**zookeeper+kafka+storm+flume 环境搭建**

# 1.环境准备

## (1) 三台主机

192.168.0.66    h1

192.168.0.67    h2

192.168.0.68    h3

(2)

66 上执行 ：hostnamectl --static set-hostname  h1

67 上执行 ：hostnamectl --static set-hostname  h2

68 上执行 ：hostnamectl --static set-hostname  h3

(3) 每个节点上执行

echo '

192.168.0.66   h1

192.168.0.67   h2

192.168.0.68   h3' >> /etc/hosts

(4)关闭防火墙

systemctl stop firewalld.service

systemctl disable firewalld.service

(5)安装 jdk

# 2、zookeeper 集群搭建

（1）下载： wgethttp://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.10/zookeeper-3.4.10.tar.gz

（2）解压： tar -zxvfzookeeper-3.4.10.tar.gz

（3）复制到 /usr/local     cp   -R  zookeeper-3.4.10  /usr/local

（4）复制配置文件   cp /usr/local/zookeeper-3.4.10/conf/zoo\_sample.cfg  /usr/local/zookeeper-3.4.10/conf/zoo.cfg

（5）修改配置文件

server.1=h1:2888:3888  
server.2=h2:2888:3888  
server.3=h3:2888:3888

（6）配置 myid

mkdir -p  /tmp/zookeeper

vi /tmp/zookeeper/myid

h1:1

h2:2

h3:3

对应每个节点的 myid 的值

（7）启动每个节点的zk

/usr/local/zookeeper-3.4.10/bin/zkServer.sh start

（8）查看zk 的状态

/usr/local/zookeeper-3.4.10/bin/zkServer.sh status

（9）测试ZK

任意节点

 /usr/local/zookeeper-3.4.10/bin/zkCli.sh -server 192.168.0.127:2181

create /c1project c1projecttest

get /c1project

# 3.kafka 集群搭建

(1) 下载 kafka

wget http://mirror.bit.edu.cn/apache/kafka/0.9.0.0/kafka\_2.10-0.9.0.0.tgz

(2) 解压并放置 /usr/local

tar -zxvf kafka\_2.10-0.9.0.0.tgz

cp -R kafka\_2.10-0.9.0.0 /usr/local

(3) 修改配置文件

vi /usr/local/kafka\_2.10-0.9.0.0/config/server.properties

h1:   broker.id=1

h2:   broker.id=2

h3:   broker.id=3

zookeeper.connect=storm1:2181,storm2:2181,storm3:2181

注: kafka 外网配置

# Licensed to the Apache Software Foundation (ASF) under one or more

# contributor license agreements. See the NOTICE file distributed with

# this work for additional information regarding copyright ownership.

# The ASF licenses this file to You 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 kafka.server.KafkaConfig for additional details and defaults

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

# The id of the broker. This must be set to a unique integer for each broker.

broker.id=0

############################# Socket Server Settings #############################

#listeners=PLAINTEXT://:9092

listeners=PLAINTEXT://192.168.0.48:9092

# The port the socket server listens on

port=9092

# Hostname the broker will bind to. If not set, the server will bind to all interfaces

host.name=192.168.0.48

# Hostname the broker will advertise to producers and consumers. If not set, it uses the

# value for "host.name" if configured. Otherwise, it will use the value returned from

# java.net.InetAddress.getCanonicalHostName().

#advertised.host.name=<hostname routable by clients>

advertised.host.name=192.168.0.48

# The port to publish to ZooKeeper for clients to use. If this is not set,

# it will publish the same port that the broker binds to.

#advertised.port=<port accessible by clients>

advertised.port=9092

advertised.listeners=PLAINTEXT://58.213.76.82:9092

#PLAINTEXT://192.168.0.48:9092,SSL://58.213.76.82:9092

# The number of threads handling network requests

num.network.threads=3

# The number of threads doing disk I/O

num.io.threads=8

# The send buffer (SO\_SNDBUF) used by the socket server

socket.send.buffer.bytes=102400

# The receive buffer (SO\_RCVBUF) used by the socket server

socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)

socket.request.max.bytes=104857600

############################# Log Basics #############################

# A comma seperated list of directories under which to store log files

log.dirs=/tmp/kafka-logs

# The default number of log partitions per topic. More partitions allow greater

# parallelism for consumption, but this will also result in more files across

# the brokers.

num.partitions=1

# The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.

# This value is recommended to be increased for installations with data dirs located in RAID array.

num.recovery.threads.per.data.dir=1

############################# Log Flush Policy #############################

# Messages are immediately written to the filesystem but by default we only fsync() to sync

# the OS cache lazily. The following configurations control the flush of data to disk.

# There are a few important trade-offs here:

# 1. Durability: Unflushed data may be lost if you are not using replication.

# 2. Latency: Very large flush intervals may lead to latency spikes when the flush does occur as there will be a lot of data to flush.

# 3. Throughput: The flush is generally the most expensive operation, and a small flush interval may lead to exceessive seeks.

# The settings below allow one to configure the flush policy to flush data after a period of time or

# every N messages (or both). This can be done globally and overridden on a per-topic basis.

# The number of messages to accept before forcing a flush of data to disk

#log.flush.interval.messages=10000

# The maximum amount of time a message can sit in a log before we force a flush

#log.flush.interval.ms=1000

############################# Log Retention Policy #############################

# The following configurations control the disposal of log segments. The policy can

# be set to delete segments after a period of time, or after a given size has accumulated.

# A segment will be deleted whenever \*either\* of these criteria are met. Deletion always happens

# from the end of the log.

# The minimum age of a log file to be eligible for deletion

log.retention.hours=168

# A size-based retention policy for logs. Segments are pruned from the log as long as the remaining

# segments don't drop below log.retention.bytes.

#log.retention.bytes=1073741824

# The maximum size of a log segment file. When this size is reached a new log segment will be created.

log.segment.bytes=1073741824

# The interval at which log segments are checked to see if they can be deleted according

# to the retention policies

log.retention.check.interval.ms=300000

# By default the log cleaner is disabled and the log retention policy will default to just delete segments after their retention expires.

# If log.cleaner.enable=true is set the cleaner will be enabled and individual logs can then be marked for log compaction.

log.cleaner.enable=false

############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).

# This is a comma separated host:port pairs, each corresponding to a zk

# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".

# You can also append an optional chroot string to the urls to specify the

# root directory for all kafka znodes.

zookeeper.connect=192.168.0.48:2181

# Timeout in ms for connecting to zookeeper

zookeeper.connection.timeout.ms=6000

(4)启动 kafka

三个节点：  /usr/local/kafka\_2.10-0.9.0.0/bin/kafka-server-start.sh   -daemon /usr/local/kafka\_2.10-0.9.0.0/config/server.properties

jps 后能看到 kafka 的进程。

(5)测试 kafka

a. 创建topic

/usr/local/kafka\_2.10-0.9.0.0//bin/kafka-topics.sh --create --zookeeper h1:2181,h2:2181,h3:2181 --replication-factor 3 --partitions 5 --topic test

b.查看topic

/usr/local/kafka\_2.10-0.9.0.0/bin/kafka-topics.sh --describe --zookeeper h1:2181,h2:2181,h3:2181 --topic test

c.生产者  消费者

/usr/local/kafka\_2.10-0.9.0.0/bin/kafka-console-producer.sh --broker-list h1:9092,h2:9092,h3:9092 --topic test  
/usr/local/kafka\_2.10-0.9.0.0/bin/kafka-console-consumer.sh --zookeeper h1:2181,h2:2181,h3:2181 --from-beginning --topic test

# 4.storm 搭建

(1)下载storm

wget http://mirror.bit.edu.cn/apache/storm/apache-storm-0.10.2/apache-storm-0.10.2.tar.gz

(2)解压并放置 /usr/local

cp -R apache-storm-0.10.2  /usr/local

(3)修改配置

vi /usr/local/apache-storm-0.10.2/conf/storm.yaml

 storm.zookeeper.servers:  
     - "h1"  
     - "h2"  
     - "h3"  
  
 storm.zookeeper.port: 2181  
 nimbus.host: "h1"

 supervisor.slots.ports:

     - 6700  
     - 6701  
     - 6702  
     - 6703

      storm.local.dir: "/tmp/storm"

（4）启动 storm

h1节点上:    /usr/local/apache-storm-0.10.2/bin/storm nimbus

后台启动：nohup /usr/local/apache-storm-0.10.2/bin/storm nimbus > /dev/null 2>&1 &

h2/h3节点上:    /usr/local/apache-storm-0.10.2/bin/storm supervisor &

后台启动：   nohup   /usr/local/apache-storm-0.10.2/bin/storm supervisor > /dev/null 2>&1 &

h2节点上：   /usr/local/apache-storm-0.10.2/bin/storm ui &

后台启动：  nohup  /usr/local/apache-storm-0.10.2/bin/storm ui > /dev/null 2>&1 &

（5）打开 http://h2:8080/ 查看

# 5.flume

(1) 下载 flume

 wget  http://www.apache.org/dist/flume/1.6.0/apache-flume-1.6.0-bin.tar.gz

（2）解压 并放置 /usr/local

tar -zxvf apache-flume-1.6.0-bin.tar.gz

cp -R apache-flume-1.6.0-bin /usr/local

(3) 配置 kafka

vi /usr/local/apache-flume-1.6.0-bin/conf/kafka.properties

agent.sources = s1  
agent.channels = c1  
agent.sinks = k1  
  
  
agent.sources.s1.type=exec  
agent.sources.s1.command=tail -F /tmp/logs/kafka.log  
agent.sources.s1.channels=c1  
agent.channels.c1.type=memory  
agent.channels.c1.capacity=10000  
agent.channels.c1.transactionCapacity=100  
  
  
#设置Kafka接收器  
agent.sinks.k1.type= org.apache.flume.sink.kafka.KafkaSink  
#设置Kafka的broker地址和端口号  
agent.sinks.k1.brokerList=h1:9092,h2:9092,h3:9092  
#设置Kafka的Topic  
agent.sinks.k1.topic=kafkatest  
#设置序列化方式  
agent.sinks.k1.serializer.class=kafka.serializer.StringEncoder  
  
  
agent.sinks.k1.channel=c1

注意 1.7配置 新增 a1.sources.r1.type = TAILDIR 属性

a1.sources = r1

a1.sinks = k1

a1.channels = c1

#Describe/configure the source avro

a1.sources.r1.type = TAILDIR

a1.sources.r1.positionFile = /tmp/flume/source/file\_position.json

a1.sources.r1.filegroups = f1

a1.sources.r1.filegroups.f1 = /logs/cfpu-web-erp/cfp.log

# Describe the sink

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink

#a1.sinks.k1.type = org.shirdrn.flume.sink.RealtimeMessageSink

a1.sinks.k1.brokerList = 192.168.0.48:9092

a1.sinks.k1.topic = project\_logs

a1.sinks.k1.serializer.class = kafka.serializer.StringEncoder

a1.sinks.k1.producer.type = async

a1.sinks.k1.message.send.max.retries = 3

a1.sinks.k1.client.id = flume\_test\_\_event\_2\_1

a1.sinks.k1.event.decoder.count = 8

a1.sinks.k1.output.stat.event.batch.size = 2000

a1.sinks.k1.event.decoder.queue.size = 1000

#a1.sinks.k1.topic = project\_logs

a1.sinks.k1.producer.type = async

a1.sinks.k1.message.send.max.retries = 3

a1.sinks.k1.client.id = flume\_test\_\_event\_2\_1

a1.sinks.k1.event.decoder.count = 8

a1.sinks.k1.output.stat.event.batch.size = 2000

a1.sinks.k1.event.decoder.queue.size = 1000

#a1.sinks.k1.topic = project\_logs

#a1.sinks.k1.brokerList = 192.168.0.48:9092

#a1.sinks.k1.requiredAcks = 1

#a1.sinks.k1.batchSize = 100

# Use a channel which buffers events in memory

#a1.channels.c1.type = memory

#a1.channels.c1.capacity = 1000

#a1.channels.c1.transactionCapacity = 100

a1.channels.c1.type = file

#写出到检查点的目录

a1.channels.c1.checkpointDir = /tmp/flume/channels/test/checkpoint

#告诉 channel 一旦它被完全写出是否支持检查点。参数值 true/false 如果设置true，backupCheckpointDir参数必须设置

a1.channels.c1.useDualCheckpoints = trues

#支持检查点的目录。如果主检查点损坏或不完整，channel 可以从备份中恢复从而避免数据文件的完整回放。不同于 checkpointDir

a1.channels.c1.backupCheckpointDir = /tmp/flume/channels/test/backup

#写入事件到以逗号分隔的列表目录。配置多个目录，每个挂载不同的磁盘，通过并行写入磁盘可以显著提高性能。

a1.channels.c1.dataDirs = /tmp/flume/channels/test/data

#单个事物可以写入或者读取的事务的最大数量

a1.channels.c1.transactionCapacity = 100000

#可以保存的提交事件的最大数量

a1.channels.c1.capacity = 500000

#连续检查点之间的时间间隔，秒为单位

a1.channels.c1.checkpointInterval = 60000

#每次写入或读取应该等待完成的最大时间周期（以秒为单位）

a1.channels.c1.keep-alive = 30

#每个数据文件的最大大小，字节为单位，一旦文件达到这个大小，该文件保存关闭并在那个目录下创建一个新的数据文件

a1.channels.c1.maxFileSize = 2000000

# Bind the source and sink to the channel

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

新建log文件

mkdir  -p /tmp/logs

vi /tmp/logs/kafka.log

（4）启动 flume  cd  /usr/local/apache-flume-1.6.0-bin

bin/flume-ng agent --conf-file  conf/kafka.properties -c conf/ --name agent -Dflume.root.logger=DEBUG,console

后台启动：nohup  bin/flume-ng agent --conf-file  conf/flume-nginx-dir.properties -c conf/ --name a1 -Dflume.monitoring.type=http -Dflume.monitoring.port=5653 -Dflume.root.logger=INFO,console > ./start.log 2>&1 &

http://192.168.0.48:5653/metrics 即可看到监控数据的信息

(5)新建脚本

mkdir mfz

vi kafkaoutput.sh

for((i=0;i<=1000;i++));  
do echo "kafka\_test-"+$i>>/tmp/logs/kafka.log;  
done

赋予可执行权限

chmod a+x  kafkaoutput.sh

./kafkaoutput.sh

/tmp/logs/kafka.log    可看到日志结果。

打开kafka的消费终端即可看到结果。

# 6.storm拓扑

(1)启动

bin/storm jar XXX.jar com.whl.XXX XXX

xx.jar 包名+类名+拓扑名称

(2)停止

bin/storm kill (拓扑名称)