1. Canal Server

Canal server的高可用方案是主备方式，正常情况下主备server都启动，但是只有主server能启动instance去监听binlog。当主server挂掉之后，zookeeper会将该server标记为不可用，其它备canal server会监听到zookeeper上的这个变化，就运行自己的instance 取代原先的主instance监听binlog。

配置时，只需要把主备server的canal.zkServers配成一样，然后canal.id不重复即可。

先启动的server为主，后为备。

1. Canal Client

Canal Client的高可用方案是主备方式，正常情况下主备Client都启动，但是只有主Client能去访问Canal server读取binlog。canal client允许同时启动多个canal client，通过running机制，可保证只有一个client在工作，其他client做为冷备. 当运行中的client挂了，running会控制让冷备中的client转为工作模式，这样就可以确保canal client也不会是单点. 保证整个系统的高可用性。

配置时，需要将主备Client原来的canal.client.hostname和canal.client.port配置去掉，把canal.zkServers配成一样即可。

先启动的Client为主，后为备。

1. Kafka Server

Kafka server的高可用方案是集群的方式。

在kafka中，每个topic都有可以定义数量的分区，topic里面的消息会被存储到这些分区中。而每个分区都有且只会有一个leader，所有对该分区的produce和consume都通过该leader来完成。同时会有若干个replica来复制消息，和leader保持同步，以保证当leader挂了之后，消息不会丢失。

集群的方式就是启动若干kafka server后，通过zookeeper来自动分配或者人为手动分配每个分区的leader和replica到集群中的server。

当集群中的某台kafka server挂掉后，zookeeper会将该server标记为不可用，集群中消息和已经挂掉的server保持同步的replica会顶上，取代已经挂掉的server成为之前被分配给该server的分区新的leader。

当已经挂掉的server再次加入时，该server会首先充当replica，去复制正在运行的leader的消息，当保证消息同步之后，再按照原先定义的分配规则，新的server再次成为某些分区的leader。

配置时，只需要把所有server的zookeeper.connect配成一样，broker.id不重复即可。

其中default.replication.factor是定义replica的数量，Replica过少会影响数据的可用性，太多则会白白浪费存储资源，一般建议在2~3为宜。

num.partitions是定义每个topic默认分区数量，集群时可以适当配大点，但也不要太大，太多分区会增加rebalance的时间，rebalance时间就是当某个server或者consumer挂掉之后，重新分配的时间，这段时间内kafka服务是不可用的。

1. Kafka Consumer

Kafka server的高可用方案是仿集群的方式。

如上所述，分区是kafka server存储topic消息的最小单位。

每个consumer group都可以独立的消费topic中的所有分区消息。

一个 consumer group中可以有若干个consumer。

一个group中的consumer可能会消费多个分区中的消息，这取决于group中consumer数量和topic的分区数量的大小关系。但是，一个分区中的消息，只能被同一个group中的唯一一个consumer消费。这样才能保证同一个分区中消息的被消费的有序性，同时也保证消息不会被重复消费。

也就是说，同一个group里面的consumer，都会被分配去单独消费某些分区中的消息。他们之间是可以水平扩展的，同一个group中的consumer数量越多，对应每个consumer需要消费的分区数量就越少，从而减轻每个consumer负担，提高并发能力。

同一个group中consumer的新增或者减少，kafka都会通过rebalance机制，重新为当前可用的consumer分配分区，保证每个分区都会有consumer去消费。

利用这个扩展性，集群时的每台tomcat都去启动group相同的consumer，即集群中的每台tomcat启动的consumer都足以单独的正确消费完topic里面的消息。增加集群tomcat的数量只是为了减轻每个consumer负担，提高并发能力，同时，当集群中的某些consumer挂掉，其它tomcat中的consumer会顶上，保证整个消费服务不会挂掉。

因为key是每个consumer的唯一标识，所以要保证每台tomcat启动的consumer的key相互之间都是不一样的。

怎么能让不同tomcat启动不同的consumer呢？

目前的方案是，在启动的时候，不同的tomcat通过不同的脚本来build，然后在不同的build脚本里面传入不同的consumer pool name。

因为consumer是从数据库通过pool name来取出，然后通过tomcat启动的。所以集群中的每个tomcat都读取不同的pool name里面的consumer即可保证consumer key不重复。

每个pool name里面定义的consumer，除了key不同，理论上其它的都要相同，而且都应该包含所有的group。要保证如果只有单个pool name里面定义的consumer启动时足以正确完成消费。

配置要点：

1、t\_mq\_consumer表里面consumer的pool\_name要加上每台服务的标识，执行consumer sql.txt即可

1. 集群中有要启动几台tomcat，就应该有几个build脚本，每个脚本里面传入对应的pool\_name
2. Build-ix.sh里面加入serverNo配置，接收build脚本传入的pool\_name参数

4、Pom.xml中加入

<properties>

<kafka.consumer.serverNo>${serverNo}</kafka.consumer.serverNo>

</properties>

1. Init\_xxx.properties中配data.sync.pool\_name=data\_sync\_${kafka.consumer.serverNo}，然后kafka.bootstrapServers有多少个kafka server最好都写完。
2. 注意，build的时候，要执行标号和workspace里面项目名称所带编号相同的build脚本。e.g：如果项目名为data\_sync\_2，则执行./build-ix-aws-2.sh。如果项目名为data\_sync，则执行./build-ix-aws.sh。这点很重要！！！

因为这里的编号对应t\_mq\_consumer表里面pool\_name定义的服务池编号。