Kafka in OSX El-Capitan

♬ / 5 Oct 15

Apache Kafka is a highly-scalable publish-subscribe messaging system that can serve as the data backbone in distributed applications. With Kafka's Producer-Consumer model it becomes easy to implement multiple data consumers that do live monitoring as well persistent data storage for later analysis.

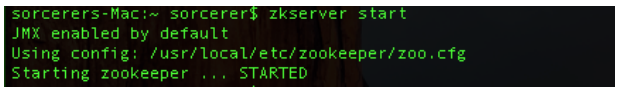
STEP 1: Installation, the best way to install the latest version of the Kafka server on OS X and to keep it up to date is via Homebrew.

```
$ brew search kafka
$ brew install kafka
```

this will also install all dependencies, like ***zookeeper*** which is required to run ***kafka*** server.

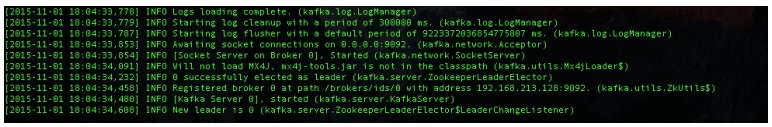
STEP 2: we need to start Zookeeper before we can start Kafka.

```
$ zkserver start
```



STEP 3: Once Zookeeper is up start the Kafka server.

```
$ cd /usr/local/Cellar/kafka/0.10.1.0/bin
$ kafka-server-start.sh /usr/local/etc/kafka/server.properties
```

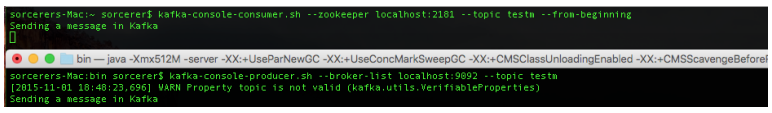


STEP 4: Start a consumer, Kafka also has a command line consumer that will dump out messages to standard out.

```
$ kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning
Sending a message in Kafka
```

STEP 5: Send some messages, Kafka comes with a command line client producer that will take input from console and send it out as messages to the Kafka cluster. By default each line will be sent as a separate message. The topic test is created automatically when messages are sent to it. Omitting logging you should see something like this:

```
$ kafka-console-producer.sh --broker-list localhost:9092 --topic test
Sending a message in Kafka
```



Run each of the above commands in different terminal, then you should now be able to type messages into the producer terminal and see them appear in the consumer terminal. Both of these command line tools have additional options.

STEP 6: Setting up a multi-broker cluster

So far we have been running against a single broker, but that's no fun. For Kafka, a single broker is just a cluster of size one, so nothing much changes other than starting a few more broker instances. But just to get feel for it, let's expand our cluster to three nodes (still all on our local machine).

First we make a config file for each of the brokers:

```
cp config/server.properties config/server-1.properties
cp config/server.properties config/server-2.properties
```

Now edit these new files and set the following properties:

```
config/server-1.properties:
    broker.id=1
    port=9093
    log.dir=/tmp/kafka-logs-1

config/server-2.properties:
```

```
    broker.id=2
    port=9094
    log.dir=/tmp/kafka-logs-2
```

The `broker.id` property is the unique and permanent name of each node in the cluster. We have to override the port and log directory only because we are running these all on the same machine and we want to keep the brokers from trying to all register on the same port or overwrite each others data.

We already have Zookeeper and our single node started, so we just need to start the two new nodes. However, this time we have to override the JMX port used by java too to avoid clashes with the running node:

```
JMX_PORT=9997 bin/kafka-server-start.sh config/server-1.properties
...
JMX_PORT=9998 bin/kafka-server-start.sh config/server-2.properties
...
```

Now create a new topic with a replication factor of three:

```
bin/kafka-create-topic.sh --zookeeper localhost:2181 --replica 3 --partition 1 --topic my-repli
```

Okay but now that we have a cluster how can we know which broker is doing what? To see that run the "list topics" command:

```
bin/kafka-list-topic.sh --zookeeper localhost:2181
topic: my-replicated-topic  partition: 0  leader: 1  replicas: 1,2,0  isr: 1,2,0
topic: test                 partition: 0  leader: 0  replicas: 0      isr: 0
```

Here is an explanation of output:

- "leader" is the node responsible for all reads and writes for the given partition. Each node would be the leader for a randomly selected portion of the partitions.
- "replicas" is the list of nodes that are supposed to server the log for this partition regardless of whether they are currently alive.
- "isr" is the set of "in-sync" replicas. This is the subset of the replicas list that is currently alive and caught-up to the leader.

Note that both topics we created have only a single partition (partition 0). The original topic has no replicas and so it is only present on the leader (node 0), the replicated topic is present on all three nodes with node 1 currently acting as leader and all replicas in sync.

As before let's publish a few messages message:

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-replicated-topic
...
my test message 1
my test message 2
^C
```

Now consume this message:

```
bin/kafka-console-consumer.sh --zookeeper localhost:2181 --from-beginning --topic my-replicated
...
my test message 1
my test message 2
```

Now let's test out fault-tolerance. Kill the broker acting as leader for this topic's only partition:

```
pkill -9 -f server-1.properties
```

Leadership should switch to one of the slaves:

```
bin/kafka-list-topic.sh --zookeeper localhost:2181
...
topic: my-replicated-topic  partition: 0    leader: 2   replicas: 1,2,0 isr: 2
topic: test partition: 0    leader: 0   replicas: 0 isr: 0
```

And the messages should still be available for consumption even though the leader that took the writes originally is down:

```
bin/kafka-console-consumer.sh --zookeeper localhost:2181 --from-beginning --topic my-replicated
...
my test message 1
my test message 2
```

Advertisements

Twitter Google LinkedIn 1 More

Like

One blogger likes this.

5 Oct 15 in DS. Tags: Kafka, OSX

Related posts

PySpark/IPython in OSX El-Capitan

← PySpark/IPython in OSX El-Capitan

Wanderbar.. →

10 thoughts on “Kafka in OSX El-Capitan”

kaveti 15 Sep 16 at 18:21

awesome buddy. nice work (y
)

★ Like

Randy 16 Nov 16 at 05:49

Have never used brew, zookeeper, or kafka before. I tried these steps: after successfully installing brew, the first line gives

$ brew search kafka
kafka kafkacat librdkafka

…but then hangs (waited ten minutes, then control-C’ed it) Looking on the web, I see others report hanging with no apparent resolution to the issue.

I ignored this as a problem and went on to the next steps. I was able to start zookeeper. But this is what I got in reply to step 3:
$ kafka-server-start.sh /usr/local/etc/kafka/server.properties
-bash: kafka-server-start.sh: command not found

help?

Just FYI:

$ ls /usr/local/etc/kafka
connect-console-sink.properties connect-file-source.properties log4j.properties zookeeper.properties
connect-console-source.properties connect-log4j.properties producer.properties
connect-distributed.properties connect-standalone.properties server.properties
connect-file-sink.properties consumer.properties tools-log4j.properties

so there is something in my system abut kafka now.

★ Like

♬ 16 Nov 16 at 06:28

First you will need to go inside /usr/local/Cellar/kafka/0.10.1.0/bin
(i have added the cd step to post also now thanks for pointing)
Then check inside what files are there if kafka is latest version then cmd might as well be kafka-server-start instead of kafka-server-start.sh

brew spills all the installs inside /usr/local/Cellar/

I again did a brew search kafka…..it took a while…..but it did not hang

★ Like

 Randy 16 Nov 16 at 07:12

Thanks for your help! I did indeed find the commands inside Cellar/ as you suspected, and was able to execute steps 4 and 5 by simply dropping the ".sh" … however, I got this warning …

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

…which I ignored. Also, I the use of consumer and producer in your steps was reversed from what I expect. That is, I typed the message in the producer, and I saw it print out in the consumer, as the names suggest would be appropriate. However, your steps 4 and 5 seem to indicate the reverse roles. I will try step 6 later. Thanks again…

★ Like

 Randy 16 Nov 16 at 07:42

Forgive my comments on the reversal of consumer and producer, I was mislead by the appearance of line numbered "2" in step 4, which is something that will appear in the near future (after step 5), and I thought it was to be typed in.

My brew still hangs, though it seems to indicate with bold text and a green checkmark that kafka is now cool.

Thanks again for your post and help.

★ Like

 zishanjms 25 Mar 17 at 21:41

How to override the JMX port used by java.

★ Like

 ♬ 27 Mar 17 at 20:59

does the setting mentioned under step 6 not work ?

★ Like

 zishanjms 28 Mar 17 at 10:40

It worked. Thanks 🙂

★ Liked by 1 person

 XnX 21 Apr 17 at 03:23

Hi ,
After step 3 ,
terminal seems to be stuck :

”

[2017-04-20 23:19:55,907] INFO [Kafka Server 0], started (kafka.server.KafkaServer)

[2017-04-20 23:29:55,689] INFO [Group Metadata Manager on Broker 0]: Removed 0 expired offsets in 0 milliseconds. (kafka.coordinator.GroupMetadataManager)

[2017-04-20 23:39:55,726] INFO [Group Metadata Manager on Broker 0]: Removed 0 expired offsets in 1 milliseconds. (kafka.coordinator.GroupMetadataManager)

[2017-04-20 23:49:55,772] INFO [Group Metadata Manager on Broker 0]: Removed 0 expired offsets in 1 milliseconds. (kafka.coordinator.GroupMetadataManager)

”

am i missing anything ?

★ Like

21 Apr 17 at 07:01

Do you see a message like ‘INFO [Kafka Server 0], started (kafka.server.KafkaServer)’
as the last message > if yes it has started leave the terminal window as is and open other
else you can run kafka-server-start /usr/local/etc/kafka/server.properties &
which will run in the background mode

★ Like

Comments are closed.

Create a free website or blog at WordPress.com.