Flume 集群 （版本1.7.0）

1，修改flume-env.sh 添加java home地址，修改环境变量，添加flume home和flume conf dir

2，添加agent配置文件 例如

kafka.conf

# example.conf: A single-node Flume configuration

# Name the components on this agent

agent.sources = r1

agent.sinks = k1

agent.channels = c1

# Describe/configure the source

agent.sources.r1.type = netcat

agent.sources.r1.bind = 10.8.240.248

agent.sources.r1.port = 6666

# Describe the sink

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

agent.sinks.k1.kafka.topic = kafka8

agent.sinks.k1.kafka.bootstrap.servers = x:9092

agent.sinks.k1.kafka.flumeBatchSize = 20

agent.sinks.k1.kafka.producer.acks = 1

agent.sinks.k1.kafka.producer.linger.ms = 1

agent.sinks.k1.kafka.producer.compression.type = snappy

# Use a channel which buffers events in memory

agent.channels.c1.type = memory

agent.channels.c1.capacity = 1000

agent.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel

agent.sources.r1.channels = c1

agent.sinks.k1.channel = c1

3，启动

./flume-ng agent --conf ../conf --conf-file ../conf/kafka.conf --name a1 -Dflume.root.logger=INFO,console

--conf 表示配置地址

--conf-file 表示具体配置文件

--name 表示具体的agent名称

Zookeeper 集群 （版本3.4.6）

1，修改环境变量，添加ZOOKEEPER HOME 和 PATH路径

2，在zookeeper目录下创建data和logs目录

3，将/home/ksuser/software/zookeeper3.4.6/zookeeper-3.4.6/conf 目录下的zoo\_sample.cfg 文件拷贝一份，命名为为zoo.cfg

# The number of milliseconds of each tick

tickTime=20000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/data/software/zookeeper-3.4.6/data

dataLogDir=/data/software/zookeeper-3.4.6/logs

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=localhost:2888:3888

# The number of milliseconds of each tick  
tickTime=2000  
# The number of ticks that the initial  
# synchronization phase can take  
initLimit=10  
# The number of ticks that can pass between  
# sending a request and getting an acknowledgement  
syncLimit=5  
# the directory where the snapshot is stored.  
# do not use /tmp for storage, /tmp here is just  
# example sakes.  
dataDir=/home/keuser/software/zookeeper3.4.6/zookeeper-3.4.6/data  
dataLogDir=/home/keuser/software/zook/zookeeper-3.4.6/logs  
# the port at which the clients will connect  
clientPort=2181  
#2888,3888 are election port  
server.1=x:2888:3888

4，在 dataDir=/usr/local/zookeeper3.4.6/zookeeper-3.4.6/data 下创建myid 文件,编辑 myid 文件，并在对应的 IP 的机器上输入对应的编号。如在 zookeeper 上， myid文件内容就是 1。 如果只在单点上进行安装配置， 那么只有一个 server.1

5，启动并查看状态 zkServer.sh start zkServer.sh status

Kafka 集群 （版本0.11.0.3）

1，下载解压，修改config下的server.properties

参数解释：

broker.id=0 #当前机器在集群中的唯一标识，和zookeeper的myid性质一样

port=19092 #当前kafka对外提供服务的端口默认是9092

host.name=192.168.7.100 #这个参数默认是关闭的，在0.8.1有个bug，DNS解析问题，失败率的问题。

num.network.threads=3 #这个是borker进行网络处理的线程数

num.io.threads=8 #这个是borker进行I/O处理的线程数

log.dirs=/opt/kafka/kafkalogs/ #消息存放的目录，这个目录可以配置为“，”逗号分割的表达式，上面的num.io.threads要大于这个目录的个数这个目录，如果配置多个目录，新创建的topic他把消息持久化的地方是，当前以逗号分割的目录中，那个分区数最少就放那一个

socket.send.buffer.bytes=102400 #发送缓冲区buffer大小，数据不是一下子就发送的，先回存储到缓冲区了到达一定的大小后在发送，能提高性能

socket.receive.buffer.bytes=102400 #kafka接收缓冲区大小，当数据到达一定大小后在序列化到磁盘

socket.request.max.bytes=104857600 #这个参数是向kafka请求消息或者向kafka发送消息的请请求的最大数，这个值不能超过java的堆栈大小

num.partitions=1 #默认的分区数，一个topic默认1个分区数

log.retention.hours=168 #默认消息的最大持久化时间，168小时，7天

message.max.byte=5242880 #消息保存的最大值5M

default.replication.factor=2 #kafka保存消息的副本数，如果一个副本失效了，另一个还可以继续提供服务

replica.fetch.max.bytes=5242880 #取消息的最大直接数

log.segment.bytes=1073741824 #这个参数是：因为kafka的消息是以追加的形式落地到文件，当超过这个值的时候，kafka会新起一个文件

log.retention.check.interval.ms=300000 #每隔300000毫秒去检查上面配置的log失效时间（log.retention.hours=168 ），到目录查看是否有过期的消息如果有，删除

log.cleaner.enable=false #是否启用log压缩，一般不用启用，启用的话可以提高性能

zookeeper.connect=192.168.7.100:12181,192.168.7.101:12181,192.168.7.107:1218 #设置zookeeper的连接端口

实际配置：

#broker.id=0 每台服务器的broker.id都不能相同

#hostname

host.name=192.168.7.100

#在log.retention.hours=168 下面新增下面三项

message.max.byte=5242880

default.replication.factor=2

replica.fetch.max.bytes=5242880

#设置zookeeper的连接端口

zookeeper.connect=192.168.7.100:12181,192.168.7.101:12181,192.168.7.107:12181

# 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=1

host.name=127.0.0.1

# Switch to enable topic deletion or not, default value is false

#delete.topic.enable=true

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

# The address the socket server listens on. It will get the value returned from

# java.net.InetAddress.getCanonicalHostName() if not configured.

# FORMAT:

# listeners = listener\_name://host\_name:port

# EXAMPLE:

# listeners = PLAINTEXT://your.host.name:9092

#listeners=PLAINTEXT://:9092

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

# it uses the value for "listeners" if configured. Otherwise, it will use the value

# returned from java.net.InetAddress.getCanonicalHostName().

#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config documentation for more details

#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL\_PLAINTEXT:SASL\_PLAINTEXT,SASL\_SSL:SASL\_SSL

# The number of threads that the server uses for receiving requests from the network and sending responses to the network

num.network.threads=3

# The number of threads that the server uses for processing requests, which may include 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=/data/software/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

############################# Internal Topic Settings #############################

# The replication factor for the group metadata internal topics "\_\_consumer\_offsets" and "\_\_transaction\_state"

# For anything other than development testing, a value greater than 1 is recommended for to ensure availability such as 3.

offsets.topic.replication.factor=1

transaction.state.log.replication.factor=1

transaction.state.log.min.isr=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 due to age

log.retention.hours=168

message.max.byte=5242880

default.replication.factor=2

replica.fetch.max.bytes=5242880

# 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. Functions independently of log.retention.hours.

#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

############################# 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=localhost:2181

# Timeout in ms for connecting to zookeeper

zookeeper.connection.timeout.ms=6000

############################# Group Coordinator Settings #############################

# The following configuration specifies the time, in milliseconds, that the GroupCoordinator will delay the initial consumer rebalance.

# The rebalance will be further delayed by the value of group.initial.rebalance.delay.ms as new members join the group, up to a maximum of max.poll.interval.ms.

# The default value for this is 3 seconds.

# We override this to 0 here as it makes for a better out-of-the-box experience for development and testing.

# However, in production environments the default value of 3 seconds is more suitable as this will help to avoid unnecessary, and potentially expensive, rebalances during application startup.

group.initial.rebalance.delay.ms=0

2，启动服务 bin/kafka-server-start.sh -daemon ../config/server.properties &

3，创建topic ./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic kafka

--replication-factor 1 #复制两份

--partitions 3 #创建3个分区

--topic #主题为kafka

./kafka-topics.sh --zookeeper localhost:2181 --list

4，'''在一台服务器上创建一个生产者'''

./kafka-console-producer.sh --broker-list localhost:9092 --topic kafka

'''在一台服务器上创建一个消费者'''

./kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning

在生产者端进行写入，可以看到消费者端有相应的输出

-server -Xss1m

Hadoop 集群 （版本2.7.3）

1，修改hadoop-env.sh 添加java环境变量地址

2，修改core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop01:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/home/ksuser/software/hadoop/tmp</value>

<configuration>

<property>

   <name>fs.default.name</name>

   <value>hdfs://x:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

   <value>/data/software/hadoop-2.7.3/tmp</value>

   </property>

</configuration>

3、修改 hdfs-site.xml

<property>

<name>dfs.namenode.name.dir</name>

<value>/root/hadoopData/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/root/hadoopData/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property><name>dfs.secondary.http.address</name>

<value>hadoop01:50090</value>

</property>

<configuration>

  <property>

    <name>dfs.namenode.name.dir</name>

    <value>/data/software/hadoop-2.7.3/hadoopData/namenode</value>

  </property>

  <property>

    <name>dfs.datanode.data.dir</name>

    <value>/data/software/hadoop-2.7.3/hadoopData/datanode</value>

  </property>

  <property>

    <name>dfs.replication</name>

    <value>1</value>

  </property>

</configuration>

4、修改 mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<configuration>

  <property>

    <name>mapreduce.framework.name</name>

    <value>yarn</value>

  </property>

</configuration>

5、修改 yarn-site.xml

<property>

<name>yarn.resourcemanager.hostname</name>

<value>hadoop01</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<configuration>

  <property>

    <name>yarn.resourcemanager.hostname</name>

    <value>localhost</value>

  </property>

  <property>

    <name>yarn.nodemanager.aux-services</name>

    <value>mapreduce\_shuffle</value>

  </property>

  <property>

     <name>yarn.log-aggregation-enable</name>

     <value>true</value>

  </property>

  <property>

   <name>yarn.log.server.url</name>

   <value>http://localhost:19888/jobhistory/logs</value>

  </property>

</configuration>

6、修改 slaves

7，修改环境变量，bin下hadoop namenode -format格式化，启动集群，50070端口看hdfs，8088端口看yarnß

Spark 集群 （版本2.3.0）

1，下载解压，添加环境变量 scala home和spark home

2，修改spark-env.sh

export JAVA\_HOME=/home/ksuser/software/jdk8/jdk1.8.0\_73

export SCALA\_HOME=/home/ksuser/software/scala-2.11.8

export HADOOP\_HOME=/home/ksuser/software/hadoop-2.7.3

export HADOOP\_CONF\_DIR=/home/ksuser/software/hadoop-2.7.3/etc/hadoop

export SPARK\_WORKING\_MEMORY=2g

export SPARK\_MASTER\_IP=x

3，修改slaves

4，sbin下启动

本地模式运行：可以从本地开发软件运行

standalone模式运行：./spark-submit --master spark://localhost:7077 --deploy-mode cluster --class run.SparkSave2ES /home/ksuser/software/knownTest/target/test.jar

yarn模式运行：

模板：

export YARN\_CONF\_DIR=/opt/hadoop/yarn-­‐client/etc/hadoop

bin/spark--submit \

--master yarn--cluster \ 运行模式:local,yarn-­‐clisnt,yarn-­‐cluster

-- class com.hulu.examples.SparkPi \ 应用程序主类

--name sparkpi \ 作业名称

--driver--memory 2g \ Driver需要的内存

--executor--memory 3g \ 每个executor需要的内存

--executor--cores 2 \ 每个executor线程数

-num-executors 2 \ 需启动的Executor总数

$FWDIR/target/scala-­‐2.10/spark-­‐example-­‐assembly-­‐1.0.jar 用户应用程序所在jar包

有些失败的application无法查看 使用命令 bin/yarn logs -applicationId (application id)

./spark-submit --master yarn --deploy-mode cluster --driver-memory 2g --executor-memory 2g --class run.SparkSave2ES /home/ksuser/software/knownTest/target/test.jar

spark on yarn 的history日志查看配置：

1，修改hadoop下的yarn-site文件

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

<property>

<name>yarn.log.server.url</name>

<value>http://localhost:19888/jobhistory/logs</value>

</property>

<property>

<name>yarn.log-aggregation.retain-seconds</name>

<value>259200</value>

</property>

**mapred-site.xml**

<configuration>

  <property>

    <name>mapreduce.framework.name</name>

    <value>yarn</value>

  </property>

<!--日志监控服务的地址，一般填写为nodenode机器地址 -->

<property>

<name>mapreduce.jobhistroy.address</name>

<value>localhost:9000</value>

</property>

<property>

<name>mapreduce.jobhistroy.webapp.address</name>

<value>localhost:19888</value>

</property>

</configuration>

2，在spark-env.sh下添加HADOOP\_CONF\_DIR=HADOOP的配置文件位置

export SPARK\_HISTORY\_OPTS="-Dspark.history.ui.port=7777 -Dspark.history.fs.logDirectory=hdfs://master:9000/directory"

3，修改spark-default

spark.master localhost

spark.yarn.historyServer.address localhost:18080

spark.history.ui.port 18080

spark.eventLog.enabled true

spark.eventLog.dir hdfs://localhost:9000/log/spark

spark.history.fs.logDirectory hdfs://localhost:9000/log/spark

spark.eventLog.compress true

4，在hdfs上创建相应文件夹

5，spark/sbin 下 启动history。 ${SPARK\_HOME}/sbin/start-history-server.sh

6，hadoop/sbin下 ./mr-jobhistory-daemon.sh start historyserver

./yarn-daemon.sh start timelineserver

jdk（版本1.8） scala （版本2.11.8）

ES 集群 （版本6.0.1）

1，下载解压 https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.0.1.tar.gz

2，修改/elasticsearch/config/elasticsearch.yml

cluster.name: cluster-es

# 集群名称

node.name: es-node1

# 节点名称，其余两台为es-node2、es-node3

path.data: /usr/local/elasticsearch/data

# 数据目录

path.logs: /usr/local/elasticsearch/logs

# 日志目录

network.host: 172.16.64.137

# 本机IP

http.port: 9200

# 本机http端口

discovery.zen.minimum\_master\_nodes: 1

# 指定集群中的节点中有几个有master资格的节点

discovery.zen.ping.unicast.hosts: ["172.16.64.137", "172.16.64.138", "172.16.64.147"]

# 指定集群中其他节点的IP

node.master: true

# 是否为master

node.data: false

# 是否为数据节点

discovery.zen.fd.ping\_timeout: 180s

# 设置集群中自动发现其它节点时ping连接超时时间

discovery.zen.fd.ping\_retries: 10

# 集群中节点之间ping的次数

discovery.zen.fd.ping\_interval: 30s

# 集群中节点之间ping的时间间隔

3，启动 nohup /usr/local/elasticsearch/bin/elasticsearch &

安装ES插件 6.0以后的head插件

1，首先要安装nodeJS wget <https://nodejs.org/dist/v8.9.3/node-v8.9.3.tar.gz>

2，安装gcc g++ sudo apt yum install gcc g++

3， ./configure –prefix=/usr/local/node/ node路径

4，make

4，make install

5，添加NODEJS\_HOME环境变量 source

6，node -v

wget https://npm.taobao.org/mirrors/node/latest-v4.x/node-v4.4.7-linux-x64.tar.gz

tar -zxvf node-v4.4.7-linux-x64.tar.gz

7，修改elasticsearch.yml文件

http.cors.enabled: true

http.cors.allow-origin: "\*"

8，安装grunt

cd /opt/elasticsearch-head-master

npm install -g grunt-cli //执行后会生成node\_modules文件夹

grunt -version

9，修改服务器监听地址:Gruntfile.js 添加connect里的hostname

修改连接地址：\_site/app.js 4363行

10，grunt server 启动

Mac版：

Homebrew简称brew，是Mac OSX上的软件包管理工具，能在Mac中方便的安装软件或者卸载软件。   
1.打开终端，执行以下命令安装Homebrew

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

* 1

执行上面命令后会提示输入系统密码，输入密码继续安装。

2.如安装成功则会看到：Installation successful!   
3.此时可以在命令行中输入命令brew进行测试，可以看到如下结果,说明安装成功。

brew install node

执行以下命令查看是否安装成功

node -v:查看node版本

npm -v：查看npm版本

es建表语句 localhost:9200/people

{

"settings":{

"number\_of\_shards":1,

"number\_of\_replicas":1

},

"mappings":{

"clicklog":{

"properties":{

"ip":{

"type":"text"

},

"time":{

"type":"text"

},

"course":{

"type":"text"

},

"statusCode":{

"type":"integer"

},

"referer":{

"type":"text"

}

}

}

}

}

filebeat--→logstash 处理数据:

1，下载好组件，找到数据源

2，配置filebeat.yml

filebeat.prospectors:

- type: log

paths:

- /path/to/file/logstash-tutorial.log

output.logstash:

hosts: ["localhost:5044"]

3，启动filebeat ./filebeat -e -c filebeat.yml -d "publish"

4，编辑一个conf文件 first-pipeline.conf

5，配置 first-pipeline.conf

input {

beats {

port => "5044"

}

}

# The filter part of this file is commented out to indicate that it is

# optional.

# filter {

#

# }

output {

stdout { codec => rubydebug }

}

测试是否通过 logstash -f first-pipeline.conf --config.test\_and\_exit

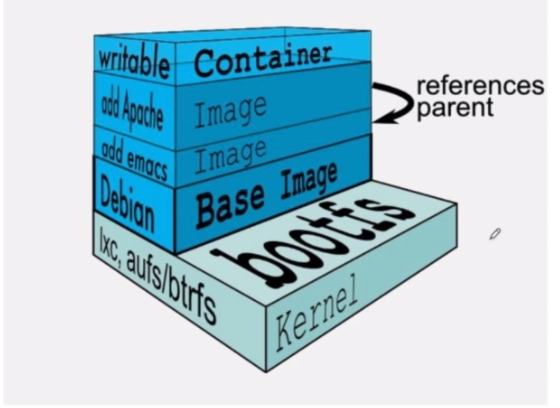
启动并自动重启 logstash -f first-pipeline.conf --config.reload.automatic

Docker

Docker是一个容器，可以把程序等放入其中。基于go语言实现，实现轻量级的操作系统虚拟化解决方案，docker基础是linux容器LXC等技术，在LXC基础上docker进行了进一步的封装，让用户不需要去关心容器的管理，使得操作更为简便，用户操作docker的容器就像操作一个快速轻量级的虚拟机一样。

Build---镜像 ship---仓库 run---容器

镜像从仓库运来，跑的程序形成一个容器



图为一个镜像 文件是分层的只读的

容器可以看成是一个进程

中央仓库。hub.docker.com. 网易蜂巢镜像仓库 c.163.com

Docker pull 拉取远程镜像到本地

Docker images 查看本机都有哪些镜像

docker pull hello-world mac上报错 因为没登陆

docker run [option] image [：tag][command运行什么命令]

docker ps 查看当前docker运行的进程

docker exec -it f4(容器名字) bash

**-t:**在新容器内指定一个伪终端或终端

**-i:**允许你对容器内的标准输入 (STDIN) 进行交互

docker build 。