第 2 章 Kafka 集群部署

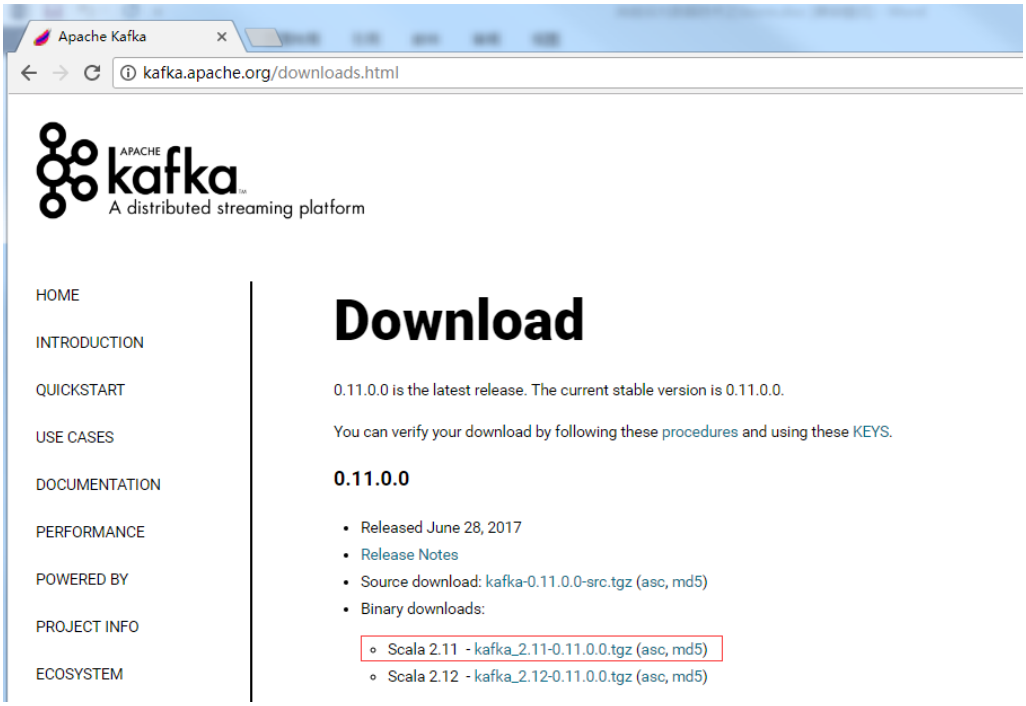
2.1 环境准备

2.1.1 集群规划

hadoop102	hadoop103	hadoop104
zk	zk	zk
kafka	kafka	kafka

2.1.2 jar 包下载

http://kafka.apache.org/downloads.html



2.1.3 虚拟机准备

1）准备 3 台虚拟机

2）配置 ip 地址


尚硅谷大数据技术
之修改为静态ip.do

3）配置主机名称


尚硅谷大数据技术
之修改主机名.doc

4）3 台主机分别关闭防火墙

[root@hadoop102 atguigu]# chkconfig iptables off

[root@hadoop103 atguigu]# chkconfig iptables off

[root@hadoop104 atguigu]# chkconfig iptables off

2.1.4 安装 jdk

尚硅谷大数据技术之安装jdk.doc

2.1.5 安装 Zookeeper

0）集群规划

在 hadoop102、hadoop103 和 hadoop104 三个节点上部署 Zookeeper。

1）解压安装

（1）解压 zookeeper 安装包到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.4.10.tar.gz -C /opt/module/

（2）在/opt/module/zookeeper-3.4.10/这个目录下创建 zkData

mkdir -p zkData

（3）重命名/opt/module/zookeeper-3.4.10/conf 这个目录下的 zoo_sample.cfg 为 zoo.cfg

mv zoo_sample.cfg zoo.cfg

2）配置 zoo.cfg 文件

（1）具体配置

dataDir=/opt/module/zookeeper-3.4.10/zkData

增加如下配置

#######################cluster##########################

server.2=hadoop102:2888:3888

server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

（2）配置参数解读

Server.A=B:C:D。

A 是一个数字，表示这个是第几号服务器；

B 是这个服务器的 ip 地址；

C 是这个服务器与集群中的 Leader 服务器交换信息的端口；

D 是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。

集群模式下配置一个文件 myid，这个文件在 dataDir 目录下，这个文件里面有一个数据就是 A 的值，Zookeeper 启动时读取此文件，拿到里面的数据与 zoo.cfg 里面的配置信息比较从而判断到底是哪个 server。

3）集群操作

（1）在/opt/module/zookeeper-3.4.10/zkData 目录下创建一个 myid 的文件

touch myid

添加 myid 文件，注意一定要在 linux 里面创建，在 notepad++里面很可能乱码

（2）编辑 myid 文件

vi myid

在文件中添加与 server 对应的编号：如 2

（3）拷贝配置好的 zookeeper 到其他机器上

scp -r zookeeper-3.4.10/ root@hadoop103.atguigu.com:/opt/app/

scp -r zookeeper-3.4.10/ root@hadoop104.atguigu.com:/opt/app/

并分别修改 myid 文件中内容为 3、4

（4）分别启动 zookeeper

[root@hadoop102 zookeeper-3.4.10]# bin/zkServer.sh start

[root@hadoop103 zookeeper-3.4.10]# bin/zkServer.sh start

[root@hadoop104 zookeeper-3.4.10]# bin/zkServer.sh start

（5）查看状态

[root@hadoop102 zookeeper-3.4.10]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

[root@hadoop103 zookeeper-3.4.10]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: leader

[root@hadoop104 zookeeper-3.4.5]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

2.2 Kafka 集群部署

1）解压安装包

[atguigu@hadoop102 software]$ tar -zxvf kafka_2.11-0.11.0.0.tgz -C /opt/module/

2）修改解压后的文件名称

[atguigu@hadoop102 module]$ mv kafka_2.11-0.11.0.0/ kafka

3）在/opt/module/kafka 目录下创建 logs 文件夹

[atguigu@hadoop102 kafka]$ mkdir logs

4）修改配置文件

[atguigu@hadoop102 kafka]$ cd config/

[atguigu@hadoop102 config]$ vi server.properties

输入以下内容：

```
#broker 的全局唯一编号，不能重复
broker.id=0
#删除 topic 功能使能
delete.topic.enable=true
#处理网络请求的线程数量
num.network.threads=3
#用来处理磁盘 IO 的线程数量
num.io.threads=8
#发送套接字的缓冲区大小
socket.send.buffer.bytes=102400
#接收套接字的缓冲区大小
socket.receive.buffer.bytes=102400
```

```
#请求套接字的缓冲区大小
socket.request.max.bytes=104857600
#kafka 运行日志存放的路径
log.dirs=/opt/module/kafka/logs
#topic 在当前 broker 上的分区个数
num.partitions=1
#用来恢复和清理 data 下数据的线程数量
num.recovery.threads.per.data.dir=1
#segment 文件保留的最长时间，超时将被删除
log.retention.hours=168
#配置连接 Zookeeper 集群地址
zookeeper.connect=hadoop102:2181,hadoop103:2181,hadoop104:2181
```

5）配置环境变量

[root@hadoop102 module]# vi /etc/profile

```
#KAFKA_HOME
export KAFKA_HOME=/opt/module/kafka
export PATH=$PATH:$KAFKA_HOME/bin
```

[root@hadoop102 module]# source /etc/profile

6）分发安装包

[root@hadoop102 etc]# xsync profile

[atguigu@hadoop102 module]$ xsync kafka/

7）分别在 hadoop103 和 hadoop104 上修改配置文件/opt/module/kafka/config/server.properties 中的 broker.id=1、broker.id=2

注：broker.id 不得重复

8）启动集群

依次在 hadoop102、hadoop103、hadoop104 节点上启动 kafka

[atguigu@hadoop102 kafka]$ bin/kafka-server-start.sh config/server.properties &

[atguigu@hadoop103 kafka]$ bin/kafka-server-start.sh config/server.properties &

[atguigu@hadoop104 kafka]$ bin/kafka-server-start.sh config/server.properties &

9）关闭集群

[atguigu@hadoop102 kafka]$ bin/kafka-server-stop.sh stop

[atguigu@hadoop103 kafka]$ bin/kafka-server-stop.sh stop

[atguigu@hadoop104 kafka]$ bin/kafka-server-stop.sh stop

2.3 Kafka 命令行操作

1）查看当前服务器中的所有 topic

[atguigu@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181 --list

2）创建 topic

[atguigu@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181 --create --replication-factor 3 --partitions 1 --topic first

选项说明：

--topic 定义 topic 名

--replication-factor 定义副本数

--partitions 定义分区数

3）删除 topic

[atguigu@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181 --delete --topic first

需要 server.properties 中设置 delete.topic.enable=true 否则只是标记删除或者直接重启。

4）发送消息

[atguigu@hadoop102 kafka]$ bin/kafka-console-producer.sh --broker-list hadoop102:9092 --topic first

>hello world

>atguigu atguigu

5）消费消息

[atguigu@hadoop103 kafka]$ bin/kafka-console-consumer.sh --zookeeper hadoop102:2181 --from-beginning --topic first

--from-beginning：会把 first 主题中以往所有的数据都读取出来。根据业务场景选择是否增加该配置。

6）查看某个 Topic 的详情

```
[atguigu@hadoop102 kafka]$ bin/kafka-topics.sh --zookeeper hadoop102:2181 --describe
--topic first
```

2.4 Kafka 配置信息

2.4.1 Broker 配置信息

属性	默认值	描述
broker.id		必填参数，broker 的唯一标识
log.dirs	/tmp/kafka-logs	Kafka 数据存放的目录。可以指定多个目录，中间用逗号分隔，当新 partition 被创建的时会被存放到当前存放 partition 最少的目录。
port	9092	BrokerServer 接受客户端连接的端口号
zookeeper.connect	null	Zookeeper 的连接串，格式为：hostname1:port1,hostname2:port2,hostname3:port3。可以填一个或多个，为了提高可靠性，建议都填上。注意，此配置允许我们指定一个 zookeeper 路径来存放此 kafka 集群的所有数据，为了与其他应用集群区分开，建议在此配置中指定本集群存放目录，格式为：hostname1:port1,hostname2:port2,hostname3:port3/chroot/path 。需要注意的是，消费者的参数要和此参数一致。
message.max.bytes	1000000	服务器可以接收到的最大的消息大小。注意此参数要和 consumer 的 maximum.message.size 大小一致，否则会因为生产者生产的消息太大导致消费者无法消费。
num.io.threads	8	服务器用来执行读写请求的 IO 线程数，此参

		数的数量至少要等于服务器上磁盘的数量。
queued.max.requests	500	I/O 线程可以处理请求的队列大小，若实际请求数超过此大小，网络线程将停止接收新的请求。
socket.send.buffer.bytes	100 * 1024	The SO_SNDBUFF buffer the server prefers for socket connections.
socket.receive.buffer.bytes	100 * 1024	The SO_RCVBUFF buffer the server prefers for socket connections.
socket.request.max.bytes	100 * 1024 * 1024	服务器允许请求的最大值， 用来防止内存溢出，其值应该小于 Java heap size.
num.partitions	1	默认 partition 数量，如果 topic 在创建时没有指定 partition 数量，默认使用此值，建议改为 5
log.segment.bytes	1024 * 1024 * 1024	Segment 文件的大小，超过此值将会自动新建一个 segment，此值可以被 topic 级别的参数覆盖。
log.roll.{ms,hours}	24 * 7 hours	新建 segment 文件的时间，此值可以被 topic 级别的参数覆盖。
log.retention.{ms,minutes,hours}	7 days	Kafka segment log 的保存周期，保存周期超过此时间日志就会被删除。此参数可以被 topic 级别参数覆盖。数据量大时，建议减小此值。
log.retention.bytes	-1	每个 partition 的最大容量，若数据量超过此

		值，partition 数据将会被删除。注意这个参数控制的是每个 partition 而不是 topic。此参数可以被 log 级别参数覆盖。
log.retention.check.interval.ms	5 minutes	删除策略的检查周期
auto.create.topics.enable	true	自动创建 topic 参数，建议此值设置为 false，严格控制 topic 管理，防止生产者错写 topic。
default.replication.factor	1	默认副本数量，建议改为 2。
replica.lag.time.max.ms	10000	在此窗口时间内没有收到 follower 的 fetch 请求，leader 会将其从 ISR(in-sync replicas)中移除。
replica.lag.max.messages	4000	如果 replica 节点落后 leader 节点此值大小的消息数量，leader 节点就会将其从 ISR 中移除。
replica.socket.timeout.ms	30 * 1000	replica 向 leader 发送请求的超时时间。
replica.socket.receive.buffer.bytes	64 * 1024	The socket receive buffer for network requests to the leader for replicating data.
replica.fetch.max.bytes	1024 * 1024	The number of byes of messages to attempt to fetch for each partition in the fetch requests the replicas send to the leader.
replica.fetch.wait.max.ms	500	The maximum amount of time to wait

		time for data to arrive on the leader in the fetch requests sent by the replicas to the leader.
num.replica.fetchers	1	Number of threads used to replicate messages from leaders. Increasing this value can increase the degree of I/O parallelism in the follower broker.
fetch.purgatory.purge.interval.requests	1000	The purge interval (in number of requests) of the fetch request purgatory.
zookeeper.session.timeout.ms	6000	ZooKeeper session 超时时间。如果在此时间内 server 没有向 zookeeper 发送心跳，zookeeper 就会认为此节点已挂掉。 此值太低导致节点容易被标记死亡；若太高，.会导致太迟发现节点死亡。
zookeeper.connection.timeout.ms	6000	客户端连接 zookeeper 的超时时间。
zookeeper.sync.time.ms	2000	ZK follower 落后 ZK leader 的时间。
controlled.shutdown.enable	true	允许 broker shutdown。如果启用，broker 在关闭自己之前会把它上面的所有 leaders 转移到其它 brokers 上，建议启用，增加集群稳定性。
auto.leader.rebalance.en	true	If this is enabled the controller will

able		automatically try to balance leadership for partitions among the brokers by periodically returning leadership to the "preferred" replica for each partition if it is available.
leader.imbalance.per.broker.percentage	10	The percentage of leader imbalance allowed per broker. The controller will rebalance leadership if this ratio goes above the configured value per broker.
leader.imbalance.check.interval.seconds	300	The frequency with which to check for leader imbalance.
offset.metadata.max.bytes	4096	The maximum amount of metadata to allow clients to save with their offsets.
connections.max.idle.ms	600000	Idle connections timeout: the server socket processor threads close the connections that idle more than this.
num.recovery.threads.per.data.dir	1	The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.
unclean.leader.election.enable	true	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though doing so may

		result in data loss.
delete.topic.enable	false	启用 deletetopic 参数，建议设置为 true。
offsets.topic.num.partitions	50	The number of partitions for the offset commit topic. Since changing this after deployment is currently unsupported, we recommend using a higher setting for production (e.g., 100-200).
offsets.topic.retention.minutes	1440	Offsets that are older than this age will be marked for deletion. The actual purge will occur when the log cleaner compacts the offsets topic.
offsets.retention.check.interval.ms	600000	The frequency at which the offset manager checks for stale offsets.
offsets.topic.replication.factor	3	The replication factor for the offset commit topic. A higher setting (e.g., three or four) is recommended in order to ensure higher availability. If the offsets topic is created when fewer brokers than the replication factor then the offsets topic will be created with fewer replicas.
offsets.topic.segment.bytes	104857600	Segment size for the offsets topic. Since it uses a compacted topic, this should be

		kept relatively low in order to facilitate faster log compaction and loads.
offsets.load.buffer.size	5242880	An offset load occurs when a broker becomes the offset manager for a set of consumer groups (i.e., when it becomes a leader for an offsets topic partition). This setting corresponds to the batch size (in bytes) to use when reading from the offsets segments when loading offsets into the offset manager’s cache.
offsets.commit.required.acks	-1	The number of acknowledgements that are required before the offset commit can be accepted. This is similar to the producer’s acknowledgement setting. In general, the default should not be overridden.
offsets.commit.timeout.ms	5000	The offset commit will be delayed until this timeout or the required number of replicas have received the offset commit. This is similar to the producer request timeout.

2.4.2 Producer 配置信息

属性	默认值	描述
metadata.broker.list		启动时 producer 查询 brokers 的列表，可以是集群中所有 brokers 的一个子集。注意，这个参数只是用来获取 topic 的元信息用，producer 会从元信息中挑选合适的 broker 并与之建立 socket 连接。格式是：host1:port1,host2:port2。
request.required.acks	0	参见 3.2 节介绍
request.timeout.ms	10000	Broker 等待 ack 的超时时间，若等待时间超过此值，会返回客户端错误信息。
producer.type	sync	同步异步模式。async 表示异步，sync 表示同步。如果设置成异步模式，可以允许生产者以 batch 的形式 push 数据，这样会极大的提高 broker 性能，推荐设置为异步。
serializer.class	kafka.serializer.DefaultEncoder	序列号类，.默认序列化成 byte[] 。
key.serializer.class		Key 的序列化类，默认同上。
partitioner.class	kafka.producer.DefaultPar	Partition 类，默认对 key 进行 hash。

	titioner	
compression.codec	none	指定 producer 消息的压缩格式，可选参数为：“none”，“gzip” and “snappy”。关于压缩参见 4.1 节
compressed.topics	null	启用压缩的 topic 名称。若上面参数选择了一个压缩格式，那么压缩仅对本参数指定的 topic 有效，若本参数为空，则对所有 topic 有效。
message.send.max.retries	3	Producer 发送失败时重试次数。若网络出现问题，可能会导致不断重试。
retry.backoff.ms	100	Before each retry, the producer refreshes the metadata of relevant topics to see if a new leader has been elected. Since leader election takes a bit of time, this property specifies the amount of time that the producer waits before refreshing the metadata.
topic.metadata.refresh.interval.ms	600 * 1000	The producer generally refreshes the topic metadata from brokers when there is a failure (partition missing, leader not available...). It will also poll regularly (default: every 10min so 600000ms). If you set this to a negative value, metadata

		will only get refreshed on failure. If you set this to zero, the metadata will get refreshed after each message sent (not recommended). Important note: the refresh happen only AFTER the message is sent, so if the producer never sends a message the metadata is never refreshed
queue.buffering.max.ms	5000	启用异步模式时，producer 缓存消息的时间。比如我们设置成 1000 时，它会缓存 1 秒的数据再一次发送出去，这样可以极大的增加 broker 吞吐量，但也会造成时效性的降低。
queue.buffering.max.messages	10000	采用异步模式时 producer buffer 队列里最大缓存的消息数量，如果超过这个数值，producer 就会阻塞或者丢掉消息。
queue.enqueue.timeout.ms	-1	当达到上面参数值时 producer 阻塞等待的时间。如果值设置为 0，buffer 队列满时 producer 不会阻塞，消息直接被丢掉。若值设置为-1，producer 会被阻塞，不会丢消息。
batch.num.messages	200	采用异步模式时，一个 batch 缓存的消息数量。达到这个数量值时 producer 才会发送消息。
send.buffer.bytes	100 * 1024	Socket write buffer size

client.id	“”	The client id is a user-specified string sent in each request to help trace calls. It should logically identify the application making the request.

2.4.3 Consumer 配置信息

属性	默认值	描述
group.id		Consumer 的组 ID，相同 goup.id 的 consumer 属于同一个组。
zookeeper.connect		Consumer 的 zookeeper 连接串，要和 broker 的配置一致。
consumer.id	null	如果不设置会自动生成。
socket.timeout.ms	30 * 1000	网络请求的 socket 超时时间。实际超时时间由 max.fetch.wait + socket.timeout.ms 确定。
socket.receive.buffer.bytes	64 * 1024	The socket receive buffer for network requests.
fetch.message.max.bytes	1024 * 1024	查询 topic-partition 时允许的最大消息大小。consumer 会为每个 partition 缓存此大小的消息到内存，因此，这个参数可以控制 consumer 的内存使用量。这个值应该至少比 server 允许的最大消息大小大，以免 producer 发送的消息大于 consumer 允许的消息。

num.consumer.fetchers	1	The number fetcher threads used to fetch data.
auto.commit.enable	true	如果此值设置为 true，consumer 会周期性的把当前消费的 offset 值保存到 zookeeper。当 consumer 失败重启之后将会使用此值作为新开始消费的值。
auto.commit.interval.ms	60 * 1000	Consumer 提交 offset 值到 zookeeper 的周期。
queued.max.message.chunks	2	用来被 consumer 消费的 message chunks 数量，每个 chunk 可以缓存 fetch.message.max.bytes 大小的数据量。
auto.commit.interval.ms	60 * 1000	Consumer 提交 offset 值到 zookeeper 的周期。
queued.max.message.chunks	2	用来被 consumer 消费的 message chunks 数量，每个 chunk 可以缓存 fetch.message.max.bytes 大小的数据量。
fetch.min.bytes	1	The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request.
fetch.wait.max.ms	100	The maximum amount of time the server

		will block before answering the fetch request if there isn’ t sufficient data to immediately satisfy fetch.min.bytes.
rebalance.backoff.ms	2000	Backoff time between retries during rebalance.
refresh.leader.backoff.ms	200	Backoff time to wait before trying to determine the leader of a partition that has just lost its leader.
auto.offset.reset	largest	What to do when there is no initial offset in ZooKeeper or if an offset is out of range ;smallest : automatically reset the offset to the smallest offset; largest : automatically reset the offset to the largest offset;anything else: throw exception to the consumer
consumer.timeout.ms	-1	若在指定时间内没有消息消费，consumer 将会抛出异常。
exclude.internal.topics	true	Whether messages from internal topics (such as offsets) should be exposed to the consumer.
zookeeper.session.timeout.ms	6000	ZooKeeper session timeout. If the consumer fails to heartbeat to ZooKeeper

		for this period of time it is considered dead and a rebalance will occur.
zookeeper.connection.timeout.ms	6000	The max time that the client waits while establishing a connection to zookeeper.
zookeeper.sync.time.ms	2000	How far a ZK follower can be behind a ZK leader