### **More Partitions Lead to Higher Throughput**

The first thing to understand is that a topic partition is the unit of parallelism in Kafka. On both the producer and the broker side, writes to different partitions can be done fully in parallel. So expensive operations such as compression can utilize more hardware resources. On the consumer side, Kafka always gives a single partition’s data to one consumer thread. Thus, the degree of parallelism in the consumer (within a consumer group) is bounded by the number of partitions being consumed. Therefore, in general, the more partitions there are in a Kafka cluster, the higher the throughput one can achieve.

A rough formula for picking the number of partitions is based on throughput. You measure the throughout that you can achieve on a single partition for production (call it p) and consumption (call it c). Let’s say your target throughput is t. Then you need to have at least max(t/p, t/c) partitions. The per-partition throughput that one can achieve on the producer depends on configurations such as the batching size, compression codec, type of acknowledgement, replication factor, etc. However, in general, one can produce at 10s of MB/sec on just a single partition as shown in this [benchmark](https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines). The consumer throughput is often application dependent since it corresponds to how fast the consumer logic can process each message. So, you really need to measure it.

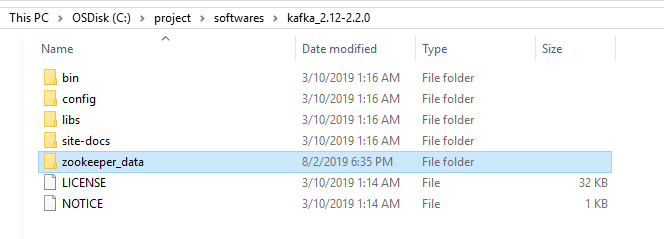
Although it’s possible to increase the number of partitions over time, one has to be careful if messages are produced with keys. When publishing a keyed message, Kafka deterministically maps the message to a partition based on the hash of the key. This provides a guarantee that messages with the same key are always routed to the same partition. This guarantee can be important for certain applications since messages within a partition are always delivered in order to the consumer. If the number of partitions changes, such a guarantee may no longer hold. To avoid this situation, a common practice is to over-partition a bit. Basically, you determine the number of partitions based on a future target throughput, say for one or two years later. Initially, you can just have a small Kafka cluster based on your current throughput. Over time, you can add more brokers to the cluster and proportionally move a subset of the existing partitions to the new brokers (which can be done online). This way, you can keep up with the throughput growth without breaking the semantics in the application when keys are used.

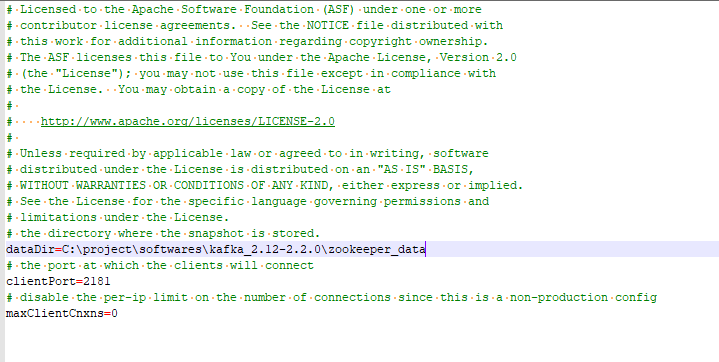
In addition to throughput, there are a few other factors that are worth considering when choosing the number of partitions. As you will see, in some cases, having too many partitions may also have negative impact.

**Installing Kafka and Zookeeper**

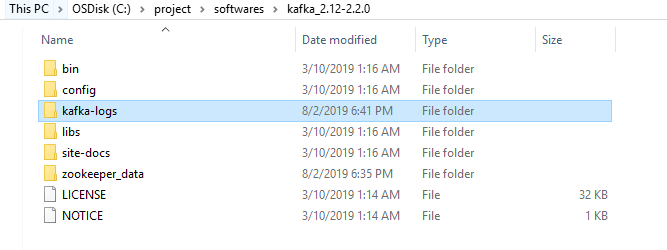
<https://kafka.apache.org/quickstart>

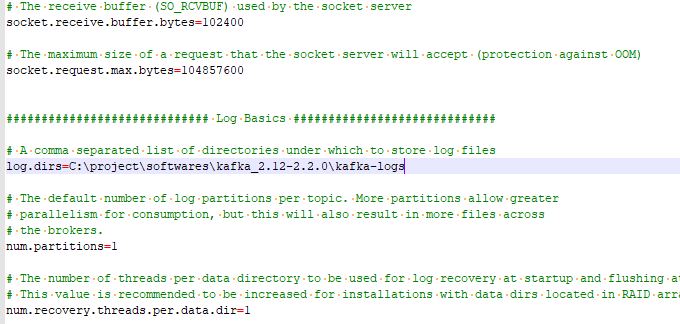
Point the data Dir to a valid Dir in windows



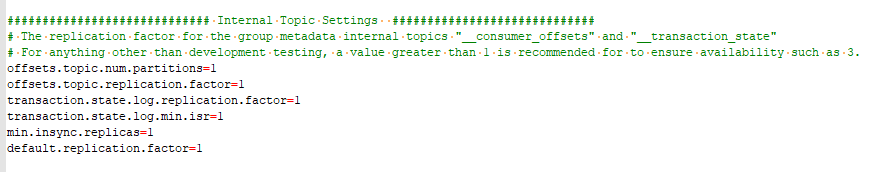


Create directory for Kafka logs





Change the log.dirs to valid directory in server.properties file



**Start Zookeeper**

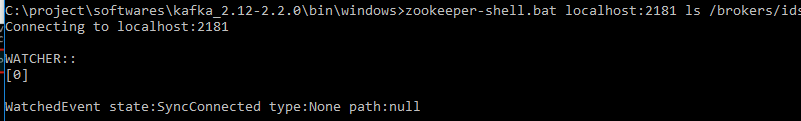
C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>zookeeper-server-start.bat C:\project\softwares\kafka\_2.12-2.2.0\config\zookeeper.properties

**Start Kafka**

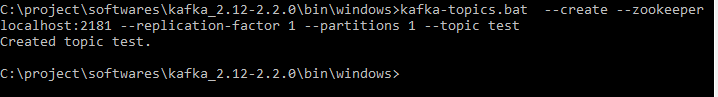
C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-server-start.bat C:\project\softwares\kafka\_2.12-2.2.0\config\server.properties

**To Test**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>zookeeper-shell.bat localhost:2181 ls /brokers/ids



C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

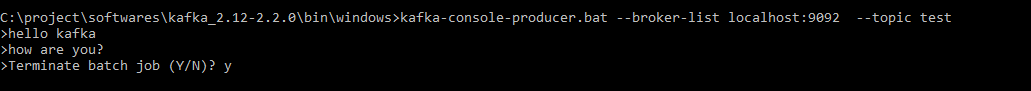


**Sending message to the topic test**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic test

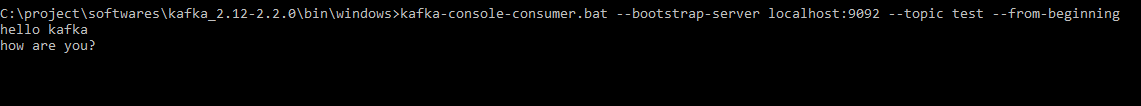
>hello kafka

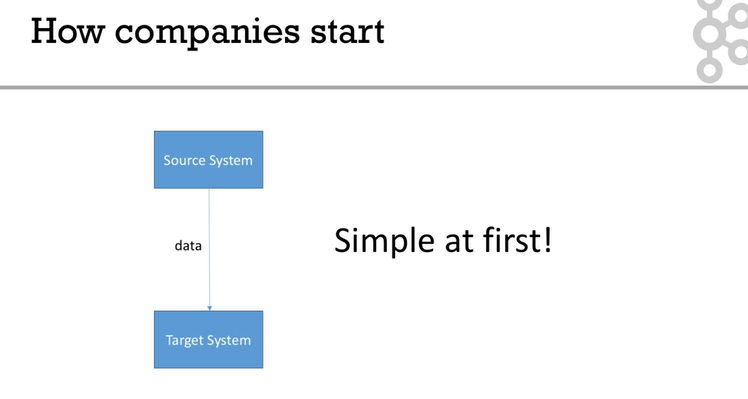
>how are you?

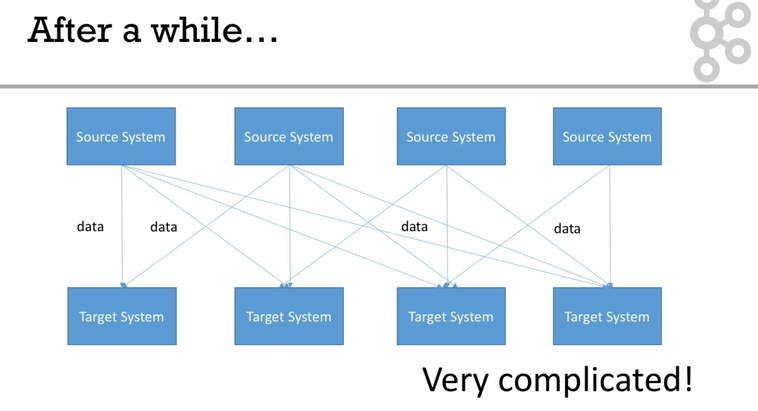


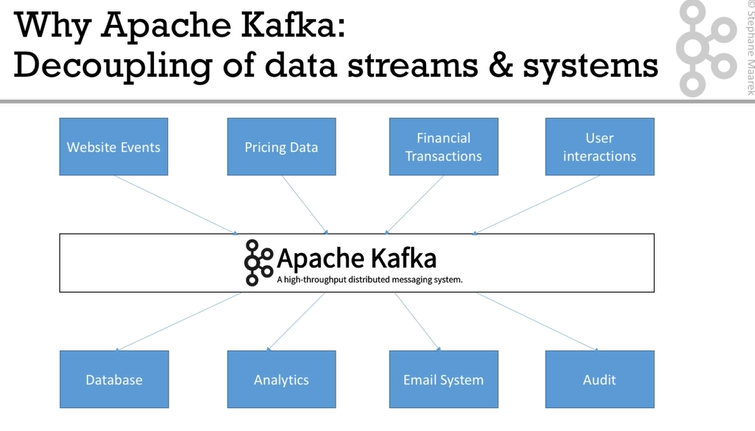
**Receiving message from topic test**

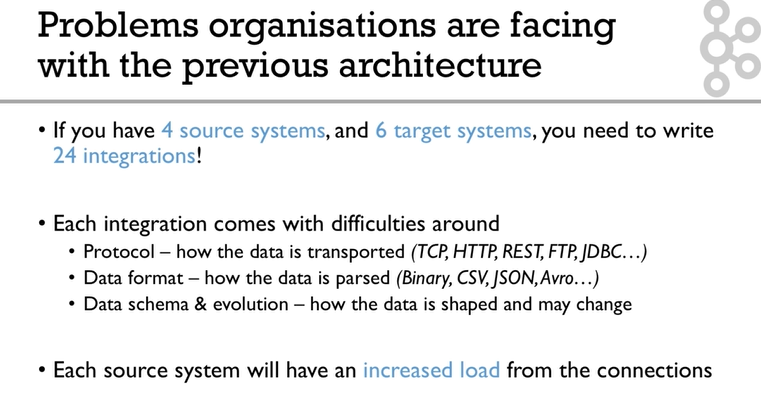
C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test --from-beginning

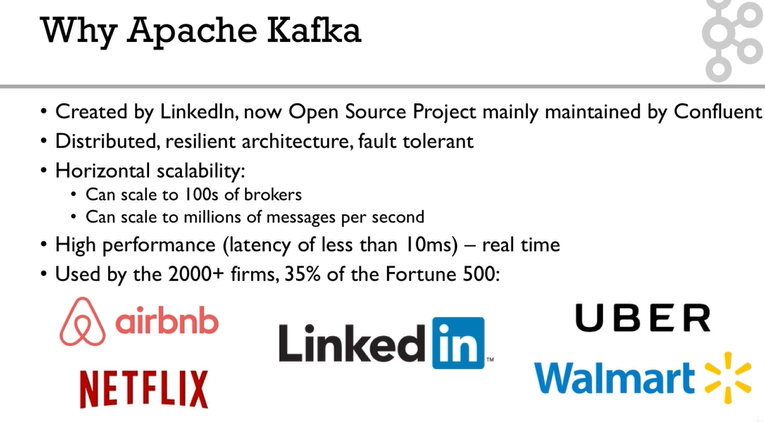


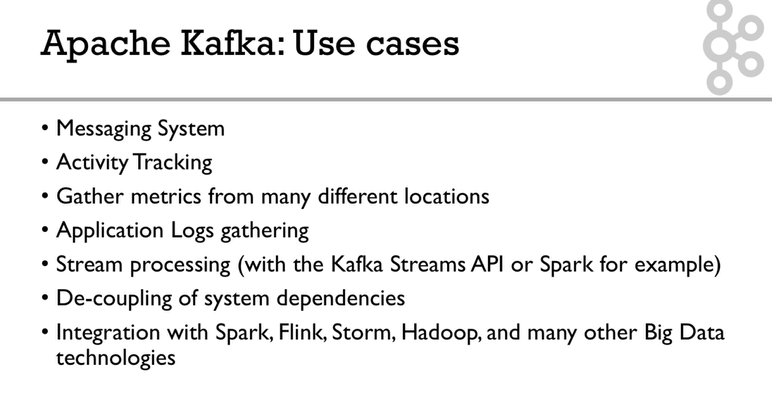


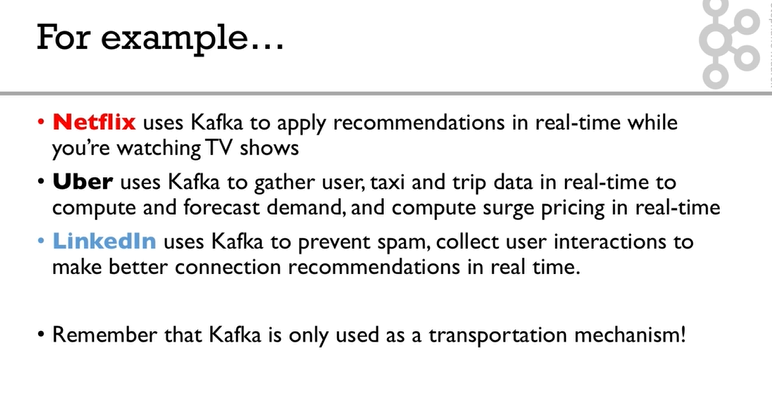


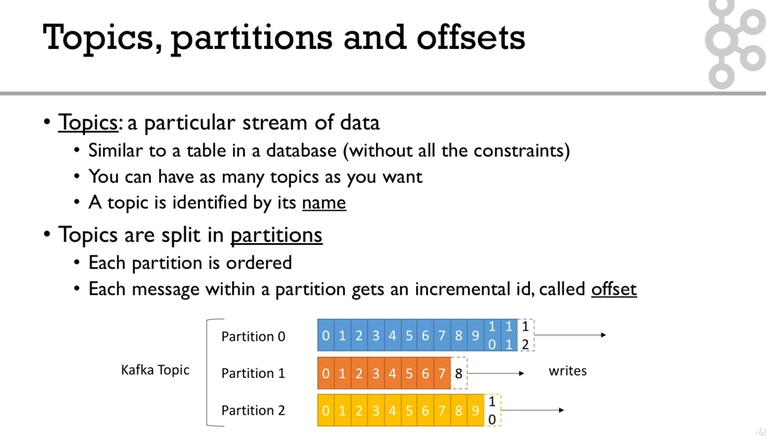


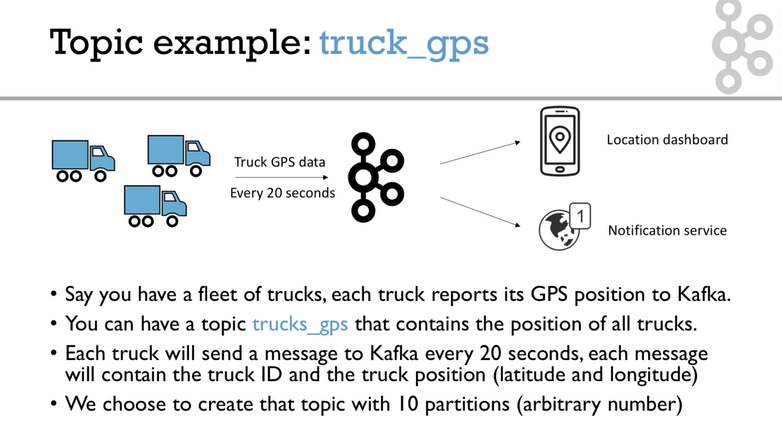


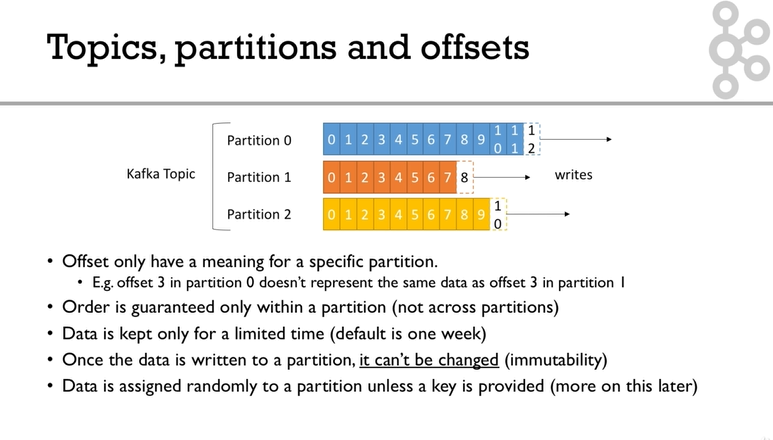


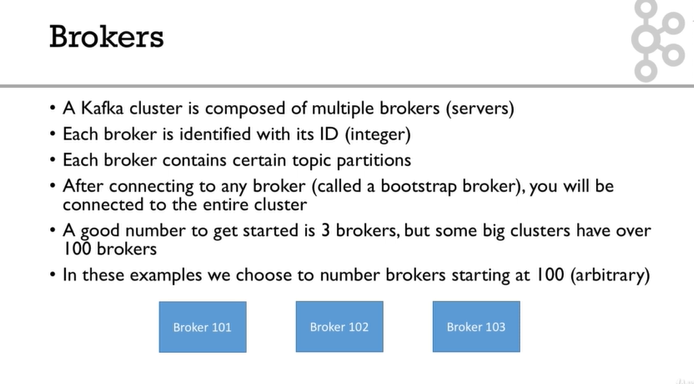


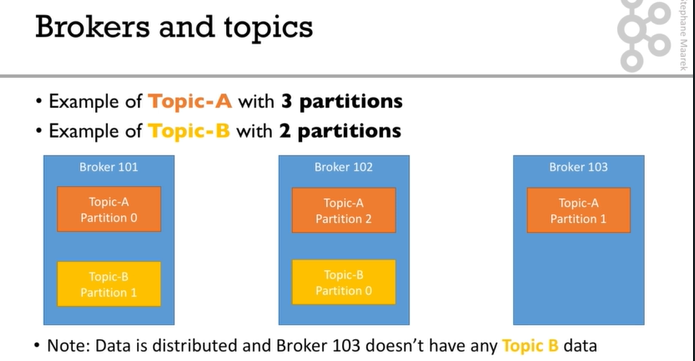


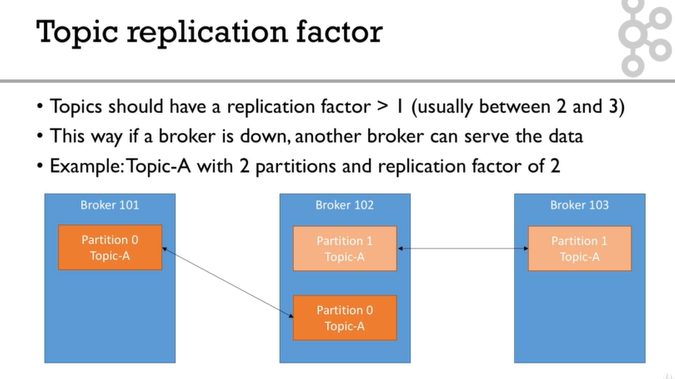


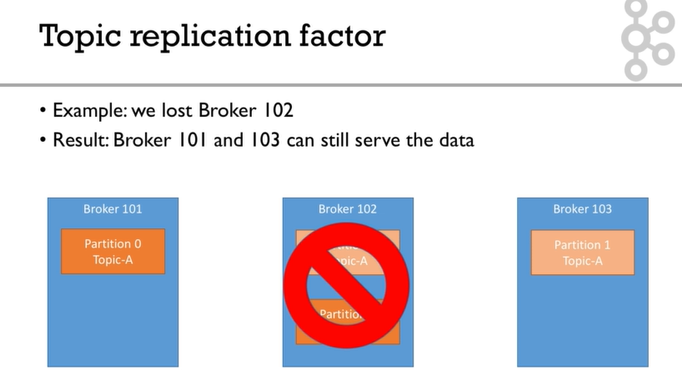


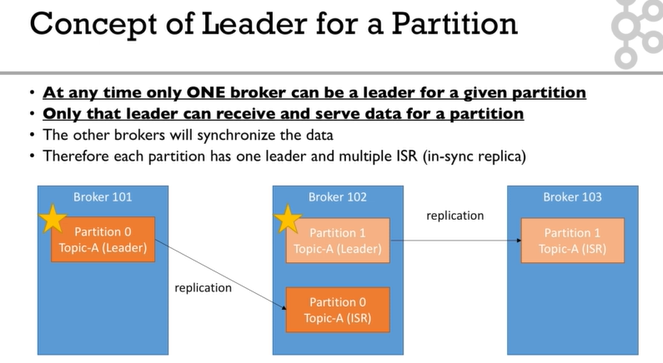




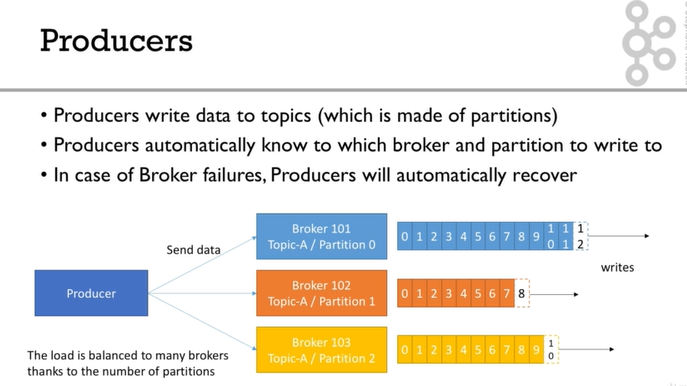


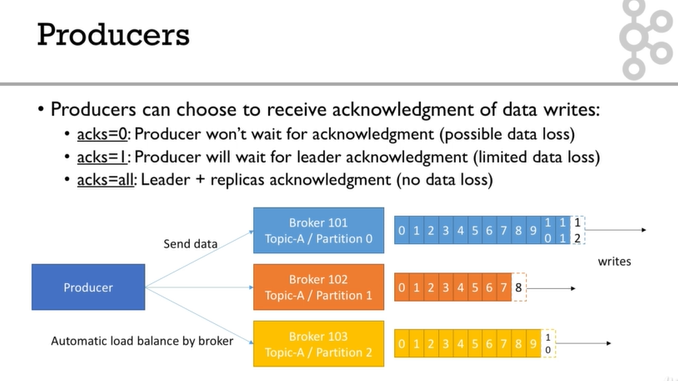


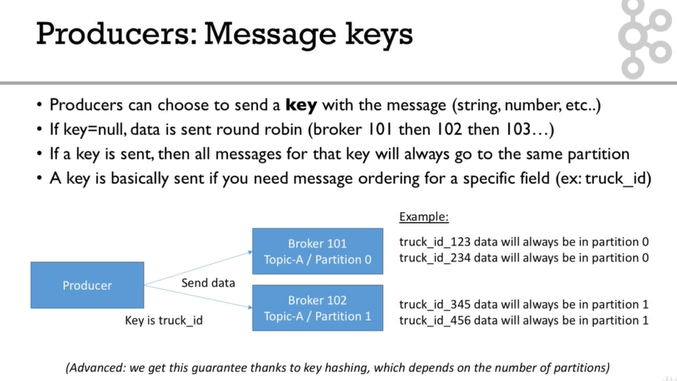


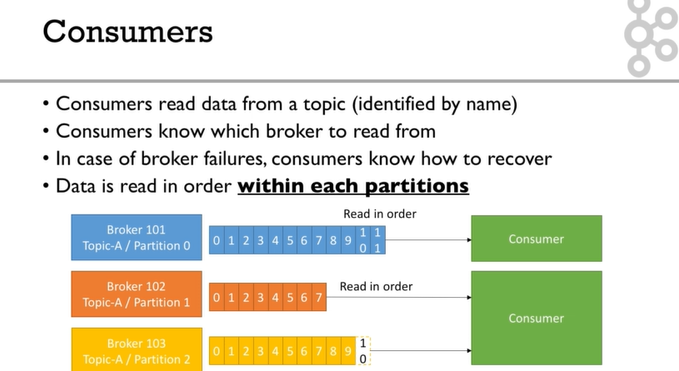


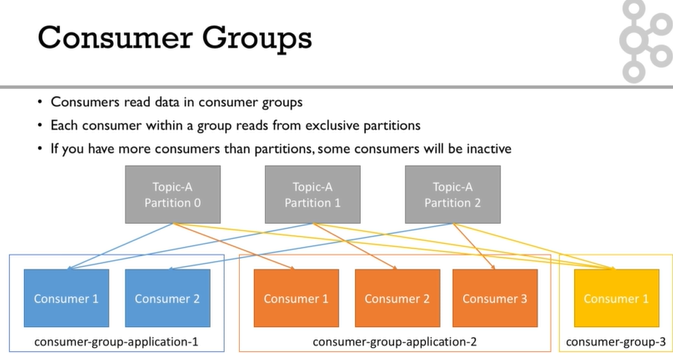
The leader and ISR are decided by zookeeper.

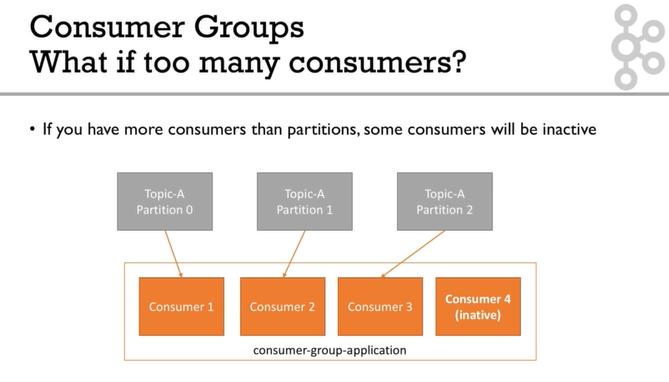


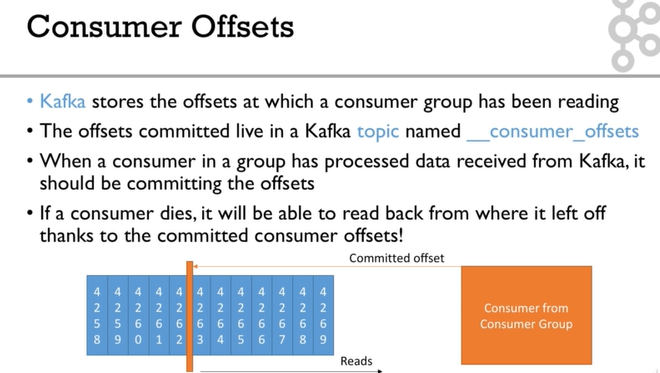


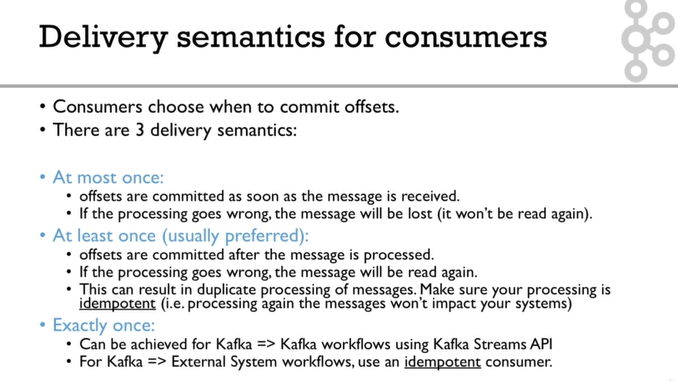


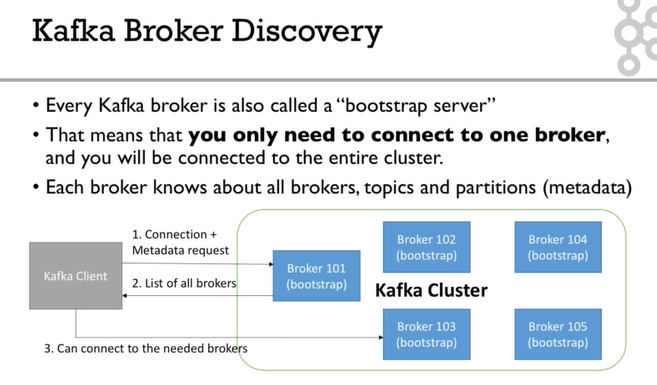


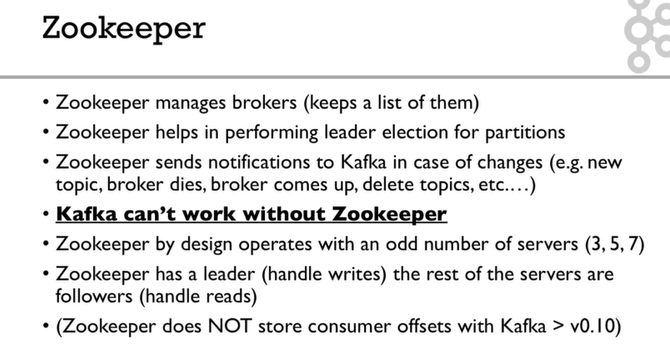


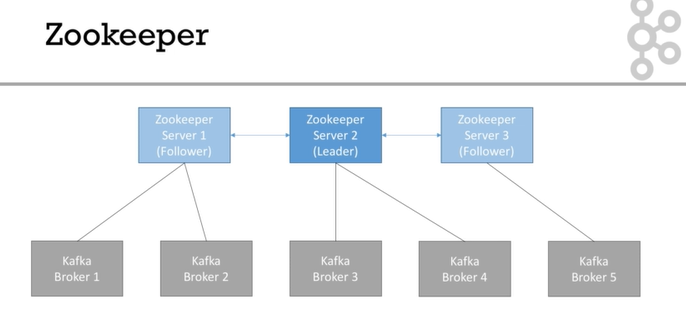


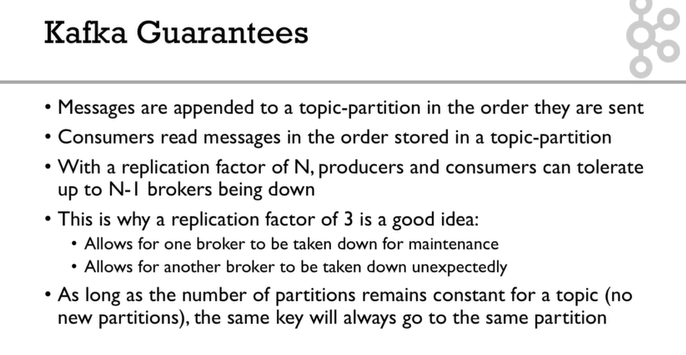


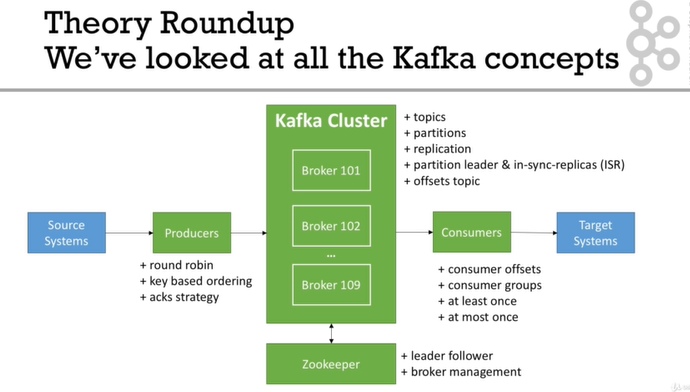












**In summary, for Linux (ex: Ubuntu)**

1. Download and Setup **Java 8 JDK:**
   1. sudo apt install openjdk-8-jdk
2. Download & Extract the Kafka binaries from <https://kafka.apache.org/downloads>
3. Try Kafka commands using bin/kafka-topics.sh (for example)
4. Edit PATH to include Kafka (in ~/.bashrc for example) PATH="$PATH:/your/path/to/your/kafka/bin"
5. Edit Zookeeper & Kafka configs using a text editor
   1. zookeeper.properties: dataDir=/your/path/to/data/zookeeper
   2. server.properties: log.dirs=/your/path/to/data/kafka
6. Start Zookeeper in one terminal window: zookeeper-server-start.sh config/zookeeper.properties
7. Start Kafka in **another** terminal window: kafka-server-start.sh config/server.properties

**Important: For the rest of the course, don't forget to add the extension .sh to commands being run**

Windows - Summary

**In summary, for Windows**

1. Download and Setup **Java 8 JDK**
2. Download the Kafka binaries from <https://kafka.apache.org/downloads>
3. Extract Kafka at the root of C:\
4. Setup Kafka bins in the **Environment variables** section by editing **Path**
5. Try Kafka commands using kafka-topics.bat (for example)
6. Edit Zookeeper & Kafka configs using NotePad++ <https://notepad-plus-plus.org/download/>
   1. zookeeper.properties: dataDir=C:/kafka\_2.12-2.0.0/data/zookeeper (yes the slashes are inversed)
   2. server.properties: log.dirs=C:/kafka\_2.12-2.0.0/data/kafka (yes the slashes are inversed)
7. Start Zookeeper in one command line: zookeeper-server-start.bat config\zookeeper.properties
8. Start Kafka in **another** command line: kafka-server-start.bat config\server.properties

**Important: For the rest of the course, don't forget to add the extension .bat to commands being run**

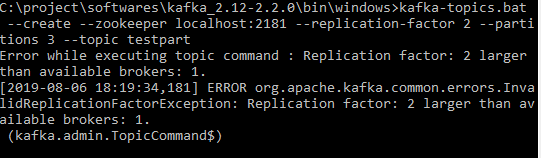
**Find all available topics**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-topics.bat --zookeeper localhost:2181 --list

\_\_consumer\_offsets

test

testjson



**Describing particular topic**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-topics.bat --zookeeper localhost:2181 --topic testpart --describe

Topic:testpart PartitionCount:3 ReplicationFactor:1 Configs:

Topic: testpart Partition: 0 Leader: 0 Replicas: 0 Isr: 0

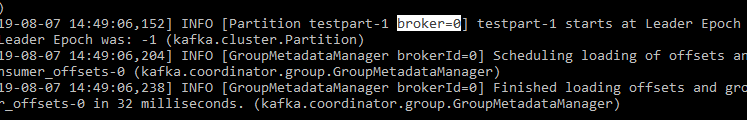
Topic: testpart Partition: 1 Leader: 0 Replicas: 0 Isr: 0

Topic: testpart Partition: 2 Leader: 0 Replicas: 0 Isr: 0

In the above example partition size is 3 replication factor is 1.

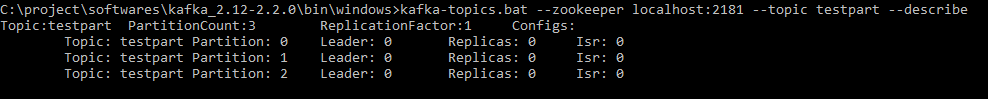
There are 3 partitions as Partition 0, Partition 1, and Partition 2

Leader is Broker 0.



For each partition number of replicas mentioned 0 means the replica is present in broker id 0.

Isr:0 means the in sync replica present in broker 0.



**Commands to start Zookeeper and Kafka**

zookeeper-server-start.bat C:\project\softwares\kafka\_2.12-2.2.0\config\zookeeper.properties

kafka-server-start.bat C:\project\softwares\kafka\_2.12-2.2.0\config\server.properties

**To see the help of Kafka topic**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-topics.bat

**Deleting a topic**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-topics.bat --zookeeper localhost:2181 --delete --topic test

Topic test is marked for deletion.

Note: This will have no impact if delete.topic.enable is not set to true.

**Reading Only New Data (Not from Beginning)**

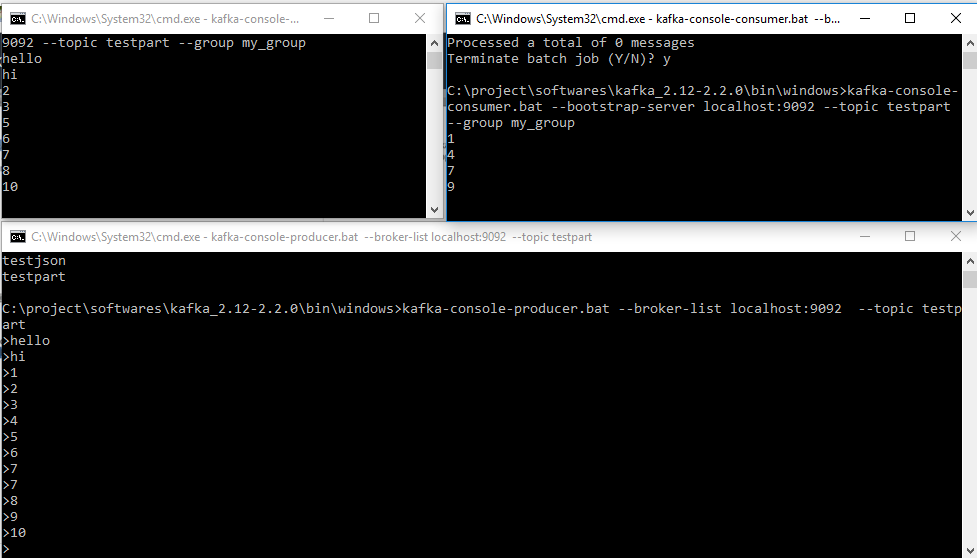
C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test

hiiiii

--bootstrap-server and --broker-list are same it is kafka

**Consumer Group**

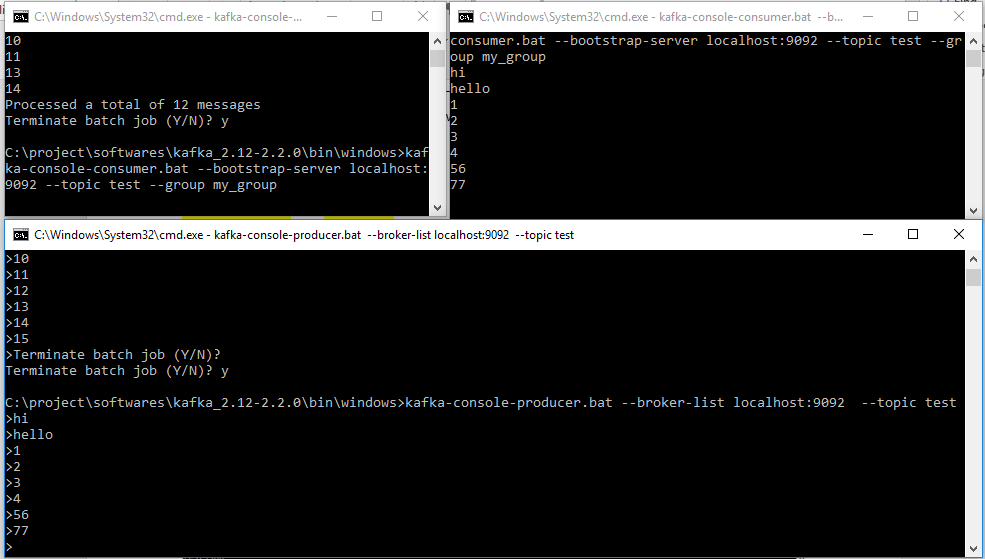
All the consumers present in same consumer group will receive the messages from producer based on some algorithm.



If number of partition is 1 then only one consumer receiving the message.

If number of partitions 3 and we have three consumers then they will read data from different partitions.

If we have 3 partitions and 2 consumers then data from 3 partitions will be read by 2 consumers (available consumers)



In the consumer if we mention --group and – from-beginning only once it will get all the messages, 2nd time for the same group any other consumer it will not read older messages because the offset has been committed.

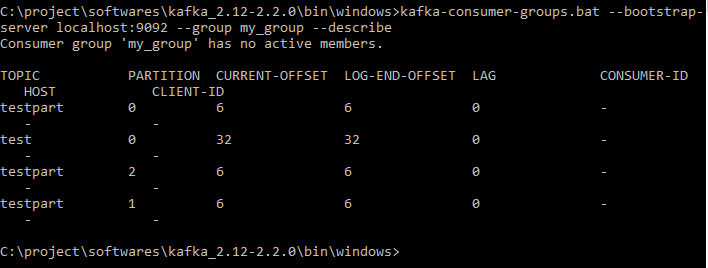
**Kafka Consumer Group**

C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-consumer-groups.bat --bootstrap-server localhost:9092 --list

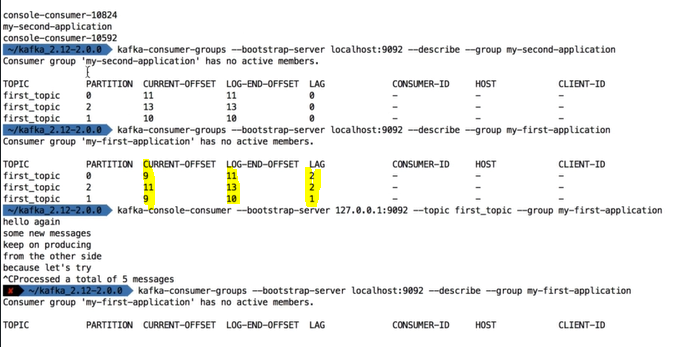
my\_group

**Describing a consumer Group**

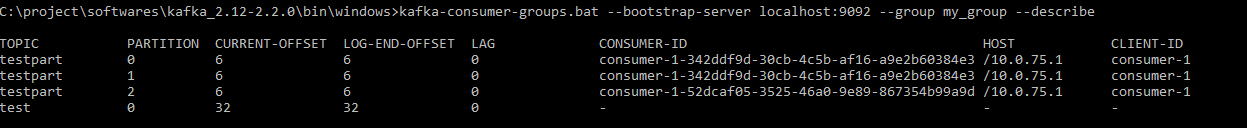
C:\project\softwares\kafka\_2.12-2.2.0\bin\windows>kafka-consumer-groups.bat --bootstrap-server localhost:9092 --group my\_group –describe



As the lag field is zero there are no pending messages present in topic testpart for consumer group my\_group



Below image shows which consumer is connected to which partition. In this case one of the consumer accessing 1 partition and another one accessing 2 partitions.



It is possible to reset or shift the current offset the offset

Producer with keys

1. kafka-console-producer --broker-list 127.0.0.1:9092 --topic first\_topic --property parse.key=true --property key.separator=,
2. > key,value
3. > another key,another value

Consumer with keys

1. kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first\_topic --from-beginning --property print.key=true --property key.separator=,

**What about UIs?**

Kafka does not come bundled with a UI, but here are some recommendations

Kafka Manager (for managing Kafka and instead of using CLI): https://github.com/yahoo/kafka-manager

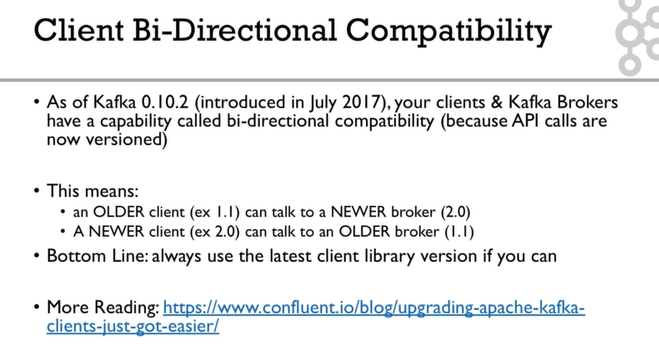
Kafka Tools (will be explore in the next lecture)

You may find other UIs on the internet

Overall in this course, we perform all the actions without any UI, which is good for learning how to properly use the CLI

//kafkaJsonTemplate.send(topic, key, data);

//providing a key ensures that a particular key always go to 1 specific partition



**Real World Exercise**

Before jumping to the next section for the solution, here are some pointers for some exercises:

**Twitter Producer**

The Twitter Producer gets data from Twitter based on some keywords and put them in a Kafka topic of your choice

Twitter Java Client: https://github.com/twitter/hbc

Twitter API Credentials: https://developer.twitter.com/

**ElasticSearch Consumer**

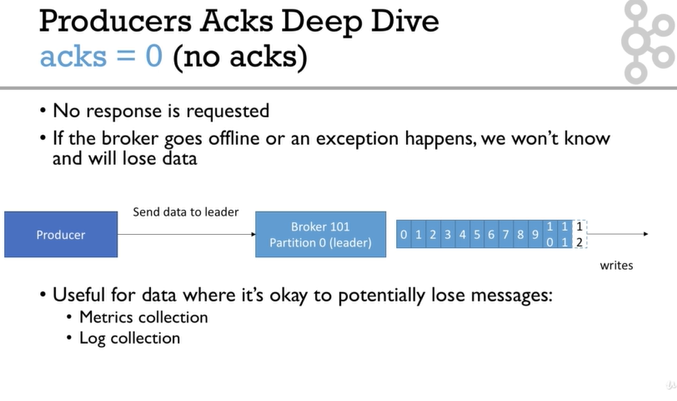
The ElasticSearch Consumer gets data from your twitter topic and inserts it into ElasticSearch

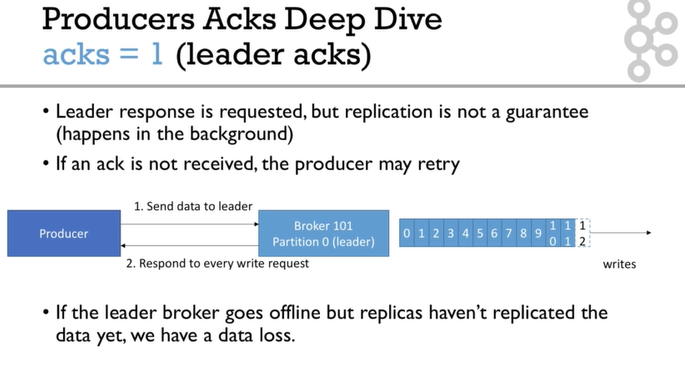
ElasticSearch Java Client: https://www.elastic.co/guide/en/elasticsearch/client/java-rest/6.4/java-rest-high.html

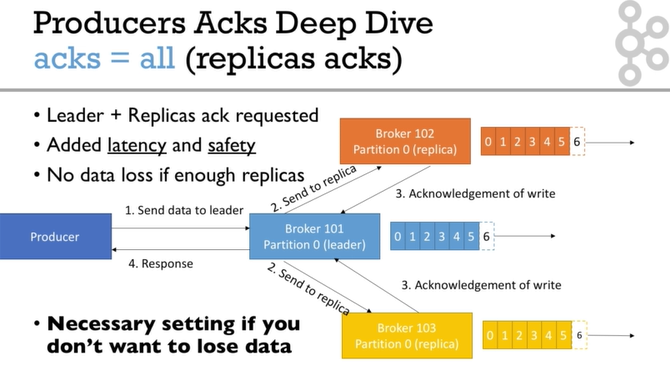
ElasticSearch setup:

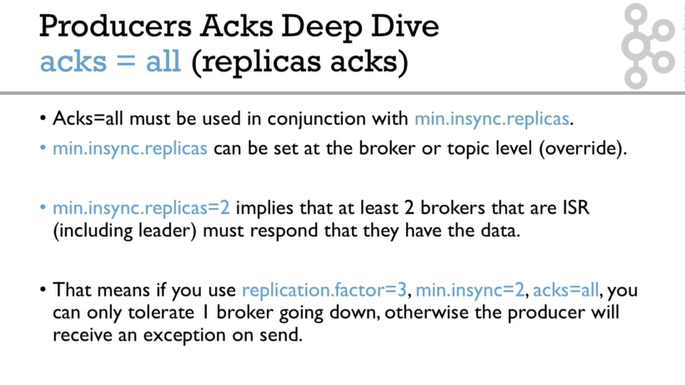
https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html

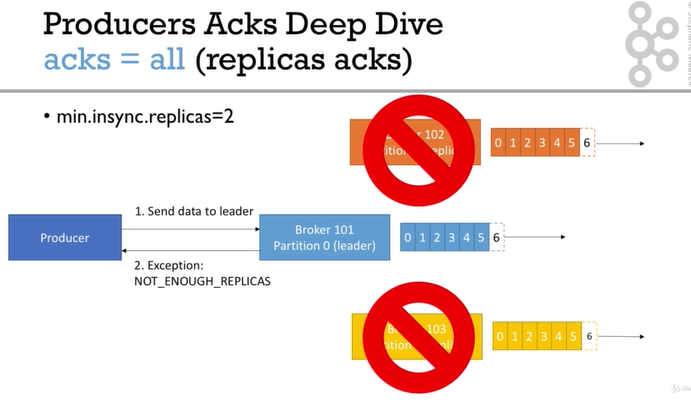
OR <https://bonsai.io/>

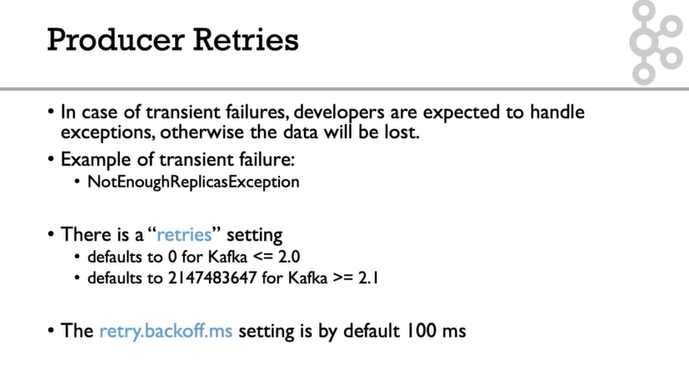


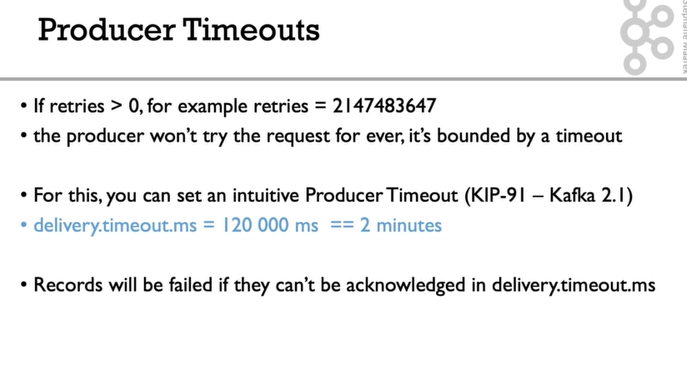


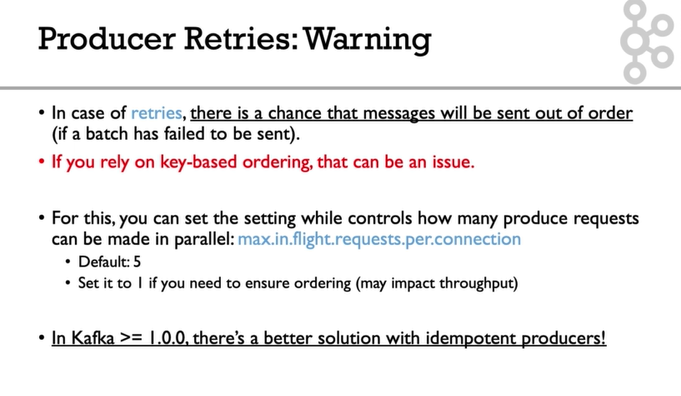


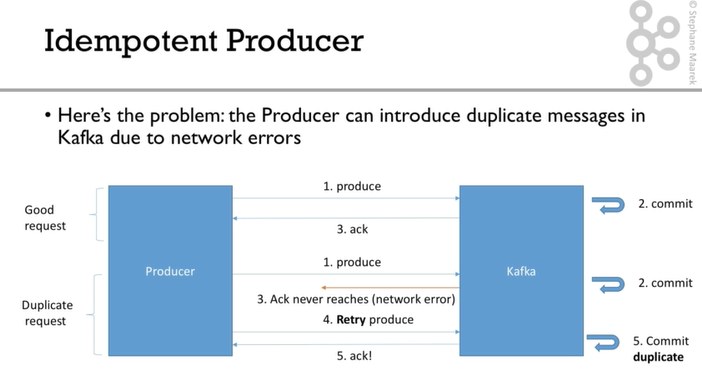


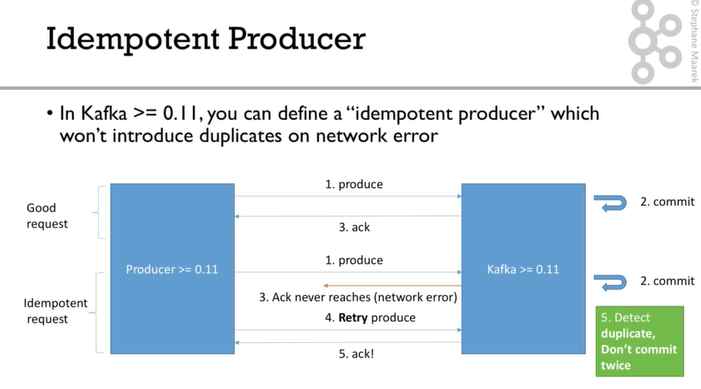


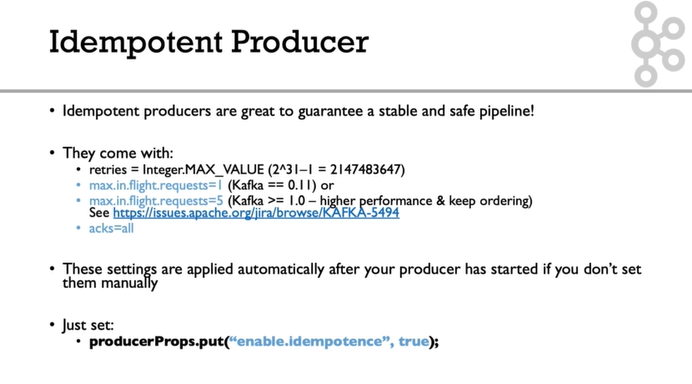






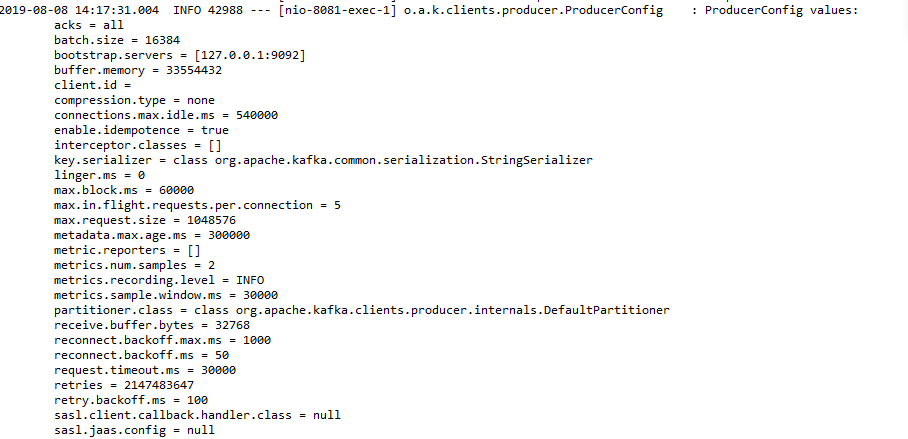


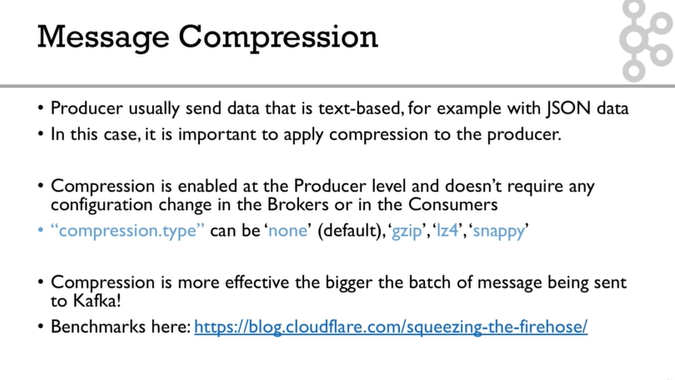


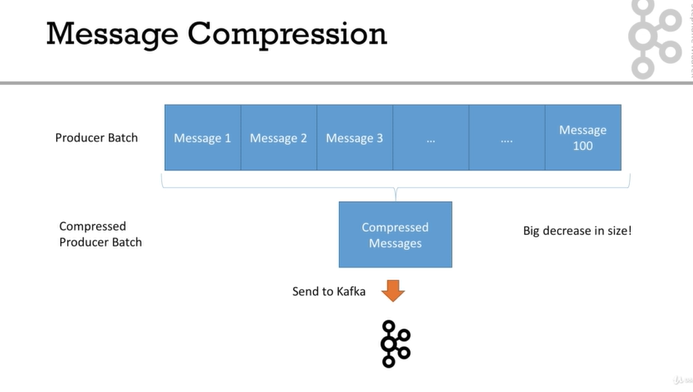


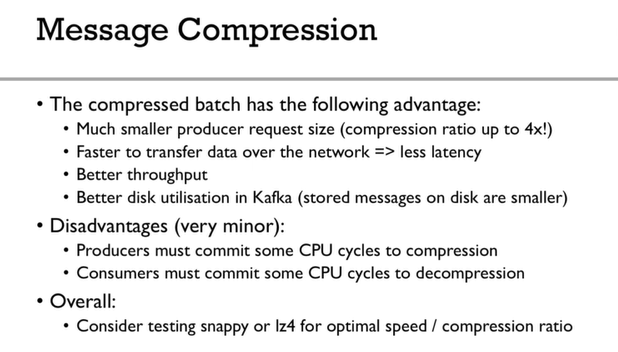


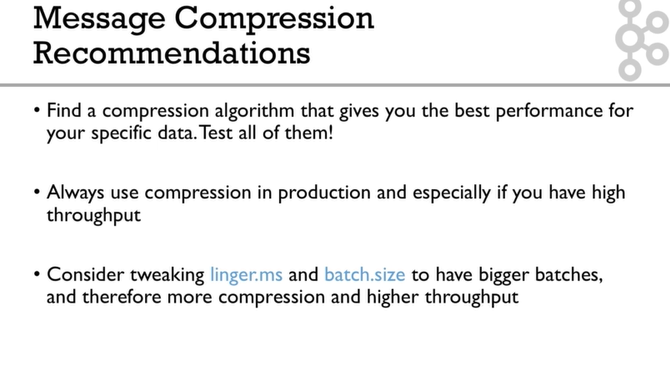
**Important configurations**

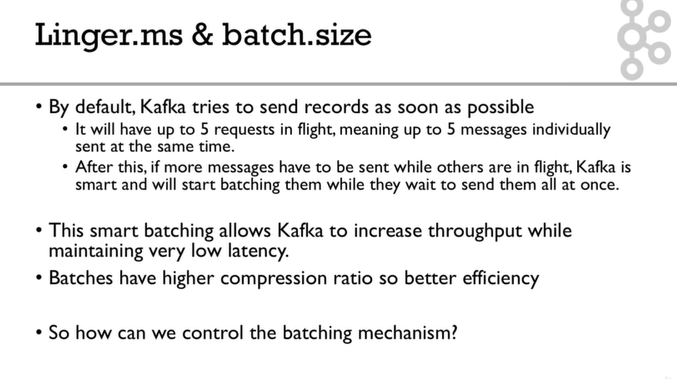


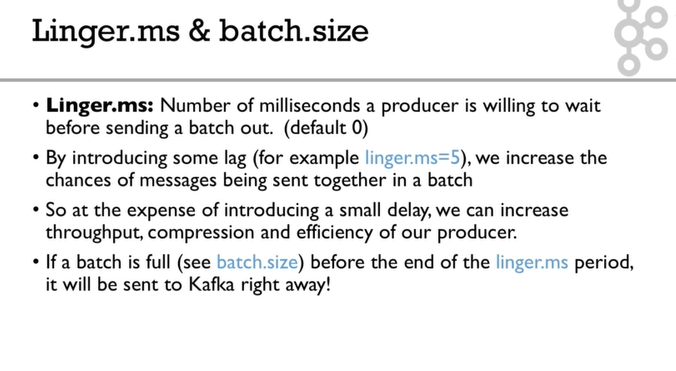


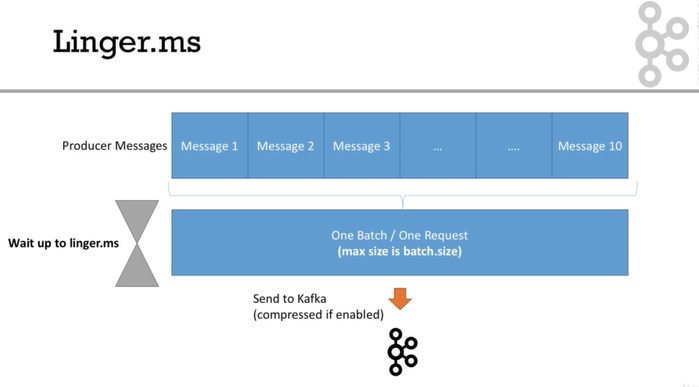


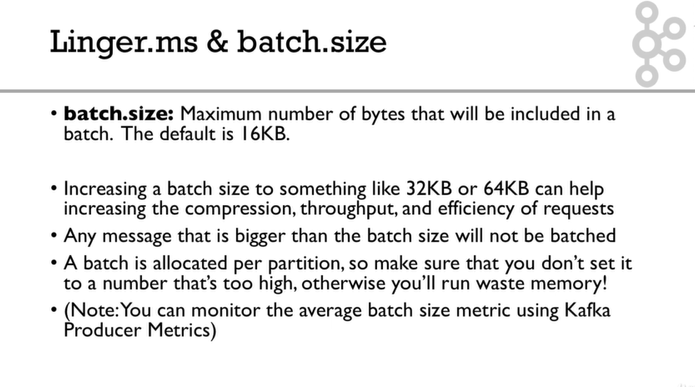


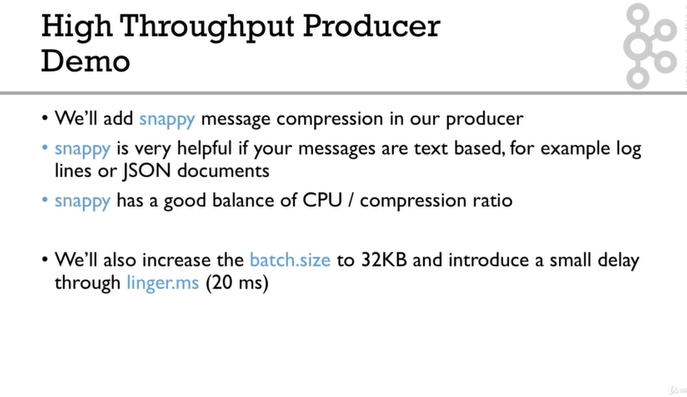


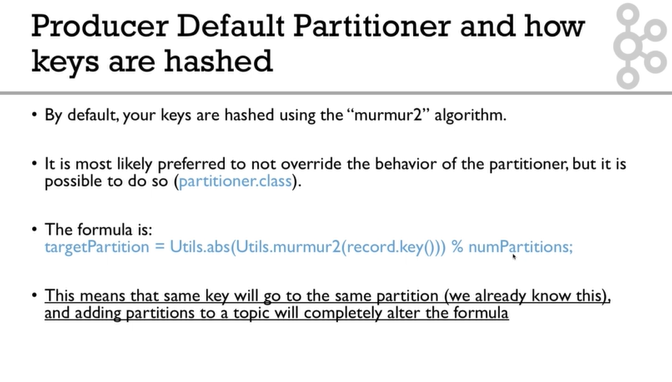


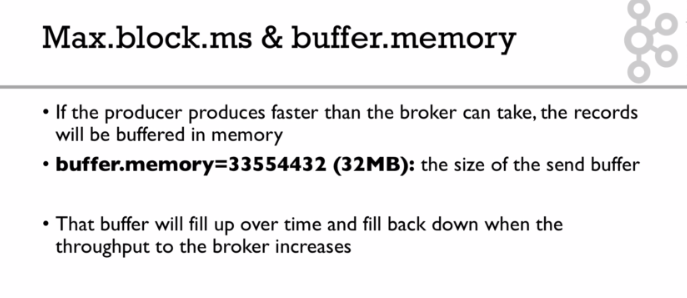


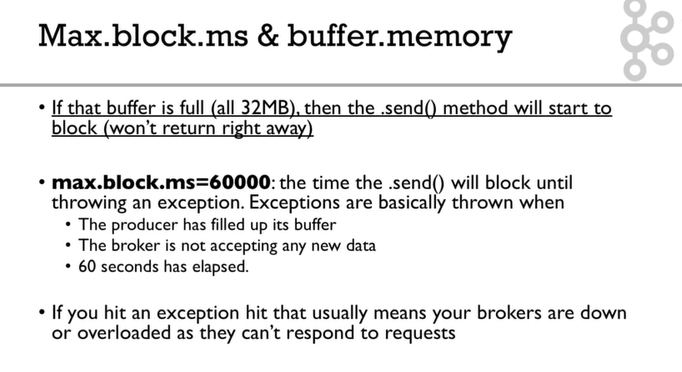


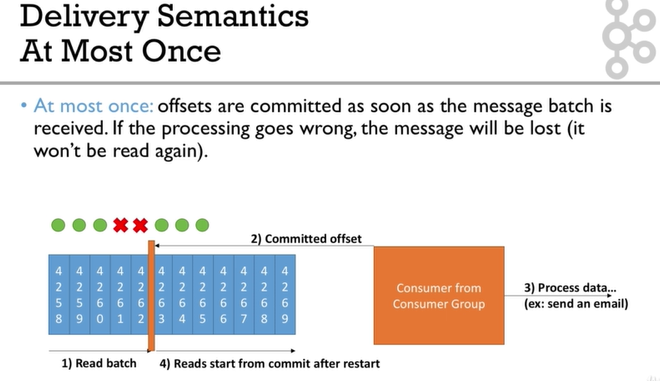


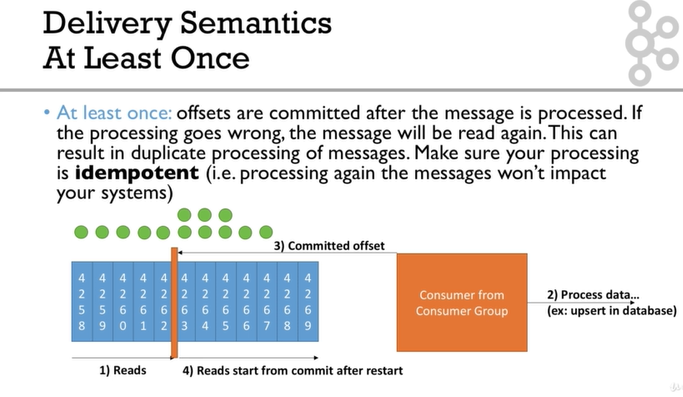




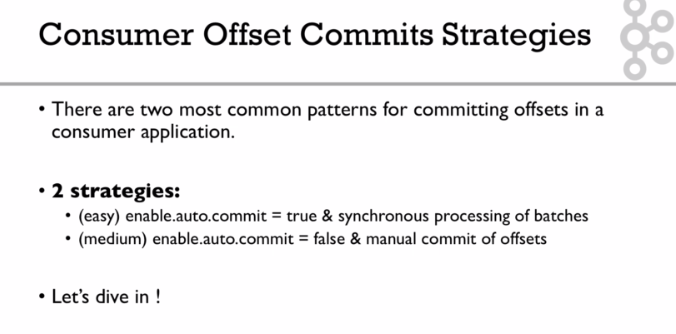


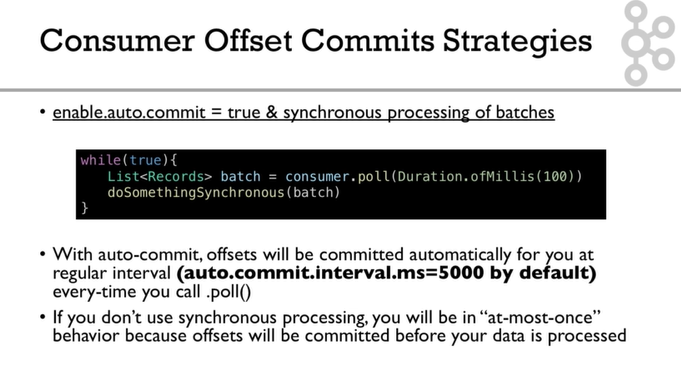


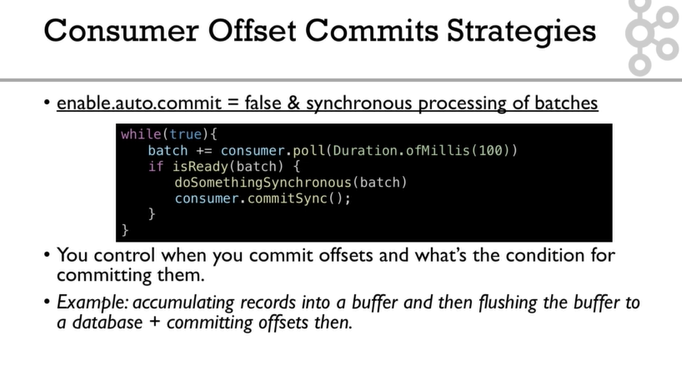


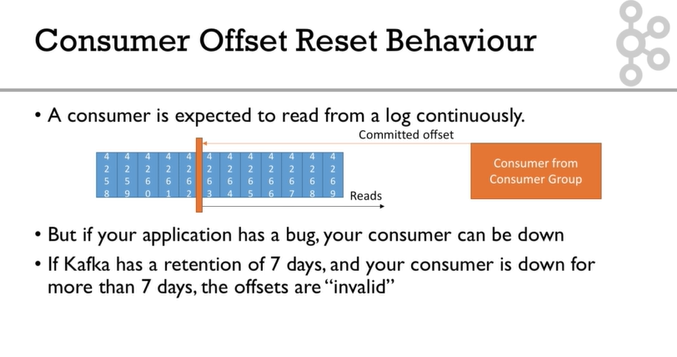


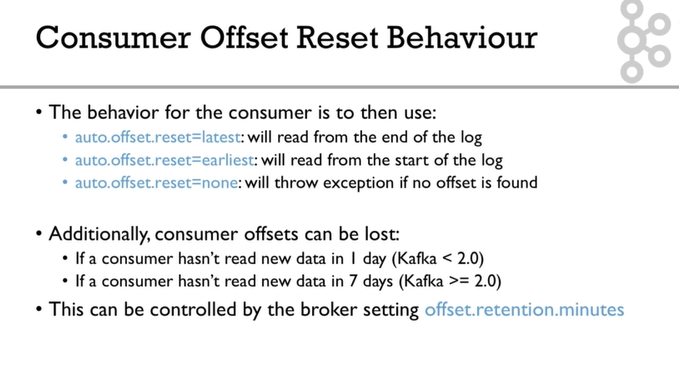


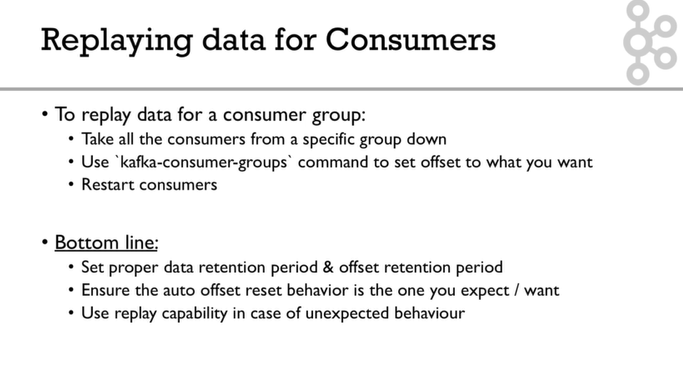


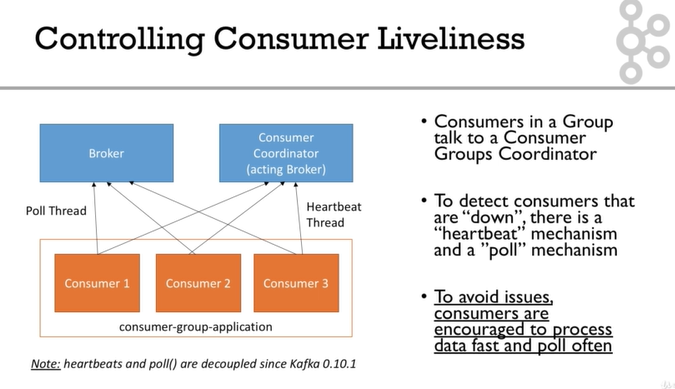




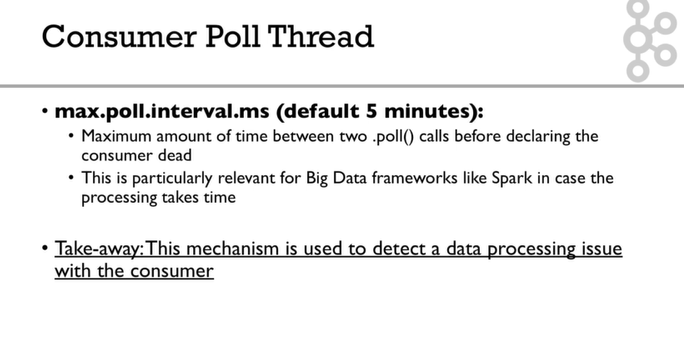


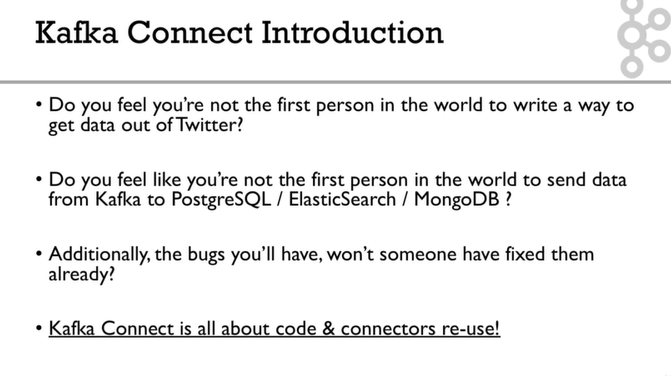


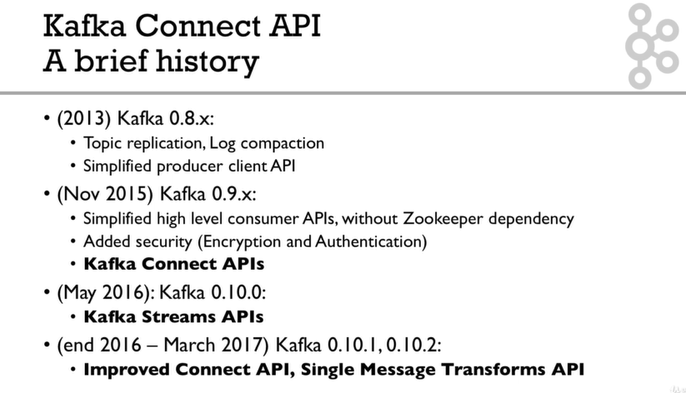


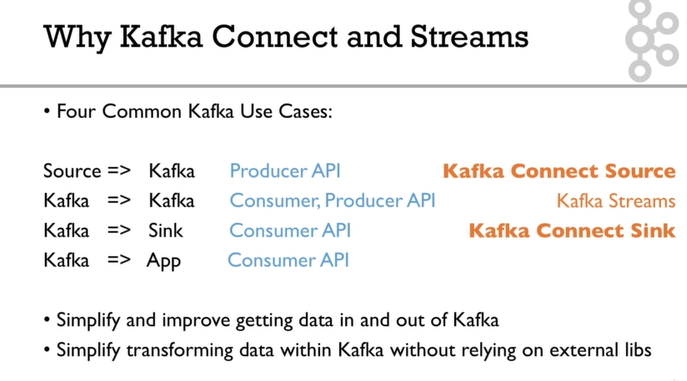


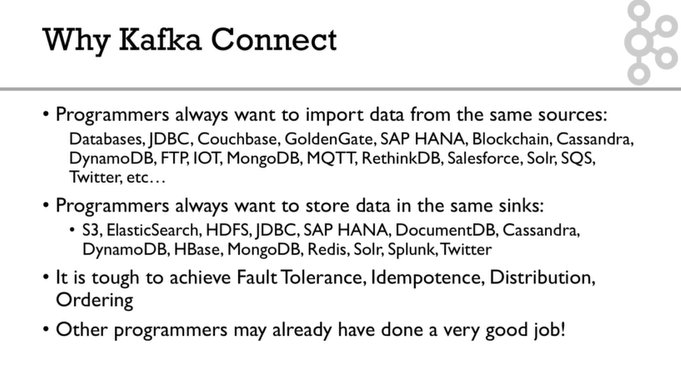


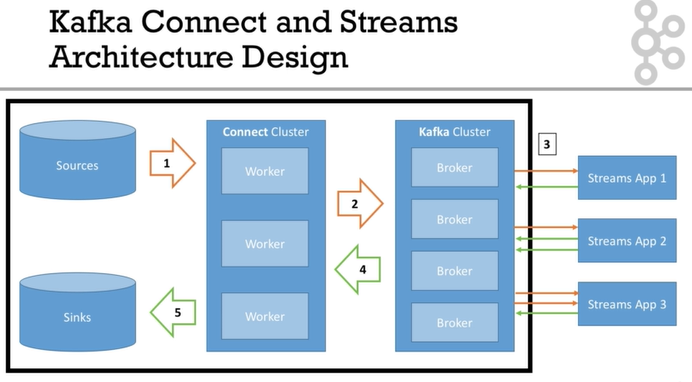


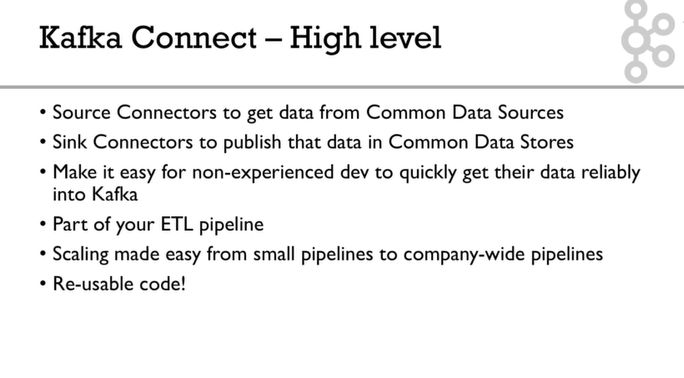












Note on Docker Setup

#### ****Note on Docker Setup****

Docker is notoriously hard to setup on some machines, and can have obscure bugs. If you don't know Docker at all, I recommend you first learn a bit about it online.

Running Kafka on Docker is even harder, and requires some deep understanding of Docker and Kafka.

Thankfully, I have created a project to make things easy for beginners: <https://github.com/simplesteph/kafka-stack-docker-compose>

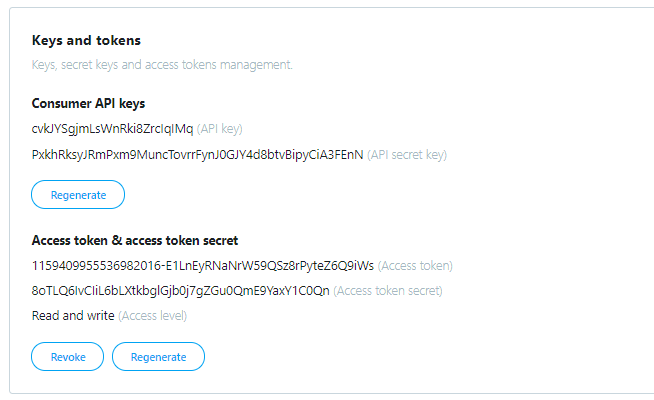
#### ****Installing Docker:****

See here: <https://store.docker.com/search?type=edition&offering=community>

Note to Windows users: Docker is really hard to setup on your machines, may require Windows 10, might require disabling Hyper-V, and tinkering VirtualBox. If you cannot setup Docker, the best help you will get will be on Google.

**Overall for all Windows users but Windows 10, I recommend not to run Kafka in Docker, and use the binaries instead**

**https://kafka-niranjan-test.com**



### Keys and tokens

Keys, secret keys and access tokens management.

##### Consumer API keys

cvkJYSgjmLsWnRki8ZrcIqIMq (API key)

PxkhRksyJRmPxm9MuncTovrrFynJ0GJY4d8btvBipyCiA3FEnN (API secret key)

Regenerate

##### Access token & access token secret

1159409955536982016-E1LnEyRNaNrW59QSz8rPyteZ6Q9iWs (Access token)

8oTLQ6IvCIiL6bLXtkbglGjb0j7gZGu0QmE9YaxY1C0Qn (Access token secret)

Read and write (Access level)

RevokeRegenerate