**Python操作分布式流处理系统Kafka**

**什么是Kafka**

Kafka是一个分布式流处理系统，流处理系统使它可以像消息队列一样publish或者subscribe消息，分布式提供了容错性，并发处理消息的机制。

**Kafka的基本概念**

kafka运行在集群上，集群包含一个或多个服务器。kafka把消息存在topic中，每一条消息包含键值（key），值（value）和时间戳（timestamp）。

kafka有以下一些基本概念：

Producer - 消息生产者，就是向kafka broker发消息的客户端。

Consumer - 消息消费者，是消息的使用方，负责消费Kafka服务器上的消息。

Topic - 主题，由用户定义并配置在Kafka服务器，用于建立Producer和Consumer之间的订阅关系。生产者发送消息到指定的Topic下，消息者从这个Topic下消费消息。

Partition - 消息分区，一个topic可以分为多个 partition，每个

partition是一个有序的队列。partition中的每条消息都会被分配一个有序的

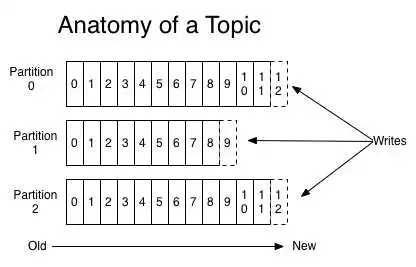
id（offset）。

Broker - 一台kafka服务器就是一个broker。一个集群由多个broker组成。一个broker可以容纳多个topic。

Consumer Group - 消费者分组，用于归组同类消费者。每个consumer属于一个特定的consumer group，多个消费者可以共同消息一个Topic下的消息，每个消费者消费其中的部分消息，这些消费者就组成了一个分组，拥有同一个分组名称，通常也被称为消费者集群。

Offset - 消息在partition中的偏移量。每一条消息在partition都有唯一的偏移量，消息者可以指定偏移量来指定要消费的消息。

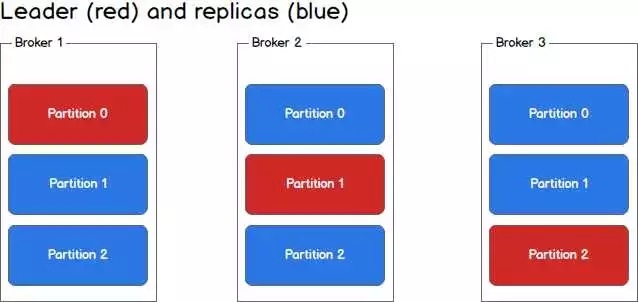
**Kafka分布式架构**



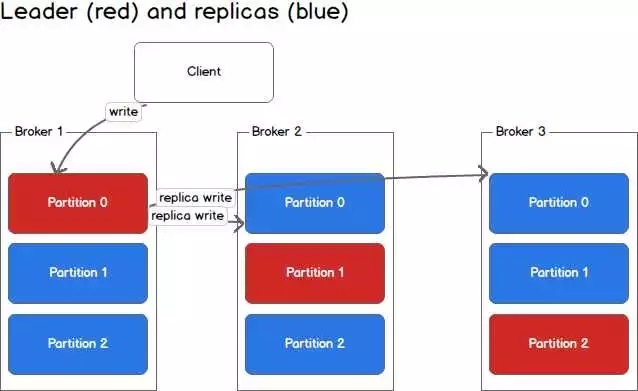
如上图所示，kafka将topic中的消息存在不同的partition中。如果存在键值（key），消息按照键值（key）做分类存在不同的partiition中，如果不存在键值（key），消息按照轮询（Round Robin）机制存在不同的partition中。默认情况下，键值（key）决定了一条消息会被存在哪个partition中。

partition中的消息序列是有序的消息序列。kafka在partition使用偏移量（offset）来指定消息的位置。一个topic的一个partition只能被一个consumer group中的一个consumer消费，多个consumer消费同一个partition中的数据是不允许的，但是一个consumer可以消费多个partition中的数据。

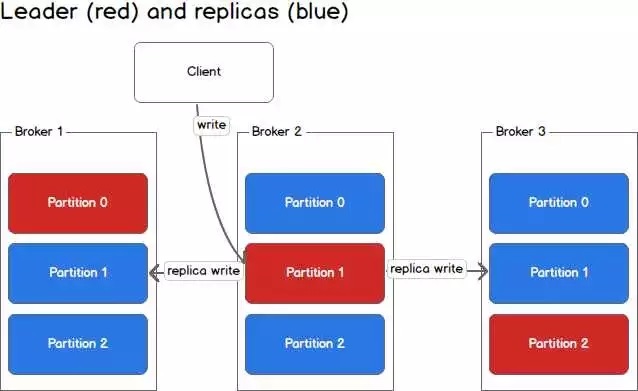
kafka将partition的数据复制到不同的broker，提供了partition数据的备份。每一个partition都有一个broker作为leader，若干个broker作为follower。所有的数据读写都通过leader所在的服务器进行，并且leader在不同broker之间复制数据。



上图中，对于Partition 0，broker 1是它的leader，broker 2和broker 3是follower。对于Partition 1，broker 2是它的leader，broker 1和broker 3是follower。



在上图中，当有Client（也就是Producer）要写入数据到Partition 0时，会写入到leader Broker 1，Broker 1再将数据复制到follower Broker 2和Broker 3。



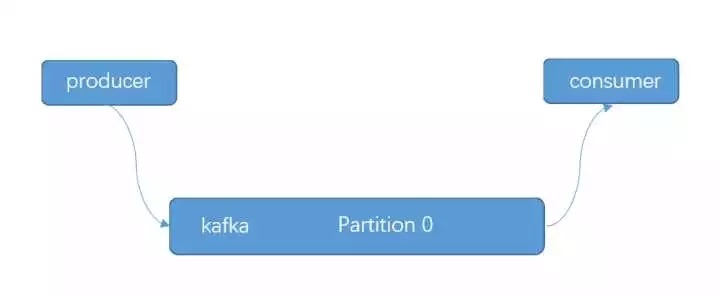
在上图中，Client向Partition 1中写入数据时，会写入到Broker 2，因为Broker 2是Partition 1的Leader，然后Broker 2再将数据复制到follower Broker 1和Broker 3中。

上图中的topic一共有3个partition，对每个partition的读写都由不同的broker处理，因此总的吞吐量得到了提升。

**实验一：kafka-python实现生产者消费者**

kafka-python是一个python的Kafka客户端，可以用来向kafka的topic发送消息、消费消息。

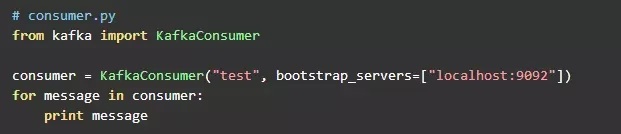
这个实验会实现一个producer和一个consumer，producer向kafka发送消息，consumer从topic中消费消息。结构如下图



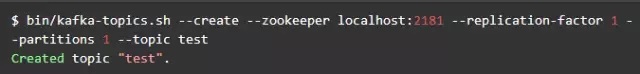
producer代码



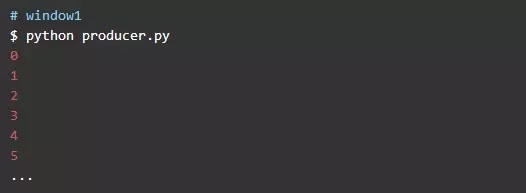
consumer代码



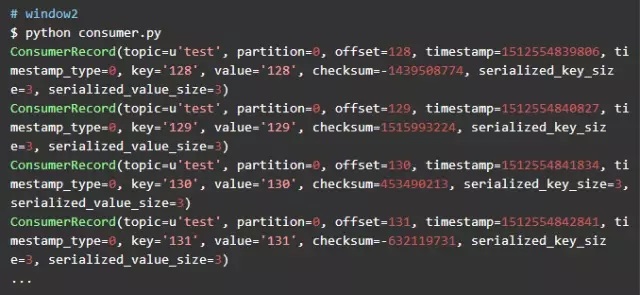
接下来创建test topic



打开两个窗口中，我们在window1中运行producer，如下



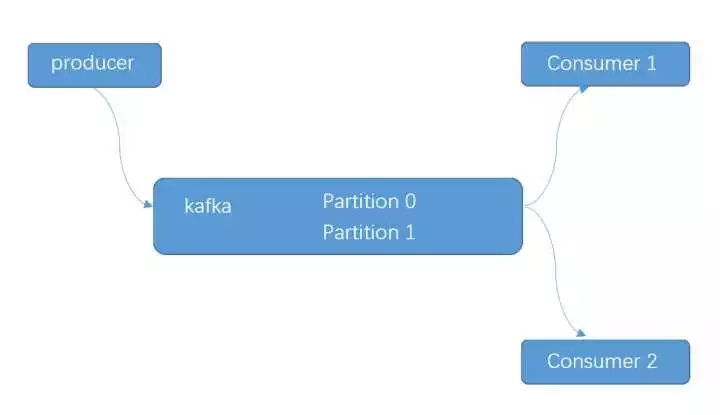
在window2中运行consumer，如下



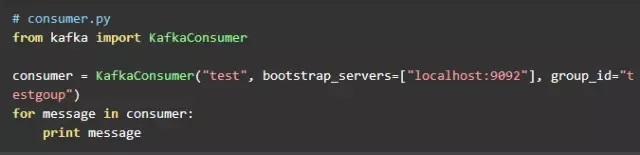
可以看到window2中的consumer成功的读到了producer写入的数据

**实验二：消费组实现容错性机制**

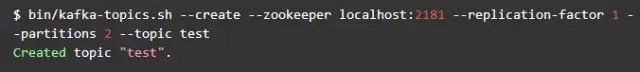
这个实验将展示消费组的容错性的特点。这个实验中将创建一个有2个partition的topic，和2个consumer，这2个consumer共同消费同一个topic中的数据。结构如下所示



producer部分代码和实验一相同，这里不再重复。consumer需要指定所属的consumer group，代码如下

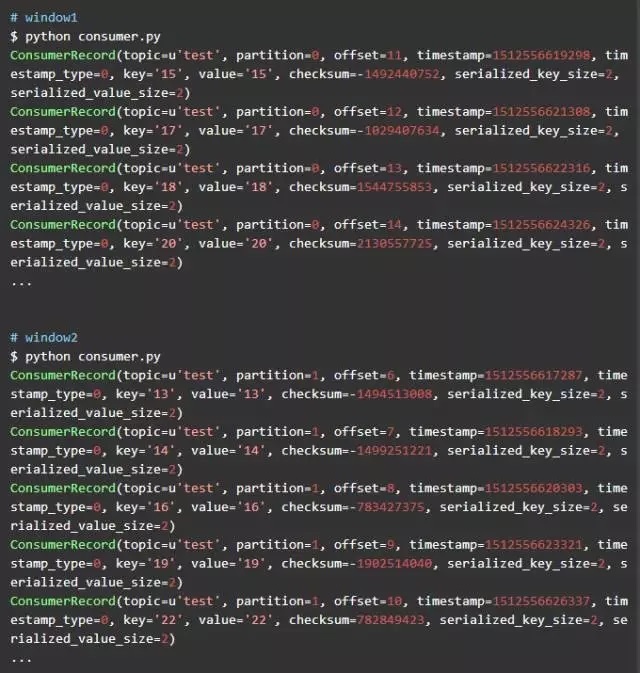


接下来我们创建topic，名字test，设置partition数量为2



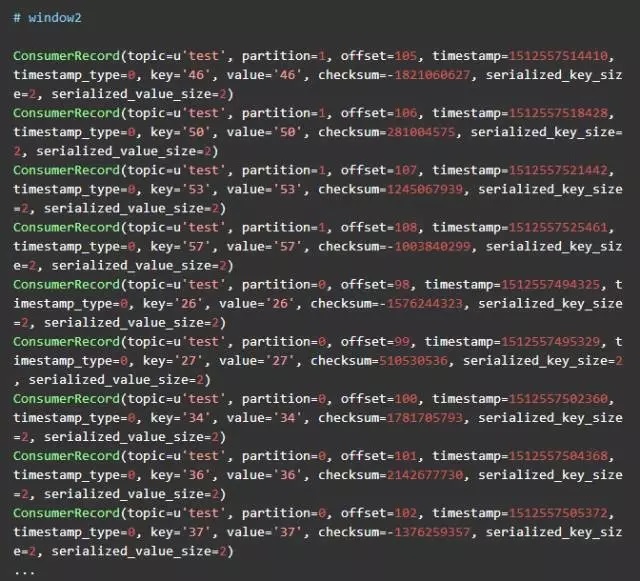
打开三个窗口，一个窗口运行producer，还有两个窗口运行consumer。

运行consumer的两个窗口的输出如下：



可以看到两个consumer同时运行的情况下，它们分别消费不同partition中的数据。window1中的consumer消费partition 0中的数据，window2中的consumer消费parition 1中的数据。

我们尝试关闭window1中的consumer，可以看到如下结果



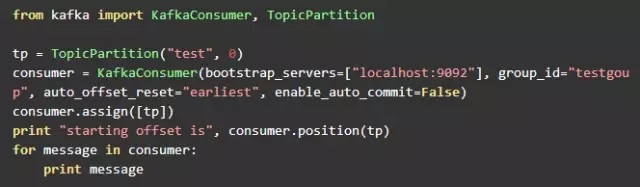
刚开始window2中的consumer只消费partition1中的数据，当window1中的consumer退出后，window2中的consumer中也开始消费partition 0中的数据了。

**实验三：offset管理**

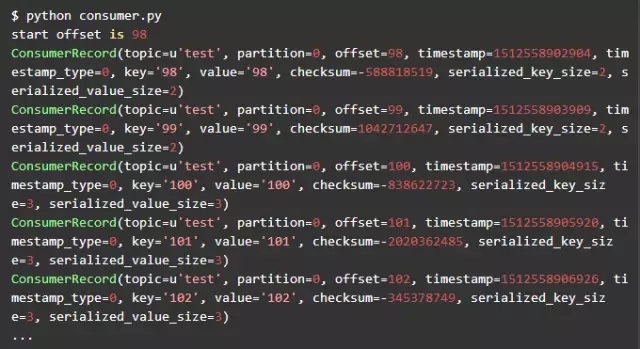
kafka允许consumer将当前消费的消息的offset提交到kafka中，这样如果consumer因异常退出后，下次启动仍然可以从上次记录的offset开始向后继续消费消息。

这个实验的结构和实验一的结构是一样的，使用一个producer，一个consumer，test topic的partition数量设为1。

producer的代码和实验一中的一样，这里不再重复。consumer的代码稍作修改，这里consumer中打印出下一个要被消费的消息的offset。consumer代码如下

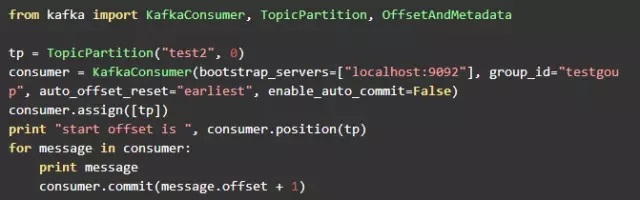


在一个窗口中启动producer，在另一个窗口并且启动consumer。consumer的输出如下

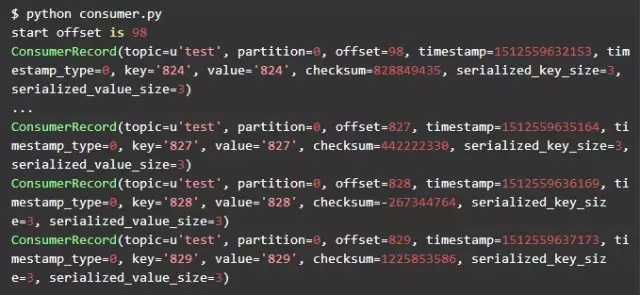


可以尝试退出consumer，再启动consumer。每一次重新启动，consumer都是从offset=98的消息开始消费的。

修改consumer的代码如下，在consumer消费每一条消息后将offset提交回kafka

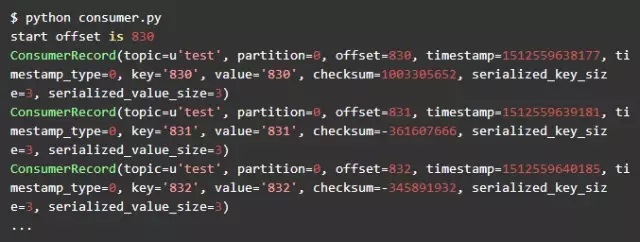


启动consumer



可以看到consumer从offset=98的消息开始消费，到offset=829时，我们Ctrl+C退出consumer。

我们再次启动consumer



可以看到重新启动后，consumer从上一次记录的offset开始继续消费消息。之后每一次consumer重新启动，consumer都会从上一次停止的地方继续开始消费。