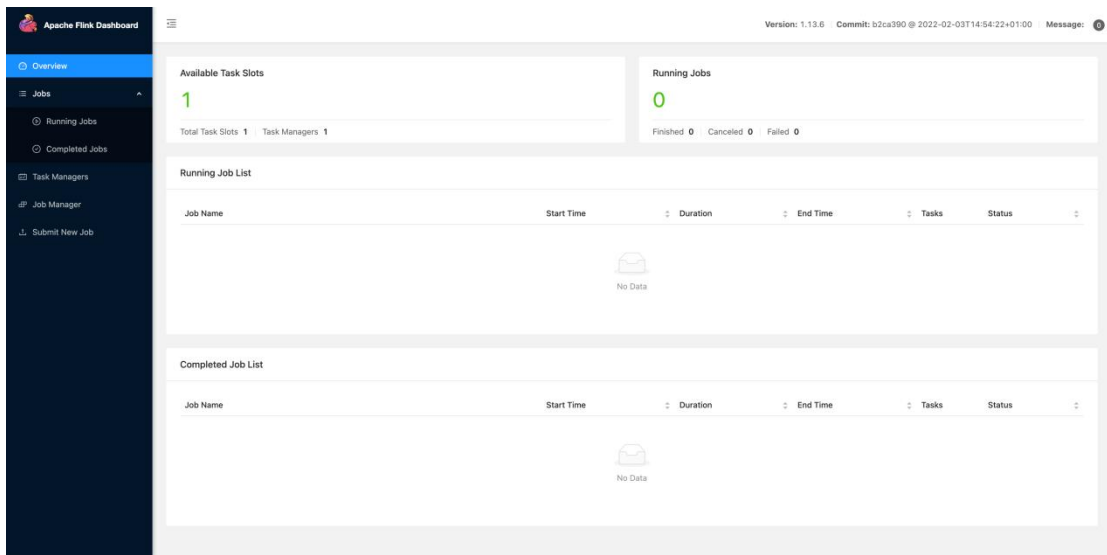
```
root@master:~# start-cluster.sh
Starting cluster.
Starting standalone session daemon on host master.
Starting taskexecutor daemon on host master.
```

```
root@master:~# jps
16336 TaskManagerRunner
17931 Jps
16031 StandaloneSessionClusterEntrypoint
```

3.4 Web界面访问

成功启动两个进程后，访问8081端口即可访问Flink的Web管理界面。（其中node01替换为对应公网IP）

<http://node01:8081/#/overview>



3.5 运行Flink自带的测试用例

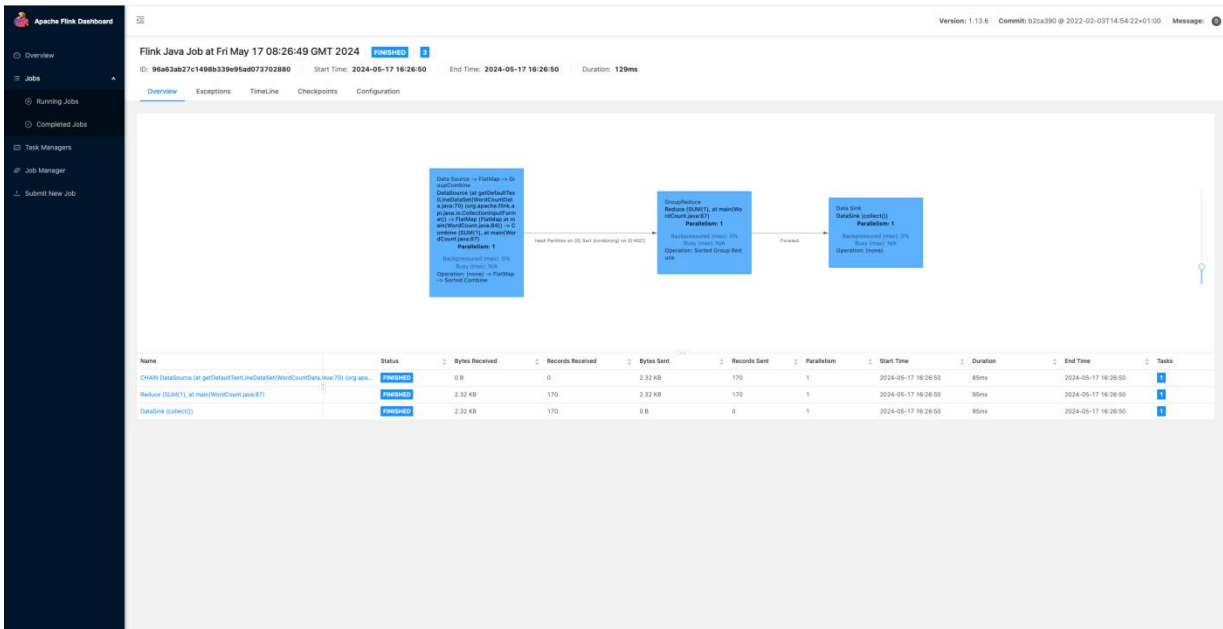
```
1 root@master:~# cd /home/flink
2 root@master:~# bin/flink run
  examples/batch/wordCount.jar
```

```
root@master:~/flink# bin/flink run examples/batch/WordCount.jar --hostname localhost --port 9000
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
Job has been submitted with JobID 84ea8b58926bd3c6eb256ac28bb91197
Program execution finished
Job with JobID 84ea8b58926bd3c6eb256ac28bb91197 has finished.
Job Runtime: 208 ms
Accumulator Results:
- 847a4c381561d6a0ac122f96ba38da40 (java.util.ArrayList) [170 elements]

(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
(and,12)
(arms,1)
(arrows,1)
(awry,1)
(ay,1)
(bare,1)
(be,4)
(bear,3)
(bodkin,1)
(bourn,1)
(but,1)
(by,2)
(calamity,1)
(cast,1)
(coil,1)
(come,1)
(continues,1)
```

3.6 查看统计结果

在Flink的Web管理界面进入Completed Jobs目录下，选择刚刚上交的任务，得到如下页面。



关闭local模式。

```
1 root@master:~# stop-cluster.sh
```

```
root@master:~# stop-cluster.sh
Stopping taskexecutor daemon (pid: 16336) on host master.
Stopping standalone-session daemon (pid: 16031) on host master.
```

二、standalone模式部署安装

1、实验介绍

使用standalone模式需要启动Flink的主节点JobManager以及从节点的TaskManager，具体的任务进程划分见下表。

服务及IP	node01	node02	node03	node04
JobManager	是	否	否	否
TaskManager	是	是	是	是

2、实验目的

实现standalone模式下Flink进程的启动。

3、实验步骤

3.1 修改配置文件

停止node01服务器local模式下的进程后，修改配置文件。
在node01节点上执行以下命令，更改Flink配置文件。

```
1 root@master:~# cd flink/conf
2 root@master:~# vim flink-conf.yaml
```

指定JobManager所在的服务器为master。

```
jobmanager.rpc.address: master
```

```
#####  
# Common  
#####  
# The external address of the host on which the JobManager runs and can be  
# reached by the TaskManagers and any clients which want to connect. This setting  
# is only used in Standalone mode and may be overwritten on the JobManager side  
# by specifying the --host <hostname> parameter of the bin/jobmanager.sh executable.  
# In high availability mode, if you use the bin/start-cluster.sh script and setup  
# the conf/masters file, this will be taken care of automatically. Yarn/Mesos  
# automatically configure the host name based on the hostname of the node where the  
# JobManager runs.  
  
jobmanager.rpc.address: master  
  
# The RPC port where the JobManager is reachable.  
  
jobmanager.rpc.port: 6123
```

更改workers配置文件。

```
1 root@master:~# vim workers
```

```
master  
slave1  
slave2  
slave3
```

3.2 分发配置文件

将改好的文件分发给其他节点。docker的分发要先到宿主机中再进行分发

```
[root@zj-2023140696 ~]# docker cp flink slave1:/root  
Successfully copied 354MB to slave1:/root  
[root@zj-2023140696 ~]# docker cp flink slave2:/root  
Successfully copied 354MB to slave2:/root  
[root@zj-2023140696 ~]# docker cp flink slave3:/root  
Successfully copied 354MB to slave3:/root
```

3.3 启动Flink集群

在node01上执行与local模式相同的命令启动Flink集群。

注：在实验第一部分中将node01节点hosts文件中127.0.0.1地址注释取消掉了，这一部分需要再次注释掉node01节点hosts文件中127.0.0.1地址，保证本机的TaskManager能够正确启动

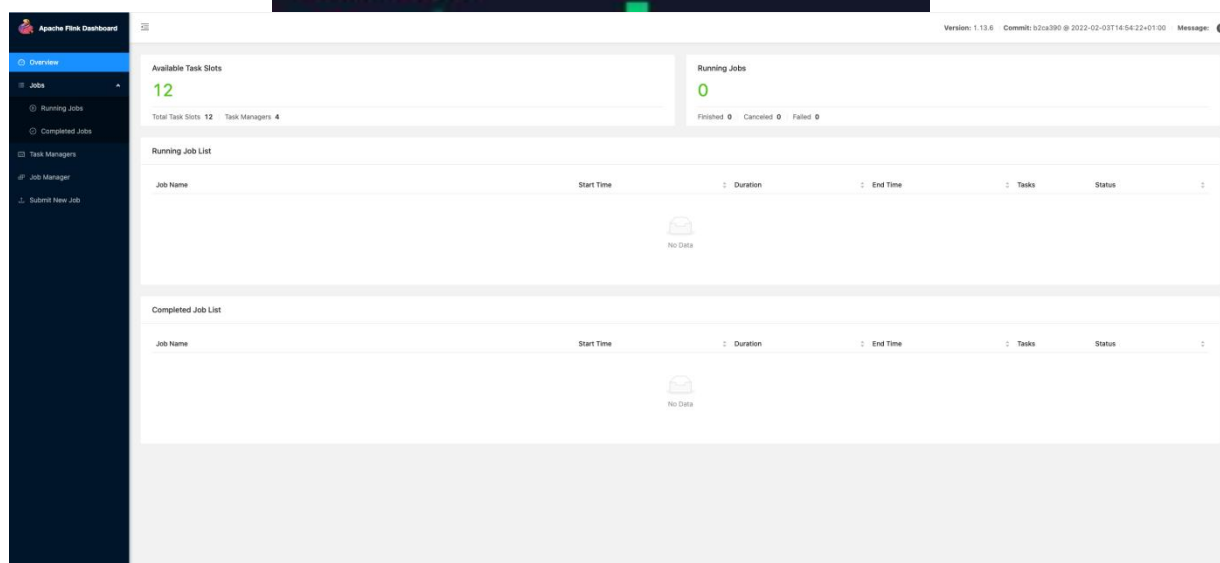
```
1 root@master:~# start-cluster.sh
```

```
root@master:~# start-cluster.sh  
Starting cluster.  
Starting standalone session daemon on host master.  
Starting taskexecutor daemon on host master.  
Starting taskexecutor daemon on host slave1.  
Starting taskexecutor daemon on host slave2.  
Starting taskexecutor daemon on host slave3.
```

进入Web管理页面（网址同第一章）能看到Task Managers和Task Slots数量为4，说明集群正确启动。（这里的task slots可以指定每个taskmanager中包含的数量，后文会提到，这里读者跟着上述步骤并未对其进行设计，故默认1个Task Managers中有1个Task Slots，所以此处应该是4个）

```
root@master:~# jps
19988 Jps
19864 TaskManagerRunner
19551 StandaloneSessionClusterEntrypoint
```

```
root@slave1:~# jps
4289 TaskManagerRunner
1061 QuorumPeerMain
4366 Jps
```



3.4 运行Flink自带测试用例

进行flink自带的测试用例的测试

```
1 root@master:~# bin/flink run examples/batch/wordCount.jar
```

```
root@master:~/flink# bin/flink run examples/batch/WordCount.jar
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
Job has been submitted with JobID 302f00b56e6d18a7f8ed91d240f4e48a
Program execution finished
Job with JobID 302f00b56e6d18a7f8ed91d240f4e48a has finished.
Job Runtime: 494 ms
Accumulator Results:
- b97fe0b7d8481c0ce4b306f37dc64657 (java.util.ArrayList) [170 elements]

(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
(and,12)
(arms,1)
(arrows,1)
```

在Web管理界面的Completed Jobs目录中，可以找到已经完成的任務。下图需要截屏作为得分点

FINISHED 3

Start Time: 2024-05-17 16:47:07

End Time: 2024-05-17 16:47:07

Duration: 49.6ms

[Overview](#) [Exceptions](#) [TimeLine](#) [Checkpoints](#) [Configuration](#)



Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Duration	End Time	Tasks
CHIN Datastore (set getBatchText,mediaSer(WaterCounData.java:70) [org.ap...	FAILED	0 B	0	2.32 kB	170	1	2024-05-17 16:47:07	219ms	2024-05-17 16:47:07	View
Reduce (EMIT1) at Main(WaterCounData.java:87)	FAILED	2.32 kB	170	2.32 kB	170	1	2024-05-17 16:47:07	220ms	2024-05-17 16:47:07	View
DataSink (collect)	FAILED	2.32 kB	170	0 B	0	1	2024-05-17 16:47:07	220ms	2024-05-17 16:47:07	View

三、Flink on Yarn模式

1、实验介绍

Flink任务也可以运行在Yarn上，将Flink任务提交到Yarn平台可以实现统一的任务资源调度管理，方便开发人员管理集群中的CPU和内存等资源。

本模式需要先启动集群，然后再提交作业，接着会向Yarn申请资源空间，之后资源保持不变。如果资源不足，下一个作业就无法提交，需要等到Yarn中的一个作业执行完成后释放资源。

2、实验目的

- ◆ 完成Flink on Yarn模式的配置；
- ◆ 在Yarn中启动Flink集群；
- ◆ 以文件的形式进行任务提交。

3、实验步骤

3.1 修改yarn-site.xml配置文件

将上个实验启动的进程全部关闭。在node01上添加以下配置属性到该文件中进行修改。

```
root@master:~/hadoop# cd etc/
root@master:~/hadoop/etc# cd hadoop/
root@master:~/hadoop/etc/hadoop# ls
capacity-scheduler.xml  hadoop-metrics2.properties  https-log4j.properties  log4j.properties  ssl-client.xml.example  yarn-site.xml
configuration.xml       hadoop-policy.xml           https-site.xml          mapred-env.cmd     ssl-server.xml.example  yarnservice-log4j.properties
container-executor.cfg  hadoop-user-functions.sh.example  kms-acls.xml          mapred-env.sh      user_ec_policies.xml.template  workers
core-site.xml           hdfs-rbf-site.xml          kms-env.sh            mapred-queues.xml.template  yarn-env.cmd
hadoop-env.cmd          hdfs-site.xml              kms-log4j.properties  mapred-site.xml    yarn-env.sh
hadoop-env.sh           https-site.xml              kms-site.xml          shellprofile
root@master:~/hadoop/etc/hadoop#

<?xml version="1.0"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<configuration>
  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>/root/hadoop/tmp/nm-local-dir</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.am.max-attempts</name>
    <value>4</value>
    <description>The maximum number of application master execution attempts.
  </description>
  </property>
  <property>
    <name>yarn.application.classpath</name>
    <value>/root/hadoop/etc/hadoop,/root/hadoop/share/hadoop/common/*,/root/hadoop/share/hadoop/common/lib/*,/root/hadoop/share/hadoop/hdfs/*,/root/hadoop/share/hadoop/hdfs/lib/*,/root/hadoop/share/hadoop/yarn/*,/root/hadoop/share/hadoop/yarn/lib/*,/root/hadoop/share/hadoop/mapreduce/*,/root/hadoop/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

然后将修改后的配置文件分发到其他节点上。之后启动HDFS和Yarn集群，使用start-all.sh

3.2 修改Flink配置文件

首先停止之前的Flink的standalone进程（使用jps命令获取对应进程号，然后直接kill掉对应进程，但要保留Hadoop进程）

在node01上执行以下命令修改FLink的配置文件。

- ◆ 修改flink-conf.yaml配置文件。


```

1 root@master:~# cd /flink/conf/
2 root@master:~# vim flink-conf.yaml 3
4
5
6
7
high-availability: zookeeper
high-availability.storageDir: hdfs://master/flink_yarn_ha high-
availability.zookeeper.path.root: /flink-yarn
high-availability.zookeeper.quorum:
master:2181,slave1:2181,slave2:2181,slave3:2181
yarn.application-attempts: 10
8

```



```

=====
# High Availability
=====

# The high-availability mode. Possible options are 'NONE' or 'zookeeper'.
#
high-availability: zookeeper
yarn.application-attempts: 10
# The path where metadata for master recovery is persisted. While ZooKeeper stores
# the small ground truth for checkpoint and leader election, this location stores
# the larger objects, like persisted dataflow graphs.
#
# Must be a durable file system that is accessible from all nodes
# (like HDFS, S3, Ceph, nfs, ...)
#
high-availability.storageDir: hdfs://master:8020/flink_yarn_ha

# The list of ZooKeeper quorum peers that coordinate the high-availability
# setup. This must be a list of the form:
# "host1:clientPort,host2:clientPort,..." (default clientPort: 2181)
#
high-availability.zookeeper.quorum: master:2181,slave1:2181,slave2:2181,slave3:2181
high-availability.zookeeper.path.root: /flink-yarn

# ACL options are based on https://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html#sc_BuiltinACLschemes
# It can be either "creator" (ZOO_CREATE_ALL_ACL) or "open" (ZOO_OPEN_ACL_UNSAFE)
# The default value is "open" and it can be changed to "creator" if ZK security is enabled
#
# high-availability.zookeeper.client.acl: open

```

修改masters配置文件

```

1 root@master:~# vim masters
2
3
4 zyw-2021140807-0001:8081
zyw-2021140807-0002:8081

```

```
master:8081
slave1:8081
```

在其他四个容器中也进行相同修改

同时将下面下划线的两个jar包分发到每个容器flink的lib中（下载地址在实验手册开头）

```
root@master:~/flink/lib# ls
commons-cli-1.4.jar      flink-json-1.13.6.jar      flink-table-blink_2.11-1.13.6.jar  log4j-api-2.17.1.jar
flink-csv-1.13.6.jar    flink-shaded-hadoop-3-uber-3.1.1.7.2.1.0-327-9.0.jar  flink-table_2.11-1.13.6.jar        log4j-core-2.17.1.jar
flink-dist_2.11-1.13.6.jar  flink-shaded-zookeeper-3.4.14.jar  log4j-1.2-api-2.17.1.jar          log4j-slf4j-impl-2.17.1.jar
```

◆ 在**各个节点**上启动ZooKeeper。

具体启动方法细节请查看实验三，大概就是在已经配置过Zookeeper的前提下，输入`zkServer.sh start`四个节点都要启动！输入启动命令后记得观察是否出现成功启动的提示，确保四个节点全部成功

3.3 在HDFS上创建文件夹

首先确保Hadoop集群已经运行，没有的话输入`start-all.sh`进行启动

输入`hadoop dfsadmin -safemode leave`解除潜在的安全模式

接下来创建文件夹，命令如下。

```
root@master:~# hdfs dfs -mkdir -p /flink_yarn_ha
2024-05-16 14:12:51,557 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
root@master:~# hdfs dfs -ls /
2024-05-16 14:13:10,544 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x - root supergroup          0 2024-05-16 13:17 /flink_yarn_ha
drwxr-xr-x - root supergroup          0 2024-05-16 02:53 /tmp
drwxr-xr-x - root supergroup          0 2024-05-09 22:47 /user
```

3.4 在Yarn中启动Flink集群

在进行下一步的操作前，请大家务必做以下四步检

1. 主节点的/etc/hosts 文件中 127.0.0.1那一行**已经被**注释掉（其余三个节点无所谓），没有被注释掉的话建议重启四台服务器或者重启linux网卡，因为这个文件修改后不会立即生效，重启之后，再查看/etc/hosts 文件时，你可能会发现127.0.0.1那一行又没有被注释掉，不用慌，注释掉该行即可，不需要再次重启，接着做后面的三步即可
2. Hadoop集群已经启动，没有的话输入`start-all.sh`进行启动
3. 已经输入过`hadoop dfsadmin -safemode leave`解除潜在的安全模式
4. 四个节点的Zookeeper已经全部启动成功

四步检查完毕后进行以下操作

在node01上执行以下命令，在Yarn中启动一个全新的Flink集群。可以使用`--help`查看`yarn-session.sh`的参数设置。

```
root@master:~# yarn-session.sh -n 2 -jm 1024 -tm
2024-05-16 14:13:57,294 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: jobmanager.rpc.address, master
2024-05-16 14:13:57,297 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: jobmanager.rpc.port, 6123
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: jobmanager.memory.process.size, 1690m
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: taskmanager.memory.process.size, 1728m
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: taskmanager.numberOfTaskSlots, 1
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: parallelism.default, 1
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: high-availability, zookeeper
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: yarn.application-attempts, 10
2024-05-16 14:13:57,298 INFO org.apache.flink.configuration.GlobalConfiguration [] - Loading configuration property: high-availability.storageDir, hdfs://master:8020/fl
tnk_yarn_ha
2024-05-16 14:14:08,136 INFO org.apache.flink.shaded.curator4.org.apache.curator.framework.state.ConnectionStateManager [] - State change: CONNECTED
2024-05-16 14:14:08,372 INFO org.apache.flink.runtime.leaderretrieval.DefaultLeaderRetrievalService [] - Starting DefaultLeaderRetrievalService with ZookeeperLeaderRetrievalDrive
r(retrievalPath=/leader/rest_server_lock')
JobManager Web Interface: http://master:40107
```

3.5 查看Yarn管理界面

下图需要截屏作为得分点

3.6 提交任务

```

root@master:~/flink# bin/flink run examples/batch/WordCount.jar
2024-05-17 08:54:45,230 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli
2024-05-17 08:54:45,230 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
2024-05-17 08:54:45,677 WARN org.apache.flink.yarn.configuration.YarnLogConfigurationUtil
nfiguration file.
2024-05-17 08:54:45,730 INFO org.apache.hadoop.yarn.client.RMProxy
2024-05-17 08:54:45,850 INFO org.apache.flink.yarn.YarnClusterDescriptor
2024-05-17 08:54:45,933 INFO org.apache.flink.yarn.YarnClusterDescriptor
Job has been submitted with JobID cc91fae13a1a39327359ac26958c135

[] - Found Yarn properties file under /tmp/.yarn-properties-root.
[] - Found Yarn properties file under /tmp/.yarn-properties-root.


[] - The configuration directory ('/root/.flink/conf') already contains a LOG4J config file.If you want to use logback, then please delete or rename the log co
[] - Connecting to ResourceManager at master/172.17.0.2:8032
[] - No path for the flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
[] - Found Web Interface slave3:43497 of application 'application_1715935927195_0001'.

Accumulator Results:
- 2b4b9160975c71fad59b84bada2037fb (java.util.ArrayList) [170 elements]

(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
(and,12)
(arms,1)
(arrows,1)
(awry,1)
(ay,1)
(bare,1)
(be,4)
(bear,3)
(bodkin,1)
(bourn,1)
(but,1)
(by,2)
(calamity,1)
(cast,1)
(coil,1)
(come,1)
(conscience,1)
(consummation,1)
(contumely,1)
(country,1)
(cowards,1)
(currents,1)
(d,4)
(death,2)
(delay,1)

```

在yarn页面中有对应的任务



All Applications

Lagged in as: dr:who

Metrics																											
Jobs Submitted		0		Apps Pending		1		Apps Running		0		Apps Completed		1		Containers Running		Used Resources		Total Resources		Reserved Resources		Physical Mem Used %		Physical VCores Used %	
																		<memory:2 GB, vCores:1>		<memory:32 GB, vCores:32>		<memory:0 B, vCores:0>		31		0	
Nodes Metrics																											
Active Nodes		0		Decommissioning Nodes				Decommissioned Nodes		0		Lost Nodes		0		Unhealthy Nodes		0		Rebooted Nodes		0		Shutdown Nodes			
Job Metrics																											
Scheduler Type		y Scheduler		Scheduling Resource Type		[memory-mb (unit-MB), vcores]		Minimum Allocation		<memory:1024, vCores:1>		Maximum Allocation		<memory:8192, vCores:4>		Maximum Cluster Application Priority		0		Scheduler Busy %		0					
Jobs																											
Search:																											
ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes				
Job: 1715935927195_0001	root	Flink session cluster	Apache Flink		default	0	Fri May 17 16:53:54 +0800 2024	Fri May 17 16:53:55 +0800 2024	N/A	RUNNING	UNDEFINED	1	1	2048	-1	0	0	-1	6.3	6.3	<div></div>	ApplicationMaster	0				
1 to 1 of 1 entries																											

点击applicationmaster,由于没有配置DNS解析所以无法访问master，这里需要返回yarn页面复制网址粘贴到此处。



无法访问此网站

无法找到 master 的 DNS 地址。正在诊断该问题。

DNS_PROBE_STARTED

重新加载

这里就能看到已经完成的job

Apache Flink Dashboard

Version: 1.13.6 Commit: b2ca390 @ 2022-02-03T14:54:22+0100 Message:

Overview	
Jobs	
Running Jobs	
Completed Jobs	
Task Managers	
Job Manager	
Submit New Job	

Available Task Slots

0

Total Task Slots 0 Task Managers 0

Running Jobs

0

Finished 1 Cancelled 0 Failed 0

Running Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
No Data					

Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
Flink Java Job at Fri May 17 08:54:45 GMT 2024	2024-05-17 16:54:46	11s	2024-05-17 16:54:58	3	FINISHED

四、Flink消费Kafka数据

1、实验介绍

对于实时处理，实际工作中的数据源一般都是使用Kafka。Flink提供了一个特有的Kafka连接器去读写Kafka topic的数据。本实验通过本地打jar包上传到Flink集群，去处理终端Kafka输入的数据。

2、实验目的

- 安装Kafka;
- 本地编辑代码读取Kafka数据，并且打成jar包;
- 将jar包上传到Flink集群运行。

3、实验步骤

3.1 安装Kafka

上传压缩包中的 kafka_2.12-1.0.2.tgz 安装包到master并解压到指定路径下。（如何将压缩包传到服务器再传到docker内部见实验二文档）

```
scp .\kafka_2.12-1.0.2.tgz root@1.92.109.41:~/  
tar -xvzf kafka_2.12-1.0.2.tgz
```

将Kafka安装包分发到各个docker中。

```
docker cp kafka_2.12-1.0.2 c4d7fb215e86:/root/ (容器编号记得修改)  
docker cp kafka_2.12-1.0.2 8d9cecf44251:/root/  
docker cp kafka_2.12-1.0.2 f79a87a959e5:/root/  
docker cp kafka_2.12-1.0.2 84aa9ac2c578:/root/
```

配置**各个节点(4个节点都要弄哈)**的全局环境变量，在 **/etc/profile** 文件中添加Kafka路径。(路径可以自己定义到别的，和上面一致就可以。)

```
root@slave3: ~  
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))  
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).  
  
if [ "${PS1-}" ]; then  
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then  
    # The file bash.bashrc already sets the default PS1.  
    # PS1='\h:\w\$ '  
    if [ -f /etc/bash.bashrc ]; then  
      . /etc/bash.bashrc  
    fi  
  else  
    if [ "$(id -u)" -eq 0 ]; then  
      PS1='# '  
    else  
      PS1='$ '  
    fi  
  fi  
fi  
  
if [ -d /etc/profile.d ]; then  
  for i in /etc/profile.d/*.sh; do  
    if [ -r $i ]; then  
      . $i  
    fi  
  done  
  unset i  
fi  
export KAFKA_HOME=/root/kafka_2.12-3.5.0  
export PATH=$KAFKA_HOME/bin:$PATH
```

```
1 #Kafka  
2 export KAFKA_HOME=/root/kafka_2.12-1.0.2  
3 export PATH=$KAFKA_HOME/bin:$PATH  
4  
5 source /etc/profile
```

进入**各个节** Kafka 安装包的 config 目录，在 **server.properties** 配置文件中添加以下属性。

- 1 broker.id=1 (id值唯一，与其他节点值不同)
- 2 host.name=master (对应节点改为对应主机名)
- 3 zookeeper.connect=master:2181,slave1:2181,slave2:2181,slave3:2181


```
##### Server Basics #####

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0
host.name=master

##### Socket Server Settings #####
zookeeper.connect=master:2181,slave1:2181,slave2:2181,slave3:2181

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
#advertised.listeners=PLAINTEXT://your.host.name:9092

##### Server Basics #####

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1
host.name=slave1

##### Socket Server Settings #####
zookeeper.connect=master:2181,slave1:2181,slave2:2181,slave3:2181
# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092
```

```
##### Server Basics #####

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=3
host.name=slave2

##### Socket Server Settings #####
zookeeper.connect=master:2181,slave1:2181,slave2:2181,slave3:2181
# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092
```

验证是否安装成功（前提要启动ZooKeeper），在各个节点分别启动Kafka（注意执行该命令时，需要处于kafka的目录下）。

```
1 | ./bin/kafka-server-start.sh ./config/server.properties
```

若遇到报错：Kafka报错ERROR Exiting Kafka due to fatal exception during startup
[Kafka报错ERROR Exiting Kafka due to fatal exception during startup error exiting kafka. \(kafka.kafka\\$\)-CSDN博客](#)

检查各个节点的JPS，当各个节点的jps都出现Kafka时说明安装成功。下图需要截屏作为得分点

```
[root@zj-2023140696 ~]# docker exec -it master bash
root@master:~# jps
20096 TaskManagerRunner
24929 Kafka
18915 StandaloneSessionClusterEntrypoint
22435 QuorumPeerMain
6180 FlinkYarnSessionCli
19236 TaskManagerRunner
13611 SecondaryNameNode
21584 FlinkYarnSessionCli
13841 ResourceManager
20914 TaskManagerRunner
6325 FlinkYarnSessionCli
14230 NodeManager
13368 NameNode
22296 YarnSessionClusterEntrypoint
25371 Jps
14431 DataNode
```

测试完成之后请务必关闭各节点的kafka进程!!!! 否则将会影响后续实验

使用命令 `kafka-server-stop.sh`

或者直接 `kill` kafka的进程号码

关闭之后输入 `jps` 检查 kafka 进程是否已经消失

```
root@master:~/kafka_2.12-3.5.0# kill -9 25061
root@master:~/kafka_2.12-3.5.0# jps
25568 Jps
452 SecondaryNameNode
1669 QuorumPeerMain
10472 TaskManagerRunner
24106 FlinkYarnSessionCli
1098 DataNode
7083 StandaloneSessionClusterEntrypoint
7662 TaskManagerRunner
11919 TaskManagerRunner
208 NameNode
22930 FlinkYarnSessionCli
692 ResourceManager
14711 TaskManagerRunner
1464 NodeManager
9083 TaskManagerRunner
13308 TaskManagerRunner
[1]+  Killed                  ./bin/kafka-server-start.sh config/server.properties
```

3.2 创建maven工程

创建项目，打开IDEA，创建maven工程WordCount（具体步骤同实验一）。

3.3 添加Flink依赖

在pom文件中找到properties配置项，新增hadoop版本号。

```
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <hadoop.version>3.3.6</hadoop.version>
```

```
docker cp kafka_2.12-1.0.2 c4d7fb215e86:/root/ (容器编号记得修改)
docker cp kafka_2.12-1.0.2 8d9cecf44251:/root/
docker cp kafka_2.12-1.0.2 f79a87a959e5:/root/
docker cp kafka_2.12-1.0.2 84aa9ac2c578:/root/
```

具体的pom文件内的依赖：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```
<groupId>org.example</groupId>
<artifactId>MyWordCount</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <hadoop.version>3.3.6</hadoop.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.javatuples</groupId>
    <artifactId>javatuples</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.7</version>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-java</artifactId>
    <version>1.13.6</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka_2.12</artifactId>
    <version>1.0.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients_2.11</artifactId>
    <version>1.13.6</version>
  </dependency>
  <dependency>
    <groupId>org.apache.kafka</groupId>
```

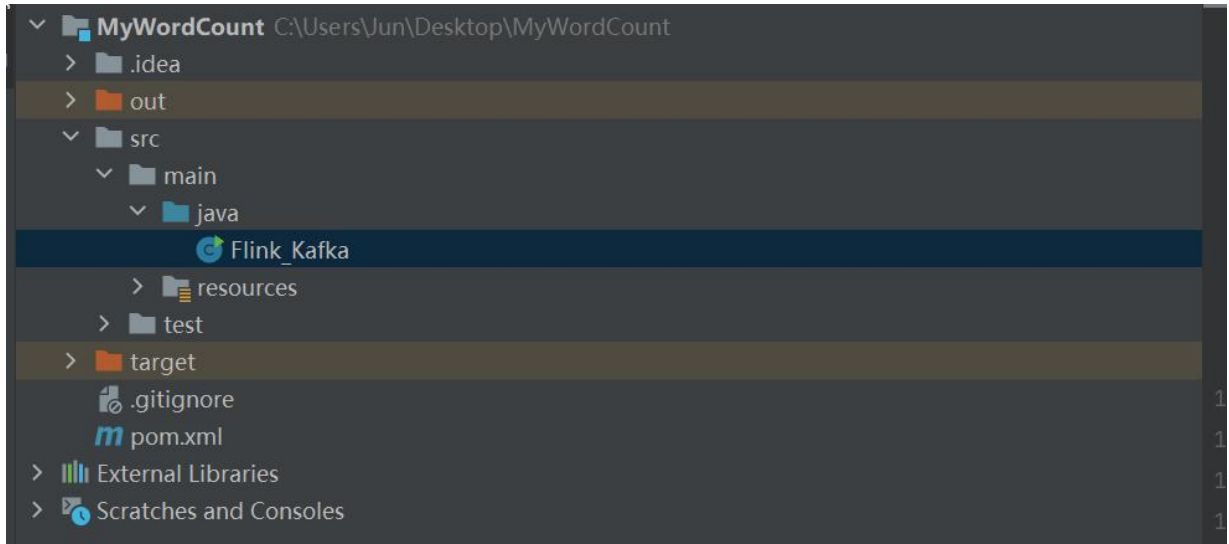
```
<artifactId>kafka-clients</artifactId>
<version>1.0.2</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-scala_2.12</artifactId>
  <version>1.13.6</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka_2.11</artifactId>
  <version>1.13.6</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-base</artifactId>
  <version>1.13.6</version>
</dependency>
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-client</artifactId>
  <version>${hadoop.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-common</artifactId>
  <version>${hadoop.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-hdfs</artifactId>
  <version>${hadoop.version}</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>RELEASE</version>
  <scope>test</scope>
</dependency>
</dependencies>

</project>
```

在pom文件中添加下列依赖并选择Maven->Reload project导入依赖包。

3.4 编写Flink代码

编写Flink读取Kafka数据的代码（可以在本机尝试运行代码正确后再打jar包）。



在项目的src/main/java目录下新建一个Flink_Kafka类，这个类名称大家可以自取，然后在该java文件中敲入以下代码

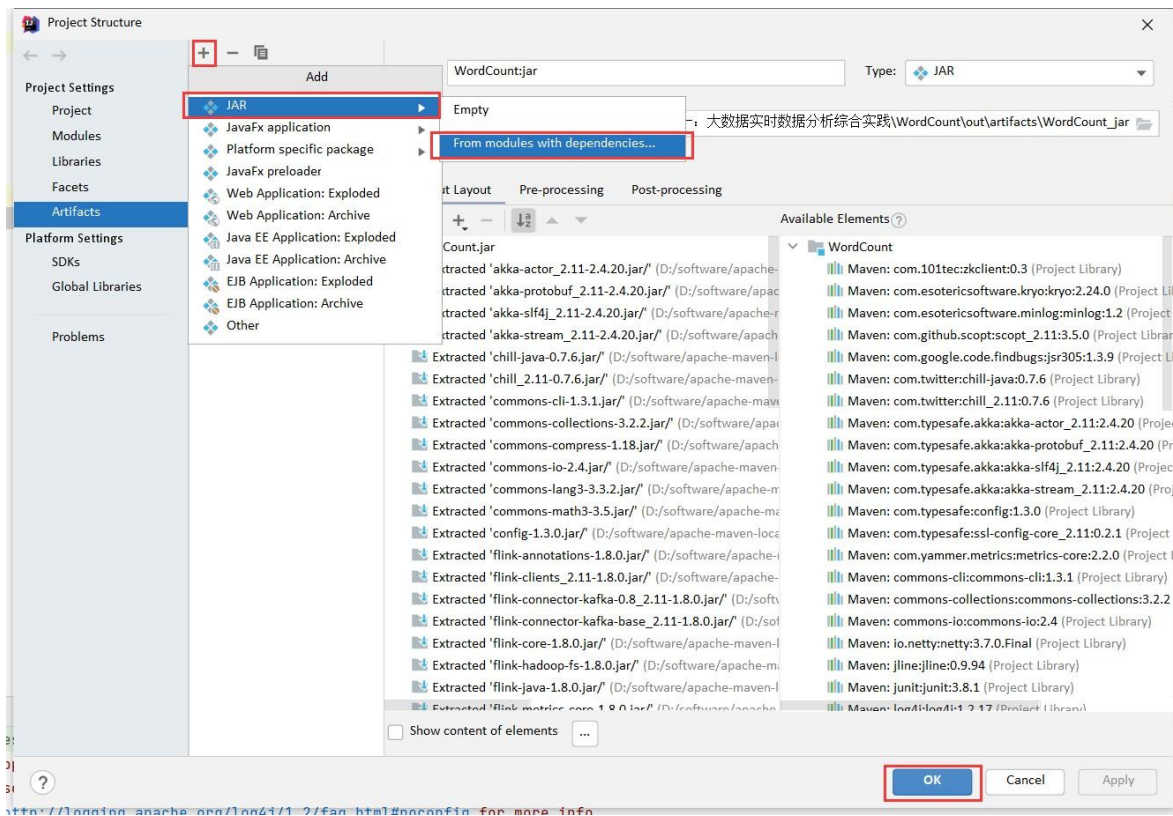

```

1  import org.apache.flink.api.common.functions.FlatMapFunction;
2  import org.apache.flink.api.common.serialization.SimpleStringSchema;
3  import org.apache.flink.api.java.tuple.Tuple2;
4  import org.apache.flink.streaming.api.datastream.DataStream;
5  import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
6  import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer;
7  import org.apache.flink.util.Collector;
8
9  import java.util.Properties;
10
11  public class Flink_Kafka {
12  public static void main(String[] args) throws Exception {
13
14      // 获取Flink运行环境
15      StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
16
17      // 配置Kafka连接属性
18      Properties properties = new Properties();
19      properties.setProperty("bootstrap.servers", "yty-2022140804-0001:9092"); //copy的时候记得修改对应的node节点名称
20      properties.setProperty("zookeeper.connect", "yty-2022140804-0001:2181"); //copy的时候记得修改对应的node节点名称
21      properties.setProperty("group.id", "1");
22
23      FlinkKafkaConsumer<String> myconsumer = new FlinkKafkaConsumer<>("test", new SimpleStringSchema(), properties);
24
25      myconsumer.setStartFromGroupOffsets();
26
27      DataStream<String> dataStream = env.addSource(myconsumer);
28
29      DataStream<Tuple2<String, Integer>> result = dataStream.flatMap(new MyFlatMapper()).keyBy(_1).sum(1);
30
31      result.print().setParallelism(1);
32
33      env.execute();
34  }
35
36  1 usage
37  public static class MyFlatMapper implements FlatMapFunction<String, Tuple2<String, Integer>> {
38
39      @Override
40      public void flatMap(String s, Collector<Tuple2<String, Integer>> out) throws Exception {
41          String[] words = s.split(" ");
42          for (String word:words){
43              out.collect(new Tuple2<>(word, 1));
44          }
45      }
46  }
47  }
48

```

3.5 IDEA打包（具体也可以见前面的实验报告）

在File->Project Structure中进行构建，点击Artifacts，点击上方的加号，选择JAR，From modules with dependencies，创建完成后点击OK。



选择Build->Build Artifacts->xxx.jar->Build，然后将生成的jar包上传到master节点。

3.6 zk运行jar包

运行本次任务的时候，请务必保证上一章 Flink on Yarn模式 中 3.4节的Flink集群环境还存在

就是下面这个命令，**不需要再次执行**，仅供大家回忆，如果你这个环境已经没有的话，请再次按照上一章3.4节给出的那四步操作，再次重新启动Flink集群环境即可

```
1 cd /root/flink
2 bin/yarn-session.sh -n 2 -jm 1024 -tm 1024 -d
```

由于我们需要用到KafKa自带的ZooKeeper。但是现在我们已经在四个节点启动了我们自己配置的ZooKeeper，所以我们现在需要停掉node1节点的ZooKeeper

执行以下命令停止ZooKeeper（只用停掉node1节点的即可）

```
1 zkServer.sh stop
```

然后启动KafKa自带的ZooKeeper。

```
1 cd /root/kafka_2.12-1.0.2
2 ./bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
[2024-05-16 14:18:35,952] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2024-05-16 14:18:35,965] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2024-05-16 14:18:35,965] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-05-16 14:18:35,968] INFO Snapshot loaded in 16 ms, highest zxid is 0x0, digest is 1371985504 (org.apache.zookeeper.server.ZKDatabase)
[2024-05-16 14:18:35,969] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-05-16 14:18:35,969] INFO Snapshot taken in 0 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2024-05-16 14:18:35,982] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
[2024-05-16 14:18:35,982] INFO zookeeper.request_throttler.shutdownTimeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2024-05-16 14:18:36,000] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2024-05-16 14:18:36,000] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2024-05-16 14:20:51,190] INFO Creating new log file: log.1 (org.apache.zookeeper.server.persistence.FileTxnLog)
^[[a^[[a
```

如果上图出现持续不停滚动的信息，说明启动失败

另外开启一个node01的终端，先进入到Kafka的安装目录下的config文件夹下，然后启动node01的Kafka。（在bin目录下）

```
1 kafka-server-start.sh -daemon server.properties
```

```
[root@zj-2023140696 ~]# docker exec -it master bash
root@master:~# jps
20096 TaskManagerRunner
24929 Kafka
18915 StandaloneSessionClusterEntrypoint
22435 QuorumPeerMain
6180 FlinkYarnSessionCli
19236 TaskManagerRunner
13611 SecondaryNameNode
21584 FlinkYarnSessionCli
13841 ResourceManager
20914 TaskManagerRunner
6325 FlinkYarnSessionCli
14230 NodeManager
13368 NameNode
22296 YarnSessionClusterEntrypoint
25371 Jps
14431 DataNode
```

(注意先要进入到kafka的安装目录下再执行命令)。下图需要截图作为一处得分点

```
./bin/kafka-topics.sh --create --bootstrap-server master:2181 --replication-factor 1 --
partitions 1 --topic test
./bin/kafka-console-producer.sh --broker-list master:9092 --topic test
```

```

[2021-10-12 10:07:39.165] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-12 10:07:39.166] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-12 10:07:39.166] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-12 10:07:39.174] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2021-10-12 10:07:39.306] INFO Accepted socket connection from /0:0:0:0:0:0:0:1:39220 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2021-10-12 10:07:39.314] INFO Client attempting to renew session 0x7c71fd9c60001 at /0:0:0:0:0:0:0:1:39220 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-12 10:07:39.320] INFO Established session 0x7c71fd9c60001 with negotiated timeout 6000 for client /0:0:0:0:0:0:0:1:39220 (org.apache.zookeeper.server.ZooKeeperServer)

```

另启动一个master的终端（也就是一共有三个master终端了），运行jar包。

```
1 [root@zyw-2021140807-0001 ~]# flink run -c [主类名] [jar包名]
```

主类名就是在idea中 src/main/java 目录下新建的类的名称，示例中起的名称是 Flink_Kafka，示例的项目名称是WordCount，因此，最后执行的命令就是，**下图需要截取作为一处得分点**

```
flink run -c Flink_Kafka WordCount.jar
```

```
root@master-8 flink run -o flink_kafka_RpsWordCount.jar
09-02-19 07:05:15,636 INFO org.apache.flink.yarn.cli.FlinkYarnSessionClient - Found Yarn properties file under /tmp/.yarn-properties-root
09-02-19 07:05:15,636 INFO org.apache.flink.yarn.cli.FlinkYarnSessionClient - Found Yarn properties file under /tmp/.yarn-properties-root
flinkKafka?yyyy!!!!!!
????????
????????
?ymlManager
????????
????????
09-02-19 07:05:16,391 WARN org.apache.flink.yarn.configuration.YarnConfigUtil - The configuration directory "/etc/flink/conf/" already contains a LOG4J config file.If you want to use logback, then please delete or rename the log configuration file.
09-02-19 07:05:16,265 INFO org.apache.hadoop.yarn.client.Proxy - Connecting to ResourceManager at master3/2:7162:0:0002
09-02-19 07:05:16,373 INFO org.apache.flink.yarn.YarnClusterDescriptor - No path for the Flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
09-02-19 07:05:16,408 INFO org.apache.flink.yarn.YarnClusterDescriptor - Found web interface class:3521c of application application_1734896930398_0002.
Job has been submitted with JobID c3e36d376da32f03b2c9e095a5a
```

若出现class无法引入的情况，参考：

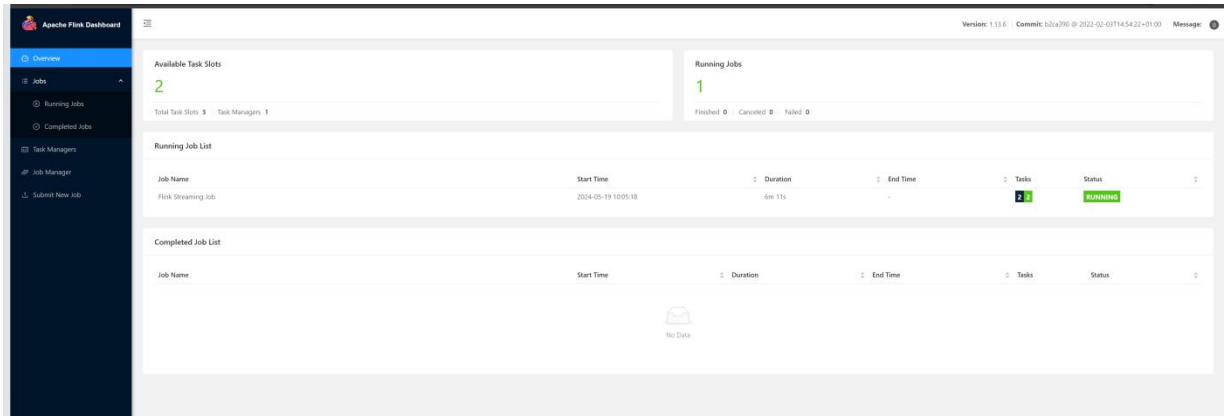
[Flink案例系列2-提交到yarn集群报错 java.lang.NoClassDefFoundError
FlinkKafkaConsumer flink yarn noclassdeffounderror kafka-CSDN博客](#)

需要将相关的jar包（从maven官网上下载）传入到flink下面的lib文件夹内。

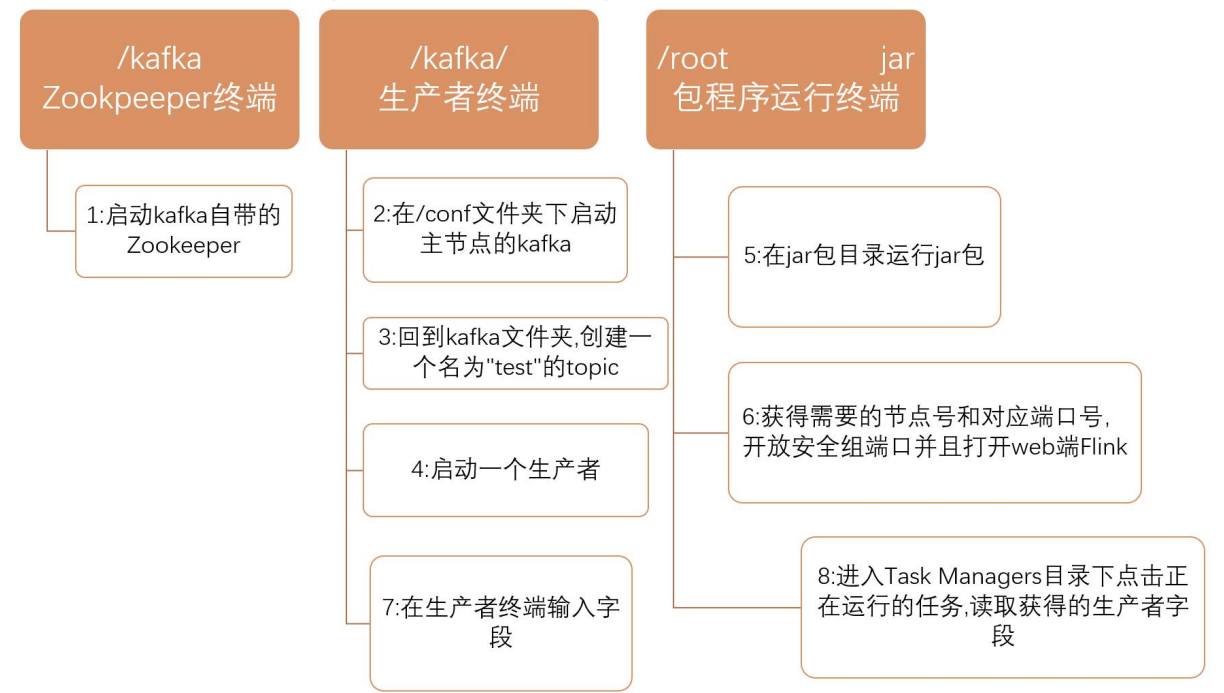
然后在生产者终端（也就是开启的第二个node1终端，那个终端最后输入的命令是kafka-console-producer.sh -- broker-list zyw-2021140807-0001:9092 --topic test， 开的终端比较多，请不要弄混）中输入单词。

```
root@master:~/kafka_2.12-1.0.2# ./bin/kafka-console-producer.sh --broker-list master:9092 --topic test
>123 321 211 21
^
```

根据前面Yarn Session(即上上张图片中红色框,框中显示的是某个节点,和一个端口号xxx,所以需要相对应节点对应的公网ip地址+xxx端口号)所在的机器公网ip访问Flink的Web管理界面。进入Task Managers目录下点击正在运行的任务。



整体的结构可以参考下图(20级张驰岳同学提供)



五、实验总结与分析

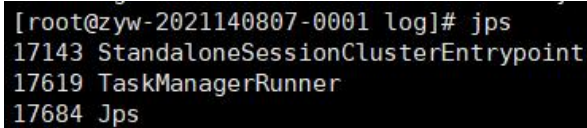
实验结束后应得到

1. Flink集群
2. Kafka集群
3. maven工程压缩包（只包含代码及pom文件即可）

实验给分点：

注：以下截图需包含具体学号/IP信息，否则不得分。此外每个实验的输入单词不可以完全相同。

1. 第一部分：local模式部署安装



```
[root@zyw-2021140807-0001 log]# jps
17143 StandaloneSessionClusterEntrypoint
17619 TaskManagerRunner
17684 Jps
```

主节点jps查看当前进程。（1分）

终端单词输入截图，Web管理界面或主节点终端输出out文件查看单词统计程序结果。（2分）

```
[root@zyw-2021140807-0001 log]# nc -lk 9000
qwe qwe qwe ^H^H^H^H^H^H flink harry harry hermin^Hone hermione
malfoy harry harry harry ronald flink malfoy
123 123 123 wer wer sdf
[

[root@zyw-2021140807-0001 ~]# cd /home/modules/flink-1.8.0/log/
[root@zyw-2021140807-0001 log]# tail -200f flink-root-taskexecutor-0-zyw-2021140807-0001.out
qwe : 3
hermione : 1
hermione : 1
harry : 2
flink : 1
malfoy : 2
flink : 1
ronald : 1
harry : 3
123 : 3
sdf : 1
wer : 2
```

2. 第二部分：standalone模式部署安装

四个节点jps查看当前进程或Web管理界面查看Task Manager目录信息。（1分）

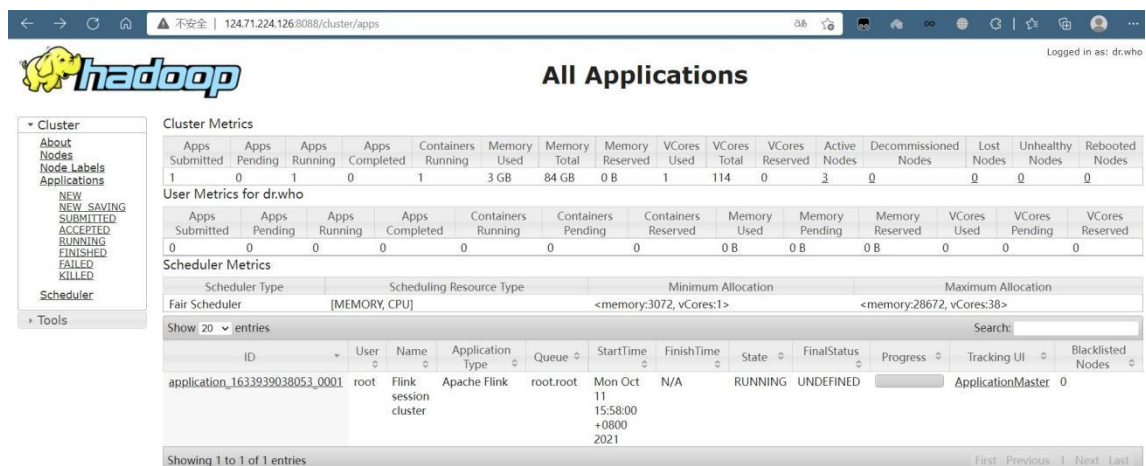
Task Managers									
Path, ID	Data Port	Last Heartbeat	All Slots	Free Slots	CPU Cores	Physical Memory	JVM Heap Size	Flink Managed Memory	
akka.tcp://flink@192.168.0.174:43857/user/taskmanager_0 724187DEAB774C4A1AC785EB78A96C44	46649	2021-10-07, 16:20:30	1	1	2	3.40 GB	922 MB	553 MB	
akka.tcp://flink@192.168.0.20:41311/user/taskmanager_0 C567339C88DA6B326F8771A272619676	43009	2021-10-07, 16:20:30	1	1	2	3.40 GB	922 MB	553 MB	
akka.tcp://flink@192.168.0.221:38757/user/taskmanager_0 DC7788E41204F806BC81A25D585BE5DE	40529	2021-10-07, 16:20:30	1	0	2	3.40 GB	922 MB	553 MB	
akka.tcp://flink@192.168.0.116:36081/user/taskmanager_0 3ADE2F99FC43FEB7132B78F550CA1FC6	42259	2021-10-07, 16:20:30	1	1	2	3.40 GB	922 MB	553 MB	

终端单词输入截图，Web管理界面或对应节点终端输出out文件查看单词统计程序结果。（2分）

```
[root@zyw-2021140807-0002 ~]# cd /home/modules/flink-1.8.0/log/
[root@zyw-2021140807-0002 log]# tail -200f flink-root-taskexecutor-0-zyw-2021140807-0002.out
flink : 4
nimbus : 2
123 : 4
zookeeper : 1
hadoop : 2
```

3. 第三部分：Flink on Yarn模式

- 给出Yarn的Web界面，体现出当前启动的Flink应用。（1分）



- 任务提交并运行结果。（3分）

```
[root@zyw-2021140807-0001 ~]# cd /home/modules/flink-1.8.0
[root@zyw-2021140807-0001 flink-1.8.0]# bin/flink run examples/batch/WordCount.jar -input hdfs://zyw-2021140807-0001:8020/flink_input -output hdfs://zyw-2021140807-0001:8020/flink_output/wordcount-result.txt
2021-10-11 16:43:47,626 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file under /tmp/.yarn-properties-root.
2021-10-11 16:43:47,626 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file under /tmp/.yarn-properties-root.
2021-10-11 16:43:47,980 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default parallelism to 2
2021-10-11 16:43:47,980 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default parallelism to 2
YARN properties set default parallelism to 2
2021-10-11 16:43:48,059 INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at zyw-2021140807-0001/192.168.0.116:8032
2021-10-11 16:43:48,162 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2021-10-11 16:43:48,162 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2021-10-11 16:43:48,165 WARN org.apache.flink.yarn.AbstractYarnClusterDescriptor - Neither the HADOOP_CONF_DIR nor the YARN_CONF_DIR environment variable is set. The Flink YARN Client needs one of these to be set to properly load the Hadoop configuration for accessing YARN.
2021-10-11 16:43:48,217 INFO org.apache.flink.yarn.AbstractYarnClusterDescriptor - Found application JobManager host name 'zyw-2021140807-0002' and port '44127' from supplied application id 'application_1633939038053_0001'
Starting execution of program
Program execution finished
Job with JobID ea0b4f433a7cc81be8c86deeb514a5ec has finished.
Job Runtime: 14885 ms
```

查看hdfs上的输出文件。（2分）

```
[root@zyw-2021140807-0001 flink-1.8.0]# hdfs dfs -cat /flink_output/wordcount-result.txt
21/10/11 16:46:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
flink 1
hadoop 1
hello 1
hive 1
spark 1
world 1
```

4. 第四部分：Flink消费Kafka数据

四个节点jps出现Kafka进程截图。（2分）

```
[root@zyw-2021140807-0001 config]# kafka-server-start.sh -daemon server.properties
[root@zyw-2021140807-0001 config]# jps
10430 Kafka
7729 QuorumPeerMain
8381 NameNode
8597 SecondaryNameNode
8778 ResourceManager
10482 Jps
```

maven项目压缩包，包含代码和pom文件即可。（3分）

启动生产者、运行jar包、Web界面输出结果截图。（3分）

```
[root@zyw-2021140807-0001 ~]# flink run -c WordCount WordCount.jar
2021-10-12 15:47:26,060 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file under /tmp/.yarn-properties-root.
2021-10-12 15:47:26,060 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - Found Yarn properties file under /tmp/.yarn-properties-root.
2021-10-12 15:47:26,510 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default parallelism to 2
2021-10-12 15:47:26,510 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - YARN properties set default parallelism to 2
YARN properties set default parallelism to 2
2021-10-12 15:47:26,560 INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at zyw-2021140807-0001/192.168.0.116:8032
2021-10-12 15:47:26,664 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2021-10-12 15:47:26,664 INFO org.apache.flink.yarn.cli.FlinkYarnSessionCli - No path for the flink jar passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
2021-10-12 15:47:26,666 WARN org.apache.flink.yarn.AbstractYarnClusterDescriptor - Neither the HADOOP_CONF_DIR nor the YARN_CONF_DIR environment variable is set. The Flink YARN Client needs one of these to be set to properly load the Hadoop configuration for accessing YARN.
2021-10-12 15:47:26,717 INFO org.apache.flink.yarn.AbstractYarnClusterDescriptor - Found application JobManager host name 'zyw-2021140807-0004' and port '46273' from supplied application id 'application_1634004428186_0001_01_000003'
Starting execution of program
Yarn Session所在的机器及对应的端口
```

← → ↺ 🏠 ⚠️ 不安全 | 124.71.238.18:46273/#/taskmanager/container_1634004428186_0001_01_000003/stdout

Apache Flink Dashboard

OverviewRunning JobsCompleted JobsTask ManagersJob ManagerSubmit new Job

Task Manager

Last Heartbeat: 2021-10-12, 15:54:32 akka.tcp://flink@zyw-2021140807-0003:43245/user/taskmanager_0

MetricsLogsStdout

Task Manager Output

(123,1)
(123,2)
(123,3)
(456,1)
(456,2)
(qwe,1)
(qwe,2)
(qwe,3)
(asd,1)
(asd,2)
(zxc,1)
(123,4)
(qwe,4)
(qwe,5)
(harry,1)
(harry,2)
(123,5)