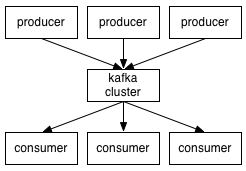
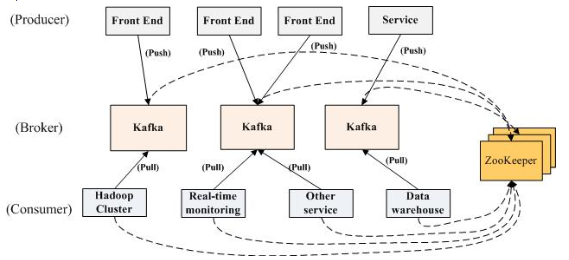
# flume+kafka组合配置应用

## A前言：Kafka概况

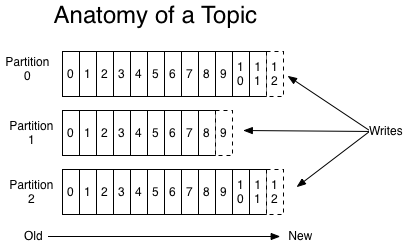
### 1. Kafka组成结构

producer产生数据，传输到brokers\_cluster中；cluster存储数据，以压缩格式在指定时间段内把数据存储到broker的topic的指定partition分区中；consumer拉取数据，每条message只能被每个group中的某一个consumer使用一次。





### 2. topic主题的partition分区结构和message读取顺序



### 3. 需明确的概念

**(1) Kafka中的consumer:**

a) 若干个consumer实例可归并到一个consumer Group中。

b) topic主题中的每个消息message将会被传送给预定消息的consumer Group组中的某一个consumer；consumer实例可以是分布式程序或分布式机器。

c) 如果所有consumer实例都在一个Group中，那就跟传统的队列一样，需在各个consumer之间进行负载均衡；

如果所有consumer实例分布在不同的Group中，那就跟发布-订阅模式一样，所有message消息会广播给所有consumer。

d) kafka 只保证在一个partition分区中顺序取数据；在多个partition分区中取数据时，不能保证所有数据的顺序。

e) 如果要保证全局数据的顺序，则只能有一个partition分区，那样的话就只能有一个consumer了。（partition一定不能比consumer少）

f) Kafka的客户端与服务器通信采用的是TCP协议。

g) 配置多个broker\_id时，最好建多个server.properties，一一对应。

h) 一个message消息只能传输给某一个Group中的某一个consumer，即：一个Group中所有consumer共享一个offset标记位，所以一条message只能供一个Group中的一个consumer消费，之后可再被其他Group中的某一个consumer消费。（**一个Group中的一次性消费**）

(2)partition存储的是什么数据

produce将数据发送给broker中的topic时，会被分发到不同的partition，这样一个topic的完整message被分布到不同的partition上了。

(3)Replication和 Leader election机制

a)每一个partition都有唯一的leader，其他broker作为该partition的follow，所有的读写操作都在leader上完成。所有partition的leader均匀分布在broker上。而Replication就是在leader和follow上进行的完整复制。

### C:\Users\suchang\Documents\Tencent Files\814848668\Image\C2C\7DF87B19679414F5F5A9F80546A0734F.jpg4. 信息处理流程

基本交互原理   
每个Topic被创建后，在zookeeper上存放有其metadata，包含其分区信息、replica信息、LogAndOffset等，默认路径/brokers/topics/<topic\_id>/partitions/<partition\_index>/state；

Producer可以通过zookeeper获得topic的broker信息，从而得知需要往哪写数据。

Consumer也从zookeeper上获得该信息，从而得知要监听哪个partition。

## 一、flume日志发送至kafka指定的topic，作为kafka的producer

### 1. flume日志生成和发送类代码：

**package** com.cmcc.flume\_kafka\_test;

**import** org.apache.log4j.Logger;

**public** **class** SendflumeLog {

**private** **static** Logger *flumelog* = Logger.*getLogger*(SendflumeLog.**class**);

**public** **static** **void** main(String[] args) {

*flumelog*.info("flume-kafka-log-test");

}

}

### 2. 日志生成和发送测试项目中的log4j.properties文件配置：

log4j.rootLogger=info

log4j.logger.com=info, flume

log4j.appender.flume=org.apache.flume.clients.log4jappender.Log4jAppender

log4j.appender.flume.layout=org.apache.log4j.PatternLayout

log4j.appender.flume.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %m%n

log4j.appender.flume.Hostname=192.168.17.128

log4j.appender.flume.Port=41414

此处发送后，如果没有设置flume的sink为logfile，那么不会生成新的日志文件，而是将日志内容直接发送至kafka的指定topic的partition分区。

### 3. 配置flume服务器的flume-conf.properties文件：

#################### producer config #######################

########agent section

producer.sources = s

producer.channels = c

producer.sinks = r

###########source section type = #seq

producer.sources.s.type = avro

producer.sources.s.channels = c

**producer.sources.s.bind = 192.168.17.128**

**producer.sources.s.port = 41414**

############# Each sink's type must be defined

**producer.sinks.r.type = org.apache.flume.plugins.KafkaSink**

**producer.sinks.r.metadata.broker.list=192.168.17.128:9092**

**#producer.sinks.r.zk.connect = 192.168.17.128:2181**

producer.sinks.r.partition.key=0

producer.sinks.r.partitioner.class=org.apache.flume.plugins.SinglePartition

producer.sinks.r.serializer.class=kafka.serializer.StringEncoder

producer.sinks.r.request.required.acks=0

producer.sinks.r.max.message.size=1000000

producer.sinks.r.producer.type=sync

producer.sinks.r.custom.encoding=UTF-8

**producer.sinks.r.custom.topic.name=test**

#topic.name = kafkaToptic

############Specify the channel the sink should use

producer.sinks.r.channel = c

############# Each channel's type is defined.

producer.channels.c.type = memory

producer.channels.c.capacity = 1000

### 4. 启动flume：

进入Apache-flume项目解压包路径，执行：

**>bin/flume-ng agent –n producer –c conf –f conf/flume-conf.properties**

## 二、搭建kafka，接收flume发送过来的日志

### 1. Step 1:下载安装kafka:

> **tar -xzf kafka\_2.9.2-0.8.1.1.tgz**

> **cd kafka\_2.9.2-0.8.1.1**

### 2. 配置kafka: server. properties：

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

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

**broker.id=0**

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

# 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 localhost

**host.name=192.168.17.128**

# 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>**

# 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>**

# The number of threads handling network requests

**num.network.threads=10**

# The number of threads doing disk I/O

**num.io.threads=10**

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

**socket.send.buffer.bytes=1048576**

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

**socket.receive.buffer.bytes=1048576**

# 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. 定义分区数量

**num.partitions=2**

############################# 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。168小时=7天

**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=536870912**

# 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=60000**

# 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=localhost:2181**

# Timeout in ms for connecting to zookeeper

**zookeeper.connection.timeout.ms=1000000**

### 3. 配置zookeeper.properties：

# the directory where the snapshot is stored.

dataDir=/tmp/zookeeper

# the port at which the clients will connect

clientPort=2181

# disable the per-ip limit on the number of connections since this is a non-production config

maxClientCnxns=0

### 4. Step 2: 启动kafka: Start the server

**进入Kafka解压路径，执行启动zookeeper：**

**>bin/zookeeper-server-start.sh config/zookeeper.properties**

**执行启动kafka-server：**

**> bin/kafka-server-start.sh config/server.properties**

### 5. 查看当前kafka中有哪些topic：

> **bin/kafka-topics.sh --list --zookeeper localhost:2181**

### 6. 启动一个 consumer/producer/topic，查看某个topic中的内容：

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

**>bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 1 --topic test2**

**>bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test**

**其中 Leader：指**负责读写指定分区的节点机器。如果有**多个broker保存同一个topic**，那么**同时只能有一个Broker负责该topic的读写**，其它的Broker作为实时备份。负责读写的Broker称为Leader.

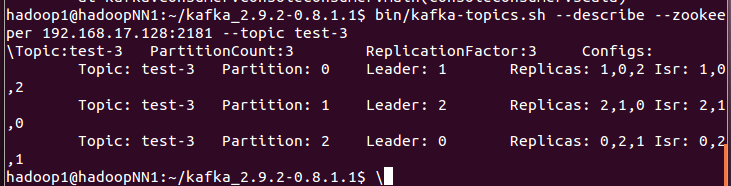
Replicas：指复制该分区Log的节点机器列表；

Isr：指”in-sync” replicas，即当前活跃的副本列表，即当前有效的broker。

### 7. 查看某个topic的属性：

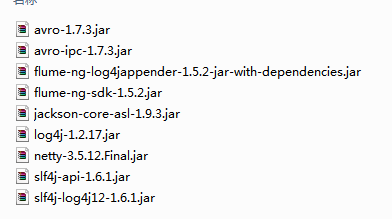
**> /bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic test2**

**一个producer，3个brokers，分区partition=3的情况下：**

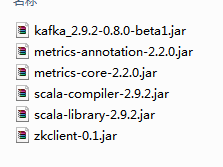


## 三、上述功能需要的依赖包：

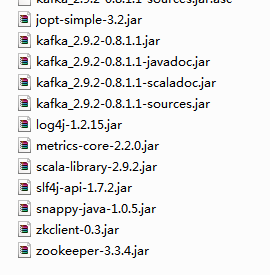
### 1. flume需要的包有：



### 2. flume+Kafka需要的包：



### 3. Kafka需要的包：



## 四、flume接收Kafka 的数据作为source

### 1. flume-conf.properties文件配置

上述我们在flume中配置了一个agent作为Kafka的producer，由flume收集日志数据，sink到Kafka的producer中，由producer将数据push到broker的topic主题partition分区里面。

如果要再将Kafka的broker中的数据作为flume的source数据源，需要再配置一个agent，以broker的topic为数据源，sink到flume的日志文件等地方去。

此时，flume相当于Kafka的consumer。

具体配置如下：

############################################

# consumer config

############################################

consumer.sources = cs

consumer.channels = cc

consumer.sinks = cr

consumer.sources.cs.channels = cc

consumer.sources.cs.type = org.apache.flume.plugins.KafkaSource

consumer.sources.cs.zookeeper.connect=192.168.17.128:2181

consumer.sources.cs.group.id=testGroup

consumer.sources.cs.zookeeper.session.timeout.ms=6000

consumer.sources.cs.zookeeper.sync.time.ms=200

consumer.sources.cs.auto.commit.interval.ms=1000

consumer.sources.cs.custom.topic.name=test

consumer.sources.cs.custom.thread.per.consumer=8

consumer.sinks.cr.type = file\_roll

consumer.sinks.cr.channel = cc

consumer.sinks.cr.sink.directory = /home/hadoop1/apache-flume-1.5.2-bin/logs

consumer.sinks.cr.sink.rollInterval = 3600

consumer.channels.cc.type = memory

consumer.channels.cc.capacity = 100

### 2. 依赖的jar包中

除了上述的之外，还需要在flume的lib目录下添加



## 五、上述配置是单机环境下的配置，整体架构图为：

WEB端

flume agent--

producer

Kafka server

log日志

sink，作为kafka的producer

指定topic-name;

指定topic的partition;

broker\_id;

partition\_num;

host\_name;

port;

flume agent--

consumer

source，作为kafka的consumer

sink

log files

## 六、后期设计架构：

data

Kafka

Storm

log File

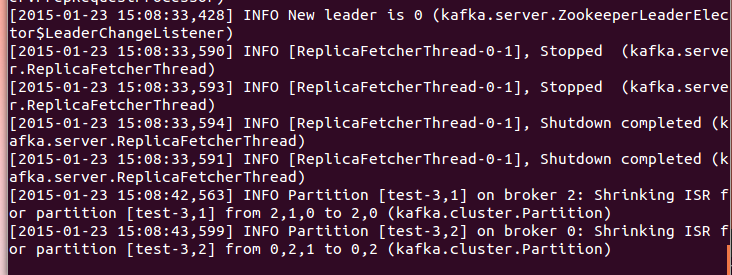
Hadoop

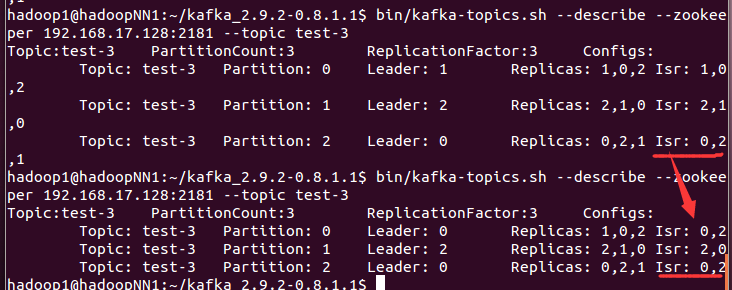
DB

## 七、kafka分布式环境部署

### 1. 一个producer，一个zookeeper，多个brokers时的配置：

当一个leader\_broker挂掉时，zookeeper会重新调整新的leader\_broker，日志显示：





#### a) flume端作为Producer，flume-conf.properties配置：

############################################

# producer config

###########################################

#agent section

producer.sources = s

producer.channels = c

producer.sinks = r

#source section type = #seq

producer.sources.s.type = avro

producer.sources.s.channels = c

producer.sources.s.bind = 192.168.17.128

producer.sources.s.port = 41414

# Each sink's type must be defined

producer.sinks.r.type = org.apache.flume.plugins.KafkaSink

**producer.sinks.r.metadata.broker.list=192.168.17.128:19092,192.168.17.128:9092,192.168.17.128:29092**

#这条配置证实可以删掉，不影响整体功能

~~producer.sinks.r.zk.connect = 192.168.17.128:2181~~

producer.sinks.r.partition.key=0

producer.sinks.r.partitioner.class=org.apache.flume.plugins.SinglePartition

producer.sinks.r.serializer.class=kafka.serializer.StringEncoder

producer.sinks.r.request.required.acks=0

producer.sinks.r.max.message.size=1000000

producer.sinks.r.producer.type=sync

producer.sinks.r.custom.encoding=UTF-8

producer.sinks.r.custom.topic.name=test-3

#topic.name = kafkaToptic

#Specify the channel the sink should use

producer.sinks.r.channel = c

# Each channel's type is defined.

producer.channels.c.type = memory

producer.channels.c.capacity = 10000

#### b) Kafka端作为flume的Sink，一台机器上配多个broker，每个broker对应一个server.properties配置文件：

# 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 #############################

# 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

#localhost

host.name=192.168.17.128

# 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>

# 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>

# The number of threads handling network requests

num.network.threads=10

# The number of threads doing disk I/O

num.io.threads=10

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

socket.send.buffer.bytes=1048576

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

socket.receive.buffer.bytes=1048576

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

socket.request.max.bytes=104857600

#auto.create.topics.enable=false

############################# 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=3

default.replication.factor=3

num.replica.fetchers=2

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

# 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=536870912

# 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=60000

# 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=192.168.17.128:2181

# Timeout in ms for connecting to zookeeper

zookeeper.connection.timeout.ms=45000

### 2. 一个producer，三个（奇数个）zookeeper，三个brokers时的配置：

见博客：http://blog.csdn.net/wangjia184/article/details/37921183

这种架构下，对flume和Kafka的配置跟上述一样，只是需要对zookeepers进行配置。

其中**每个zookeeper配置**：

##################### config begin ##########################

# The number of milliseconds of each tick

#这个时间是作为 Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，每隔这个时间，就会发送一个心跳

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take，配置 Zookeeper 接受Leader 的 Follower 服务器初始化连接时最长能忍受多少个心跳时间间隔数

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement，标识 Leader 与 Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个心跳时间

syncLimit=5

# the directory where the snapshot is stored.需要手动创建好

dataDir=/home/hadoop1/zookeeper-cluster/data/server2

# the port at which the clients will connect，其中同一台机器上的每个zookeeper的Port必须不同，否则冲突

**clientPort=2182**

#server.id=hostname:port1:port2

#port1:communicate with followers as a leader using this port;

#port2:communicate with other servers when chooing a leader

#同一台机器上，这里的两个port必须跟上面的clientPort不同，因为三个port的用途都不一样。第一个端口号表示这个服务器与集群Leader信息交换的端口，第二个端口用于leader挂掉后重新选举的信息交换端口。

**server.1=192.168.17.128:22181:3181**

**server.2=192.168.17.128:22182:4181**

**server.3=192.168.17.128:22183:5181**

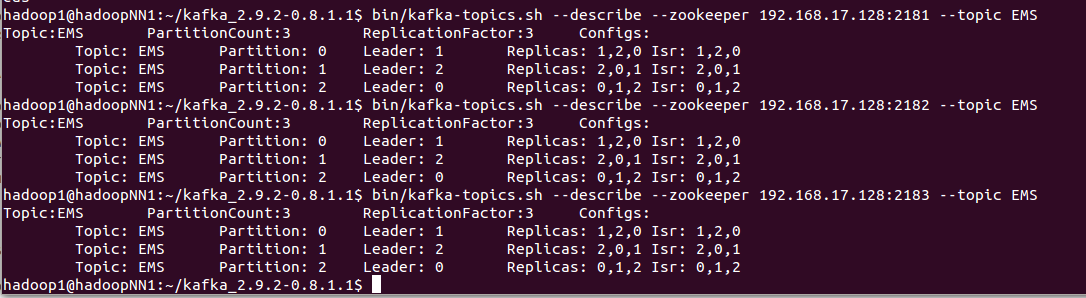
#################### config end ##################################

**注：同步 zookeeper 的配置以及相关目录到其他两台服务中：**

scp -r /apps/svr/zookeeper-3.4.5 solr@192.168.0.3:/apps/svr

scp -r /apps/data/zookeeper-data solr@192.168.0.3:/apps/data

三个zookeeper节点机器上都保存了相同的brokers、partition、topic等信息：



### 3. 多个zookeepers时，flume作为consumer消费kafka的数据，flume-consumer.properties配置：

############################################

# consumer config

############################################

consumer.sources = cs

consumer.channels = cc

consumer.sinks = cr

consumer.sources.cs.channels = cc

consumer.sources.cs.type = org.apache.flume.plugins.KafkaSource

**consumer.sources.cs.zookeeper.connect=192.168.17.128:2181,192.168.17.128:2182,192.168.17.128:2183**

consumer.sources.cs.group.id=testGroup

consumer.sources.cs.zookeeper.session.timeout.ms=15000

consumer.sources.cs.zookeeper.sync.time.ms=6000

consumer.sources.cs.auto.commit.interval.ms=1000

consumer.sources.cs.custom.topic.name=**test-3**

consumer.sources.cs.custom.thread.per.consumer=8

consumer.sinks.cr.type = file\_roll

consumer.sinks.cr.channel = cc

consumer.sinks.cr.sink.directory = /home/hadoop1/apache-flume-1.5.2-bin/logs

consumer.sinks.cr.sink.rollInterval = 3600

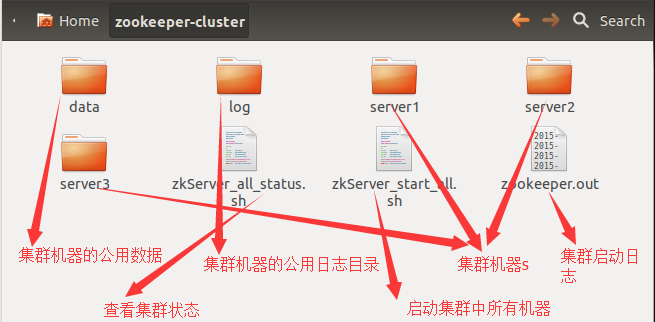
consumer.channels.cc.type = memory

consumer.channels.cc.capacity = 10000

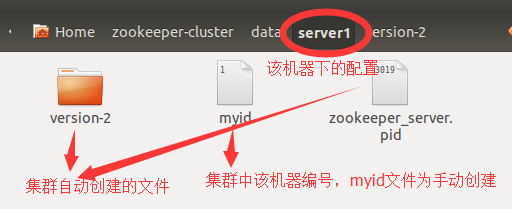
由于Zookeeper是快速失败（fail-fast)的，且遇到任何错误情况，进程均会退出，因此，最好能通过监控程序将Zookeeper管理起来，保证Zookeeper退出后能被自动重启

Zookeeper运行过程中会在dataDir目录下生成很多日志和快照文件，而Zookeeper运行进程并不负责定期清理合并这些文件，导致占用大量磁盘空间，因此，需要通过cron等方式定期清除没用的日志和快照文件。

集群搭建：



集群中某个机器的配置：



### 4. 启动zookeeper集群：

sh server1/bin/zkServer.sh start &

echo "zookeeper\_1 starting------------------"

sh server2/bin/zkServer.sh start &

echo "zookeeper\_2 starting------------------"

sh server3/bin/zkServer.sh start &

echo "zookeeper\_3 starting------------------"

### 5. 查看zookeeper集群状态：

sh server1/bin/zkServer.sh status

sh server2/bin/zkServer.sh status

sh server3/bin/zkServer.sh status

### 6. Kafka自动生成flume日志的topic主题

集群中，为flume配置多个Sink，其中一个存储到file文件中，一个存储到Kafka中，并根据flume日志信息自动生成topic。

这种拓扑结构下，是由一个Source对应多个Channel，其中每个Channel对应一个Sink。

对应的flume配置文件如下。

############################################

# producer config

###########################################

#agent section

producer.sources = s

producer.channels = c file\_c

producer.sinks = r file\_r

#----------- another channel and sink -----------------------

#producer.sinks=file\_r

#producer.channels=file\_c

producer.sinks.file\_r.type = file\_roll

producer.sinks.file\_r.channel = file\_c

producer.sinks.file\_r.sink.directory = /home/hadoop1/apache-flume-1.5.2-bin/logs

producer.sinks.file\_r.sink.rollInterval = 36000

producer.channels.file\_c.type = memory

producer.channels.file\_c.capacity = 10000

producer.sources.s.type = avro

producer.sources.s.channels = c file\_c

producer.sources.s.bind = 192.168.17.128

producer.sources.s.port = 41414

# Each sink's type must be defined

producer.sinks.r.type = org.apache.flume.plugins.KafkaSink

producer.sinks.r.metadata.broker.list=192.168.17.128:19092,192.168.17.128:9092,192.168.17.128:29092

# -----------------------------------------

#### producer.sinks.r.zk.connect = 192.168.17.128:2181

# -----------------------------------------

producer.sinks.r.partition.key=0

producer.sinks.r.partitioner.class=org.apache.flume.plugins.SinglePartition

producer.sinks.r.serializer.class=kafka.serializer.StringEncoder

producer.sinks.r.request.required.acks=0

producer.sinks.r.max.message.size=1000000

producer.sinks.r.producer.type=sync

producer.sinks.r.custom.encoding=UTF-8

#-----------------------------------------

# producer.sinks.r.custom.topic.name=20150313-test

#### topic.name = kafkaToptic

# -----------------------------------------

#Specify the channel the sink should use

producer.sinks.r.channel = c

# Each channel's type is defined.

producer.channels.c.type = memory

producer.channels.c.capacity = 10000

**flumeng-kafka-plugin插件中的KafkaSink源码修改时，从日志数据eventData中截取最后一位（标识日志来源）作为这条日志的topic主题。**

## 八、Kafka+storm环境部署

### 1. 安装storm

storm的安装和单机版也一样，但比较纠结的是storm的配置文件storm.yaml。当然，这也是每台机器都要配的。

这个脚本文件写的不咋地，所以在配置时一定注意在每一项的开始时要加空格，冒号后也必须要加空格，否则storm就不认识这个配置文件了，切记切记。

storm.yaml配置文件：

########### These MUST be filled in for a storm configuration#############

storm.zookeeper.servers:

- "192.168.17.128"

# - "server2"

#

storm.zookeeper.port: 2182

nimbus.host: "192.168.17.128"

storm.local.dir: "/home/hadoop1/storm-cluster/server2/data"

supervisor.slots.ports:

- 6700

- 6701

- 6702

- 6703

#

########### end #############

向storm 提交Topology时，需要将Topology打包成jar文件，这个过程可以用Maven去构建。

### 2. 启动storm节点

bin/storm nimbus &

bin/storm supervisor &

查看storm状态，启动storm ui：

bin/storm ui &

### 3. 提交Topology

上面，已经使用Maven构建storm-starter工程，在target目录下生成一个jar文件，然后将该storm-starter工程中的WordCountTopology提交到Nimbus，执行如下命令：

bin/storm jar ../storm-starter/target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.WordCountTopology myFirstStormApp

或者：

bin/storm jar /home/storm/storm-examples-0.0.1-SNAPSHOT.jar storm\_kafka\_start.storm\_kafka\_topology.testStormKafkaTopology h1

其中h1表示Nimbus的主机名

这时，可以通过查看worker的日志，来确定我们提交的Topology的执行情况：

[shirdrn@nn storm-0.8.1]$ tail -100f logs/worker-6700.log

可以通过如下命令查看Storm的管理操作命令：

bin/storm help

查询当前运行的Topology：

bin/storm list

杀掉运行中的Topology：

bin/storm kill myFirstStormApp