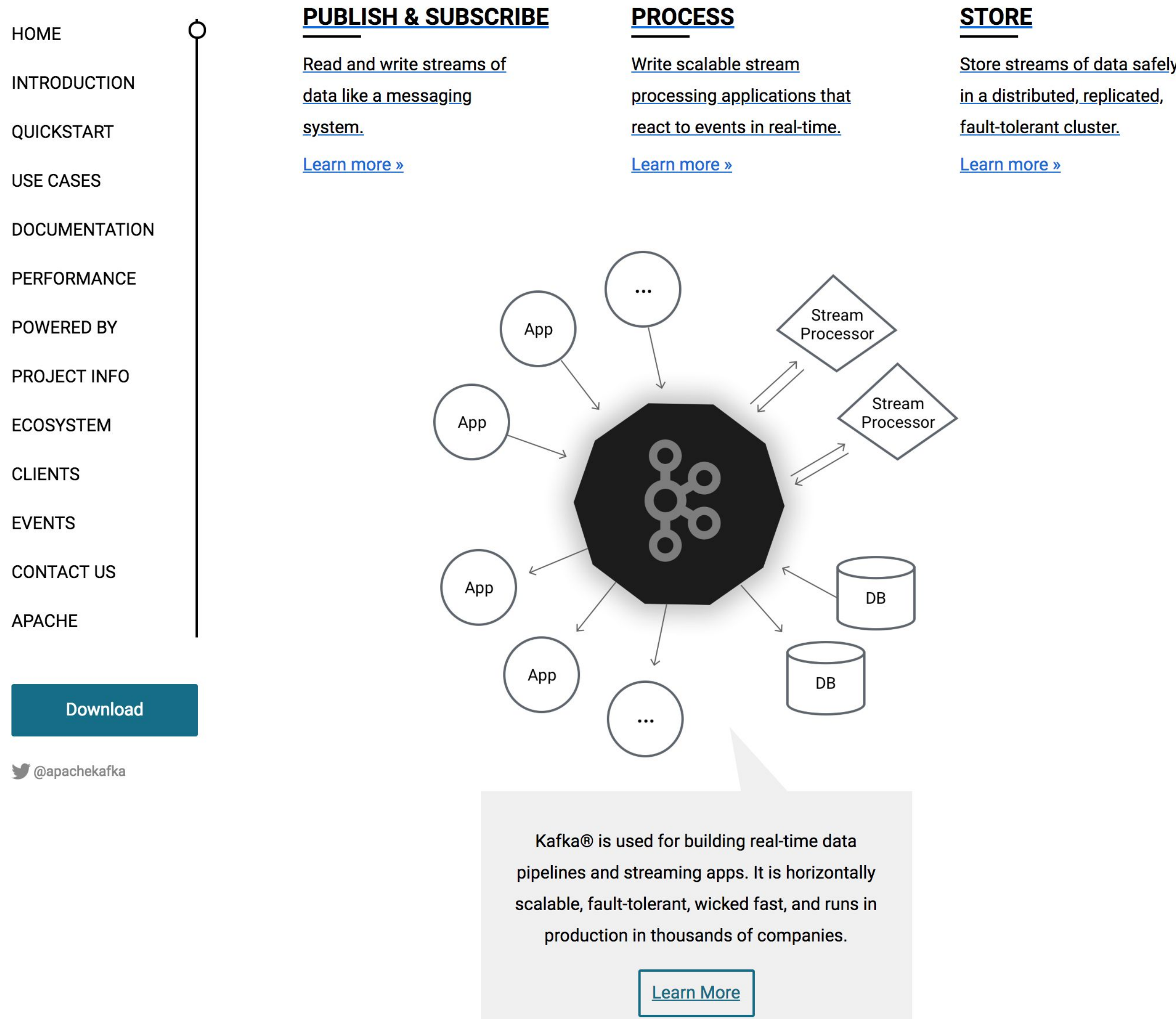
Kafka & Kafka Streams

MINING STREAMING DATA

Apache Kafka

kafka.apache.org



HOME
INTRODUCTION
QUICKSTART
USE CASES
DOCUMENTATION
PERFORMANCE
POWERED BY
PROJECT INFO
ECOSYSTEM
CLIENTS
EVENTS
CONTACT US
APACHE

Download

@apachekafka

PUBLISH & SUBSCRIBE

Read and write streams of data like a messaging system.

Learn more »

PROCESS

Write scalable stream processing applications that react to events in real-time.

Learn more »

STORE

Store streams of data safely in a distributed, replicated, fault-tolerant cluster.

Learn more »

Kafka® is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.

Learn More

Introduction

A streaming platform has three key capabilities:

- Publish and subscribe to streams of records, similar to a message queue.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Few basic Kafka concepts

- Kafka is run as a cluster on one or more servers (Kafka brokers)
- The Kafka cluster stores streams of records in categories called topics.
- Each record consists of a key (optional), a value and timestamp

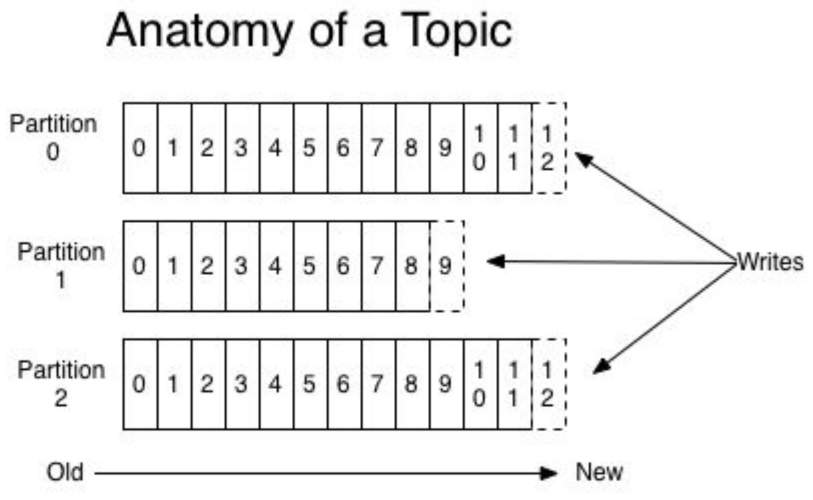
Kafka Topic

A Kafka topic ...

- Is a named stream of records
- Can have zero, one, or many consumers that subscribe to the data written to it.

For each topic, the Kafka cluster maintains a partitioned log.

Records are stored in a partition based on record key or round-robin if the key is missing.

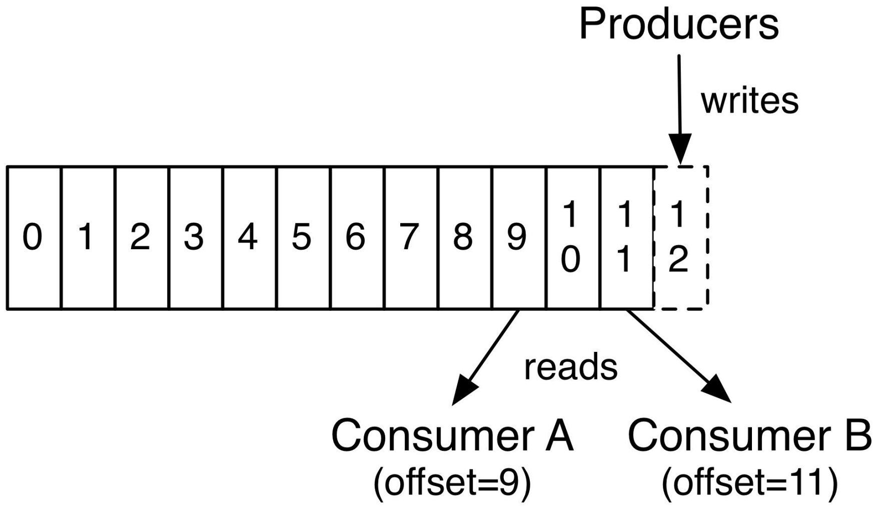


Kafka Topic Partitions

Many partitions can handle an arbitrary amount of data and writes.

Records in the partitions are each assigned a sequential id number called the offset.

An offset uniquely identifies each record within the partition



Kafka Consumers

A consumer is a subscriber to a Kafka topic

Consumers are grouped into a consumer group

A consumer group maintains its offset per topic partition

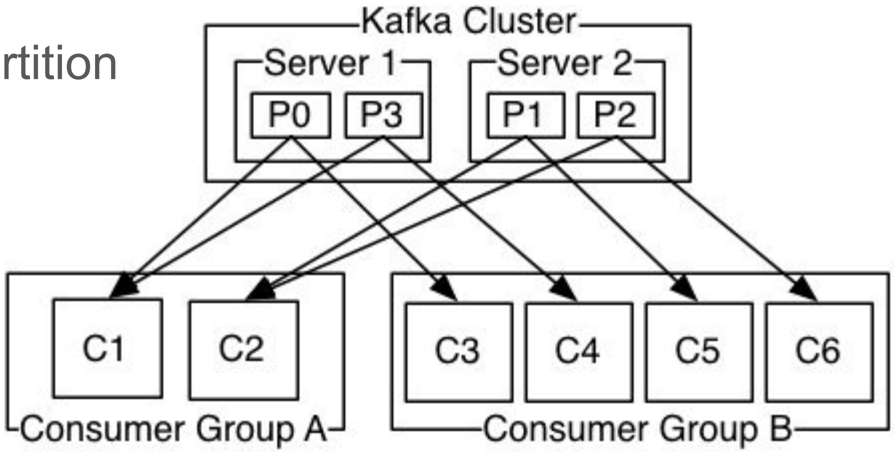
Scaling Kafka Consumers

Kafka scales consumers by partition

Each consumer gets its share of partitions.

A partition can only be used by one consumer in a consumer group at a time

A consumer can have more than one partition



Kafka Producer

A Kafka producer sends records to topics.

Records are send to a partition based on record key or round-robin if the key is missing.

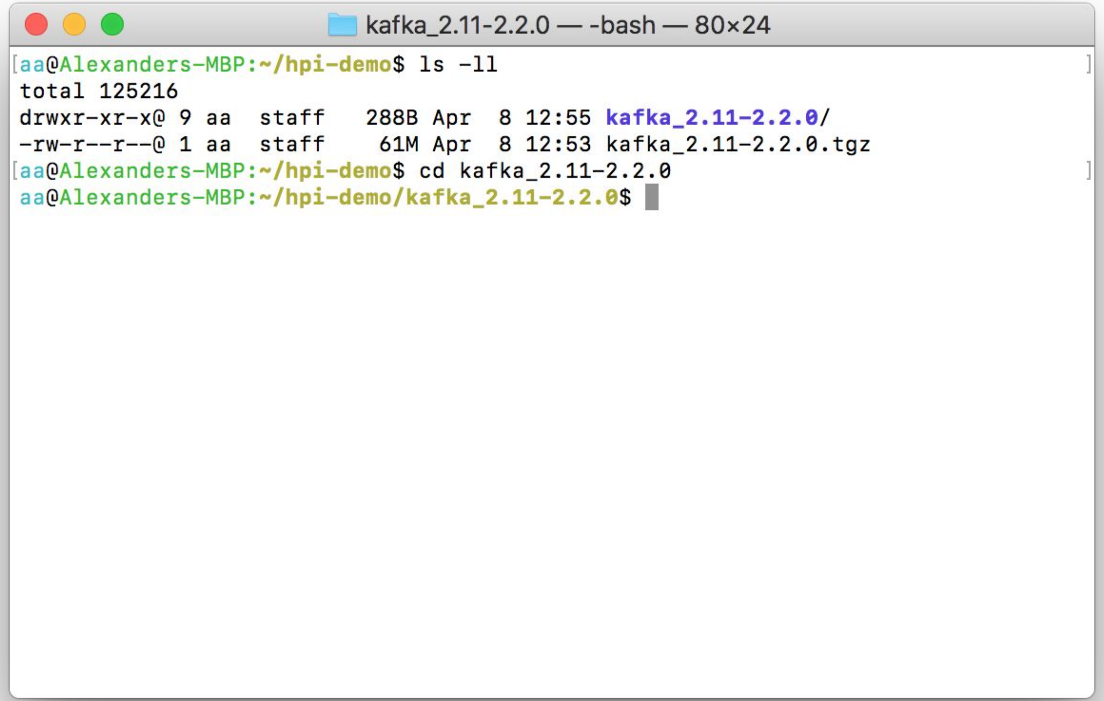
Records with the same key get sent to the same partition.

More to learn

- Replication
- Consistency and durability levels
- Serializer/Deserializer
- Kafka brokers
- Storage
- Zookeeper
- ...

Download Apache Kafka

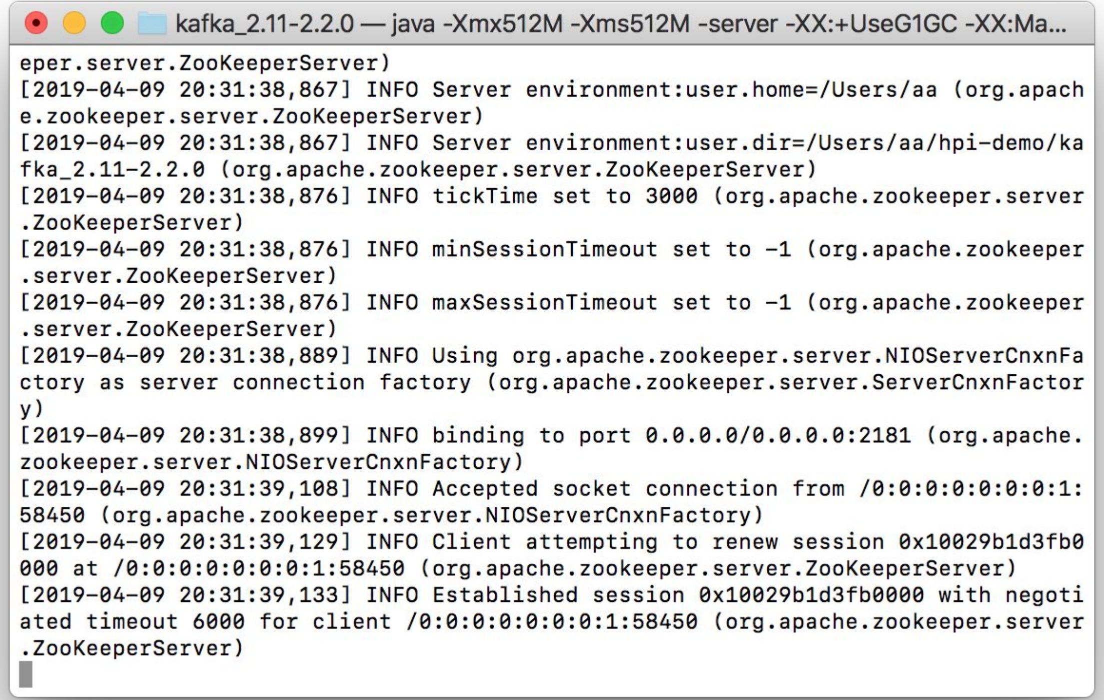
kafka.apache.org


kafka_2.11-2.2.0 — -bash — 80×24

```
aa@Alexanders-MBP:~/hpi-demo$ ls -ll
total 125216
drwxr-xr-x@ 9 aa  staff   288B Apr  8 12:55 kafka_2.11-2.2.0/
-rw-r--r--@ 1 aa  staff    61M Apr  8 12:53 kafka_2.11-2.2.0.tgz
aa@Alexanders-MBP:~/hpi-demo$ cd kafka_2.11-2.2.0
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ 
```

Start Zookeeper

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



```
kafka_2.11-2.2.0 — java -Xmx512M -Xms512M -server -XX:+UseG1GC -XX:Ma...
```

```
eper.server.ZooKeeperServer)
[2019-04-09 20:31:38,867] INFO Server environment:user.home=/Users/aa (org.apach
e.zookeeper.server.ZooKeeperServer)
[2019-04-09 20:31:38,867] INFO Server environment:user.dir=/Users/aa/hpi-demo/ka
fka_2.11-2.2.0 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-09 20:31:38,876] INFO tickTime set to 3000 (org.apache.zookeeper.server
.ZooKeeperServer)
[2019-04-09 20:31:38,876] INFO minSessionTimeout set to -1 (org.apache.zookeeper
.server.ZooKeeperServer)
[2019-04-09 20:31:38,876] INFO maxSessionTimeout set to -1 (org.apache.zookeeper
.server.ZooKeeperServer)
[2019-04-09 20:31:38,889] INFO Using org.apache.zookeeper.server.NIOServerCnxnFa
ctory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactor
y)
[2019-04-09 20:31:38,899] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.
zookeeper.server.NIOServerCnxnFactory)
[2019-04-09 20:31:39,108] INFO Accepted socket connection from /0:0:0:0:0:0:0:1:
58450 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2019-04-09 20:31:39,129] INFO Client attempting to renew session 0x10029b1d3fb0
000 at /0:0:0:0:0:0:0:1:58450 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-09 20:31:39,133] INFO Established session 0x10029b1d3fb0000 with negoti
ated timeout 6000 for client /0:0:0:0:0:0:0:1:58450 (org.apache.zookeeper.server
.ZooKeeperServer)
```

Small Demo

Start Kafka cluster on your machine.

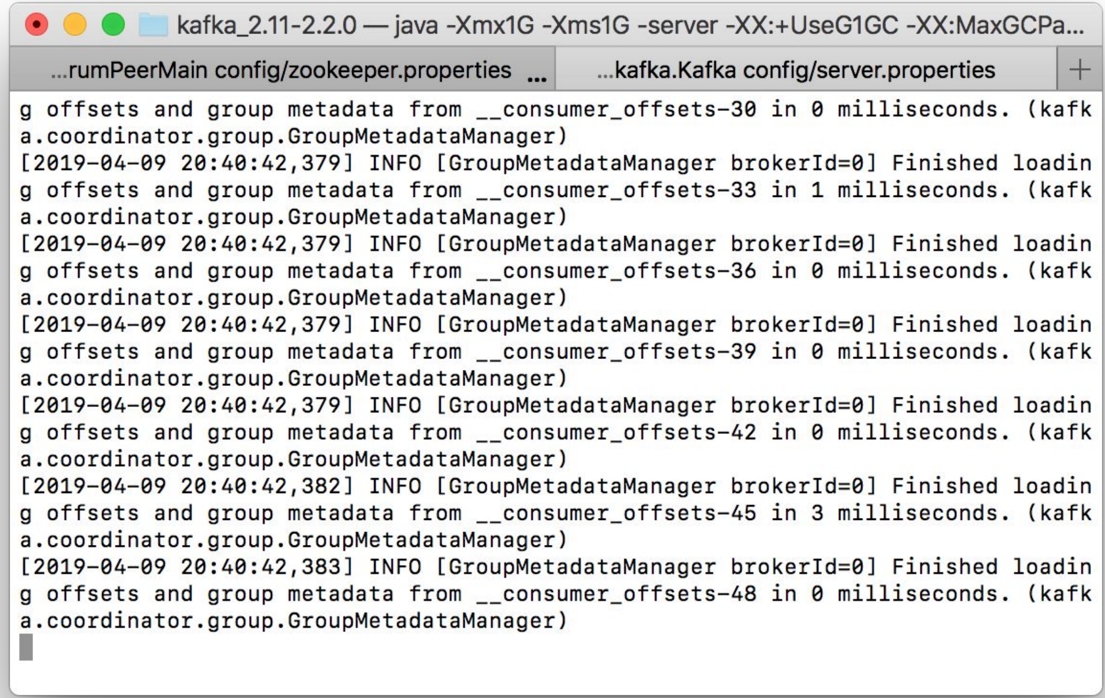
Write example input data to a Kafka topic, using the console producer

Process the input data with WordCountDemo, an example Java application that uses the Kafka Streams library.

Inspect the output data of the application, using the console consumer

Start Apache Kafka

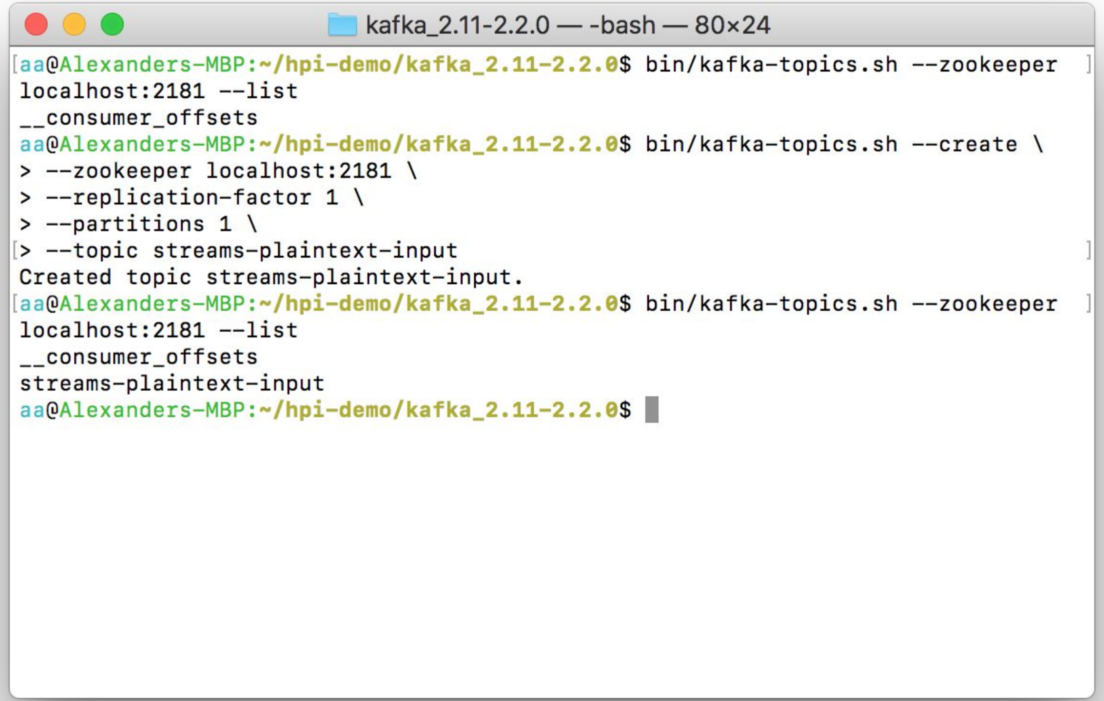
bin/kafka-server-start.sh config/server.properties


kafka_2.11-2.2.0 — java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPa...
...rumPeerMain config/zookeeper.properties
...kafka.Kafka config/server.properties

```
g offsets and group metadata from __consumer_offsets-30 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,379] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-33 in 1 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,379] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-36 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,379] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-39 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,379] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-42 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,382] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-45 in 3 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2019-04-09 20:40:42,383] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-48 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
```

Create a Kafka input topic

```
bin/kafka-topics.sh --create \
        --zookeeper localhost:2181 \
        --replication-factor 1 \
        --partitions 1 \
        --topic streams-plaintext-input
```



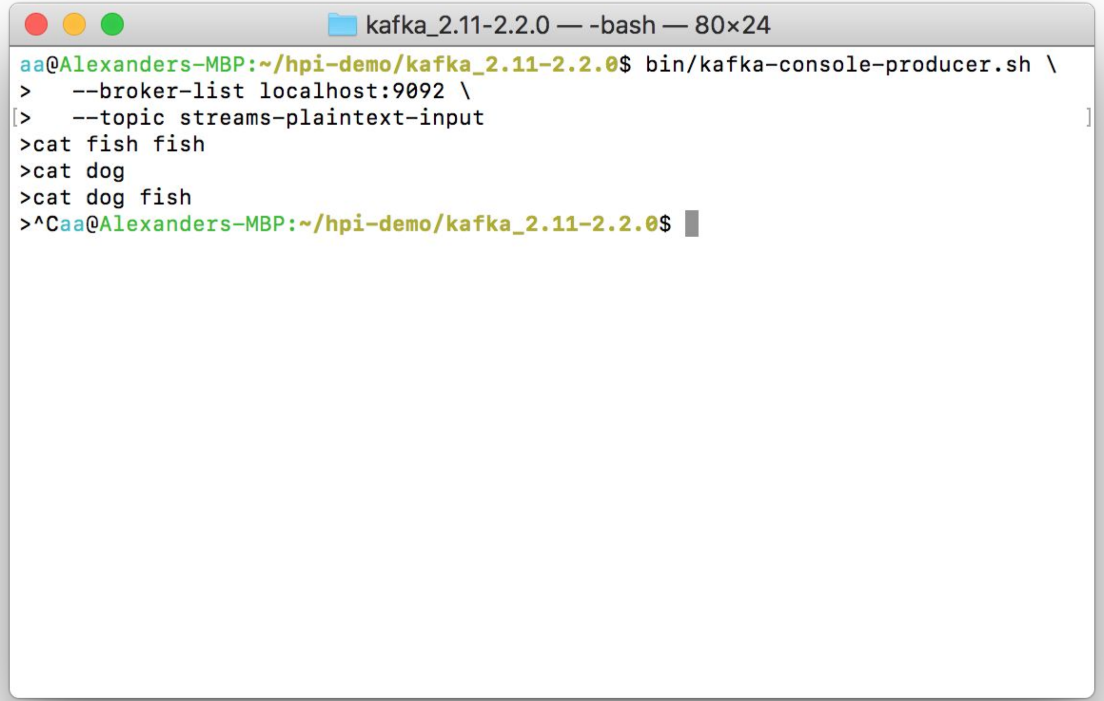
```
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-topics.sh --zookeeper
localhost:2181 --list
__consumer_offsets
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-topics.sh --create \
> --zookeeper localhost:2181 \
> --replication-factor 1 \
> --partitions 1 \
> --topic streams-plaintext-input
Created topic streams-plaintext-input.
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-topics.sh --zookeeper
localhost:2181 --list
__consumer_offsets
streams-plaintext-input
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$
```

Create a Kafka output topic

```
bin/kafka-topics.sh --create \
    --zookeeper localhost:2181 \
    --replication-factor 1 \
    --partitions 1 \
    --topic streams-wordcount-output
```

Publish data to input topic

```
bin/kafka-console-producer.sh \
        --broker-list localhost:9092 \
        --topic streams-plaintext-input
```

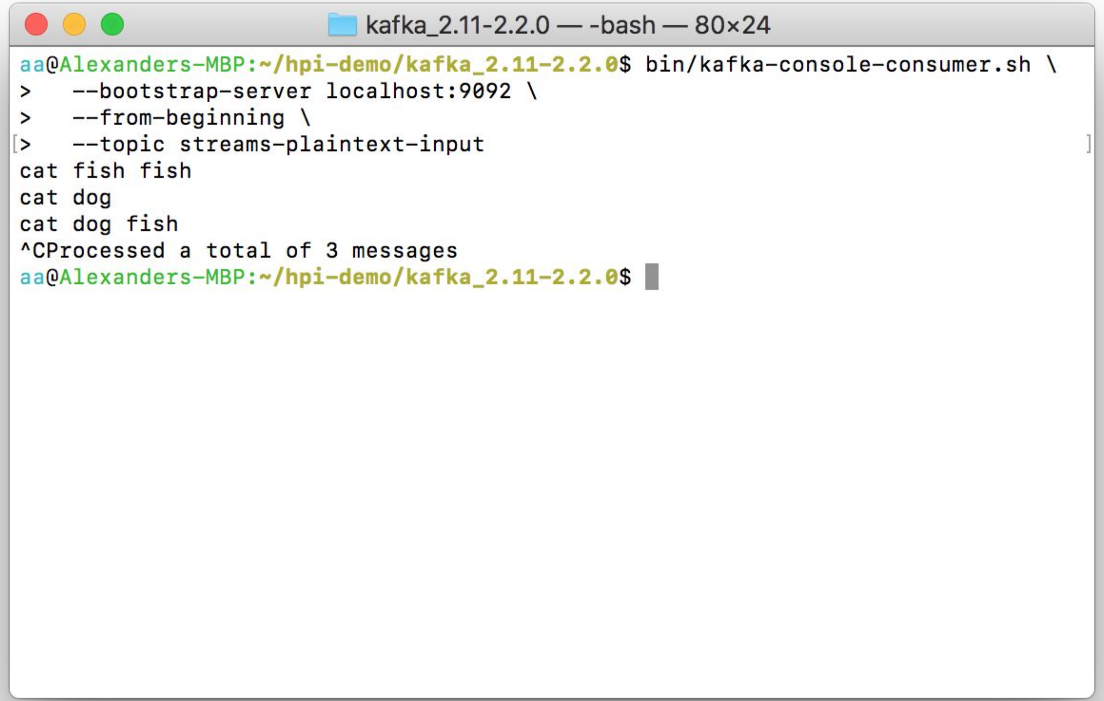


kafka_2.11-2.2.0 — -bash — 80×24

```
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-console-producer.sh \
>   --broker-list localhost:9092 \
[>   --topic streams-plaintext-input                                            ]
>cat fish fish
>cat dog
>cat dog fish
>^Caa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ 
```

Subscribe data from input topic

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 \
        --from-beginning \
        --topic streams-plaintext-input
```



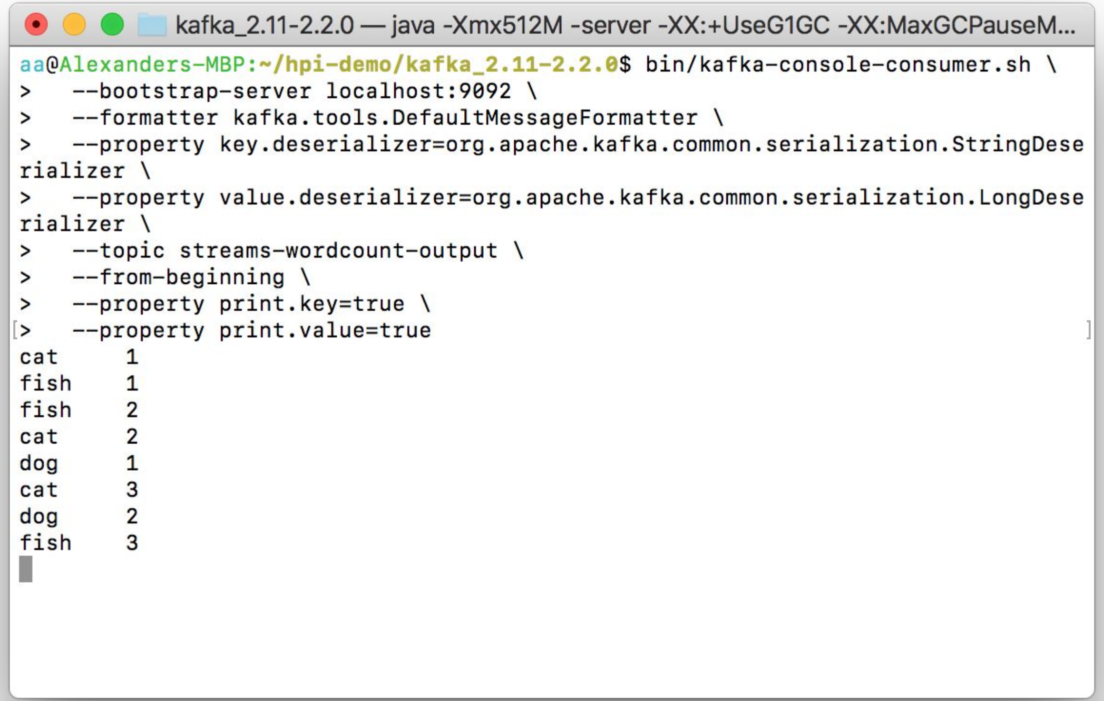
kafka_2.11-2.2.0 — -bash — 80×24

```
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-console-consumer.sh \
>   --bootstrap-server localhost:9092 \
>   --from-beginning \
>   --topic streams-plaintext-input
cat fish fish
cat dog
cat dog fish
^CProcessed a total of 3 messages
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$
```

Process data in input topic with Kafka Streams App

bin/kafka-run-class.sh

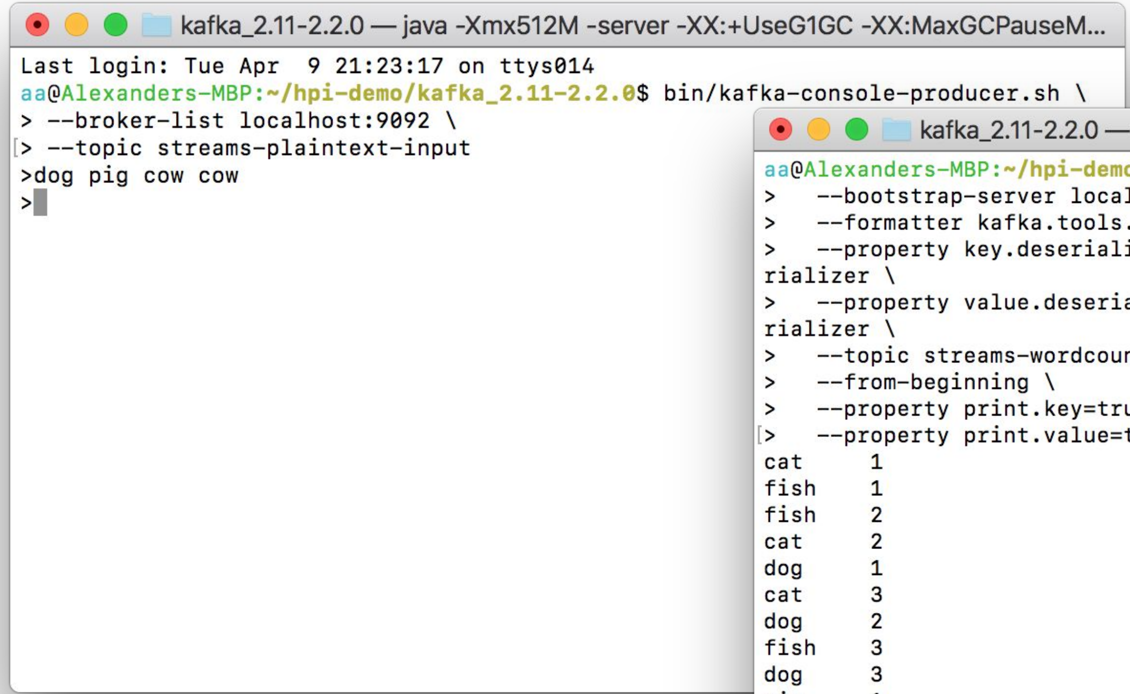
org.apache.kafka.streams.examples.wordcount.WordCountDemo



kafka_2.11-2.2.0 — java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseM...

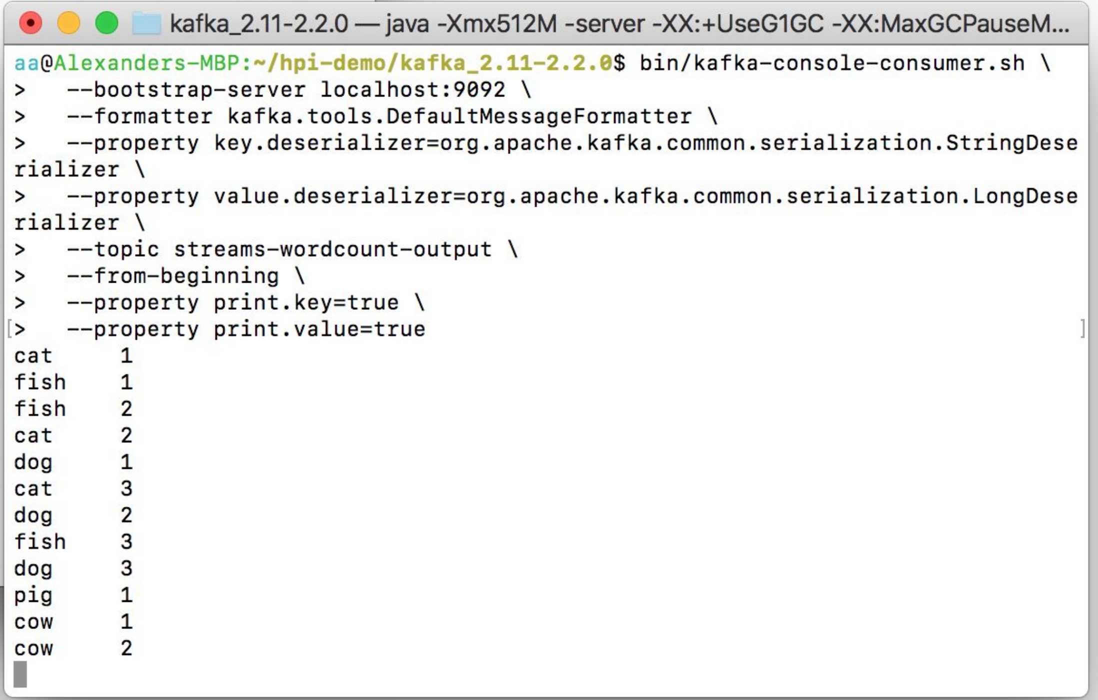
```
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-console-consumer.sh \
>   --bootstrap-server localhost:9092 \
>   --formatter kafka.tools.DefaultMessageFormatter \
>   --property key.deserializer=org.apache.kafka.common.serialization.StringDese
rializer \
>   --property value.deserializer=org.apache.kafka.common.serialization.LongDese
rializer \
>   --topic streams-wordcount-output \
>   --from-beginning \
>   --property print.key=true \
>   --property print.value=true
cat     1
fish    1
fish    2
cat     2
dog     1
cat     3
dog     2
fish    3
```

Kafka Streams Processing



kafka_2.11-2.2.0 — java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseM...

```
Last login: Tue Apr  9 21:23:17 on ttys014
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-console-producer.sh \
> --broker-list localhost:9092 \
> --topic streams-plaintext-input
>dog pig cow cow
>
```



kafka_2.11-2.2.0 — java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseM...

```
aa@Alexanders-MBP:~/hpi-demo/kafka_2.11-2.2.0$ bin/kafka-console-consumer.sh \
>   --bootstrap-server localhost:9092 \
>   --formatter kafka.tools.DefaultMessageFormatter \
>   --property key.deserializer=org.apache.kafka.common.serialization.StringDese
rializer \
>   --property value.deserializer=org.apache.kafka.common.serialization.LongDese
rializer \
>   --topic streams-wordcount-output \
>   --from-beginning \
>   --property print.key=true \
>   --property print.value=true
cat     1
fish    1
fish    2
cat     2
dog     1
cat     3
dog     2
fish    3
dog     3
pig     1
cow     1
cow     2
```

Writing a Kafka Streams Application

Create a new Maven project in your Java IDE, for example IntelliJ IDEA

Add dependencies to pom.xml, see
https://kafka.apache.org/10/documentation/streams/developer-guide/write-streams

Run WordCoutDemo in your Java IDE
https://github.com/apache/kafka/blob/trunk/streams/examples/src/main/java/org/apache/kafka/streams/examples/wordcount/WordCountDemo.java

Kafka Streams Applications

Kafka Streams is a library for developing applications for processing records from Apache Kafka topics.

Any Java application that makes use of the Kafka Streams library is considered a Kafka Streams application.

```
public final class WordCountDemo {
        public static void main(final String[] args) {
                …
                final StreamsBuilder builder = new StreamsBuilder();
                final KStream<String, String> source = builder.stream("streams-plaintext-input");
                ...
        }
}
```

WordCountDemo

KStream<String, String>	<null, "cat fish fish">
flatMapValues	<null, "cat">, <null, "fish">, <null, "fish">
groupBy((key, value) -> value)	(<"cat", "cat">), (<"fish", "fish">, <"fish", "fish">)
Count occurences in each group	<"cat", 1>, <"fish", 2>

KStream & KGroupedStream

Abstraction of a record stream (of key-value pairs), where each data record represents an INSERT

Created directly from one or many Kafka topics

```
final KStream<String, String> source = builder.stream("streams-plaintext-input");
```

Comes with a rich set of operators (KStream API)

- filter
- flatMapValues
- join
- groupBy
- ...

KTable

Abstraction of a record stream (of key-value pairs), where each data record represents an UPSERT (UPDATE or INSERT)

Related to Kafka Log Compaction, see https://kafka.apache.org/documentation.html#compaction

Records with null values (so-called tombstone records) are deleted

Putting it all together

```
final StreamsBuilder builder = new StreamsBuilder();
final KStream<String, String> source = builder.stream("streams-plaintext-input");

final KTable<String, Long> counts = source
        .flatMapValues(value ->
                Arrays.asList(value.toLowerCase(Locale.getDefault()).split(" "))
        .groupBy((key, value) -> value)
        .count();
// need to override value serde to Long type
counts.toStream().to("streams-wordcount-output", Produced.with(Serdes.String(),
Serdes.Long()));
final KafkaStreams streams = new KafkaStreams(builder.build(), props);
```

Boilerplate

Configuration (props),
for example APPLICATION_ID_CONFIG ("streams-wordcount")

Graceful shutdown: Java shutdown hook to run streams.close() when the stream application terminates.

https://github.com/apache/kafka/blob/trunk/streams/examples/src/main/java/org/apache/kafka/streams/examples/wordcount/WordCountDemo.java

More to learn

- High-level Streams DSL
- Low-level Processor API
- StateStores
- ...