



INCEPTEZ TECHNOLOGIES KAFKA WORKOUTS

Install Kafka:

1) Go to the below path.

```
cd /home/hduser/install/
```

2) Extract the tarball,

```
tar xvfz kafka_2.10-0.10.1.0.tgz
sudo mv kafka_2.10-0.10.1.0 /usr/local/kafka
sudo chown -R hduser:hadoop /usr/local/kafka
```

3) Start the Zookeeper Coordination service:

```
zookeeper-server-start.sh -daemon /usr/local/kafka/config/zookeeper.properties
```

4) Start the Kafka server:

```
kafka-server-start.sh -daemon /usr/local/kafka/config/server.properties
```

Workouts:

Create a topic (with one replica and one partition)

```
kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic tk1
```

We can now see that topic if we run the list topic command:

```
kafka-topics.sh --list --zookeeper localhost:2181
```

Produce messages using Producer:

Kafka comes with a command line client that will take input from a file or from standard input and send it out as messages to the Kafka cluster. By default each line will be sent as a separate message. Run the producer and then type a few messages into the console to send to the server.

```
kafka-console-producer.sh --broker-list localhost:9092 --topic tk1
```

Start a Consumer

Kafka also has a command line consumer that will dump out messages to standard output.

```
kafka-console-consumer.sh --zookeeper localhost:2181 --topic tk1 --from-beginning
```

Only look at the incremental logs

```
kafka-console-consumer.sh --zookeeper localhost:2181 --topic tk1
```

If you have each of the above commands running in a different terminal then you should now be able to type messages into the producer terminal and see them appear in the consumer terminal.

Setting up a multi-broker cluster

So far we have been running against a single broker. 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:

```
cd /usr/local/kafka
```

```
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:

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 all trying to register on the same port or overwrite each others data.

```
vi config/server-1.properties
```

```
listeners=PLAINTEXT://:9093
broker.id=1 port=9093
log.dir=/tmp/kafka-logs-1
```

```
vi config/server-2.properties
```

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

We already have Zookeeper and our single node started, so we just need to start the two new nodes:

```
kafka-server-start.sh -daemon $KAFKA_HOME/config/server-1.properties kafka-server-start.sh -daemon
$KAFKA_HOME/config/server-2.properties
```

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

```
kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic tr43
```

```
kafka-topics.sh --list --zookeeper localhost:2181
```

Want to alter partitions use --alter

```
kafka-topics.sh --zookeeper localhost:2181 --alter --topic tr43 --partitions 4
```

How can we know which broker is doing what? To see that run the "describe topics" command:

```
kafka-topics.sh --describe --zookeeper localhost:2181 --topic tr43
```

We can run the same command on the original topic we created to see where it is:

```
kafka-topics.sh --describe --zookeeper localhost:2181 --topic tk1
```

Let's publish a few messages to our new topic tr43:

```
kafka-console-producer.sh --broker-list localhost:9092 --topic tr43
```

Now let's consume these messages:

```
kafka-console-consumer.sh --zookeeper localhost:2181 --topic tr43 --from-beginning
```

Now let's test out fault-tolerance. Broker 1 was acting as the leader so let's kill it:

```
ps -ef | grep server-1.properties
```

```
7564 ttys002
```

```
kill -9 7564
```

Leadership has switched to one of the slaves and node 1 is no longer in the in-sync replica set:

```
kafka-topics.sh --describe --zookeeper localhost:2181 --topic tr43
```

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

```
kafka-console-consumer.sh --zookeeper localhost:2181 --from-beginning --topic tr43
```

If you want to delete the topic (mark for deletion, at the time of flush will be deleted else use delete.topic.enable=true)

```
kafka-topics.sh --delete --zookeeper localhost:2181 --topic tk1
```