Lab 5.3: Configuring Producer Durability

Welcome to the session 5 lab 3. The work for this lab is done in `~/kafka-training/labs/lab5.3`. In this lab, you are going to set up producer durability for our advanced producer.

Find the latest version of this lab here.

Lab Configuring Producer Durability

In this lab we will configure producer durability.

Set default acks to all

ACTION EDIT StockPriceKafkaProducer and set Producer config acks to all (this is the default). This means that all ISRs in-sync replicas have to respond for producer write to go through.

StockPriceKafkaProducer.java

```
public class StockPriceKafkaProducer {

    private static Producer<String, StockPrice>
                                    createProducer() {
        final Properties props = new Properties();
        ...

        //Set number of acknowledgments - acks - default is all
        props.put(ProducerConfig.ACKS_CONFIG, "all");
        return new KafkaProducer<>(props);
    }
}
```

Notice we set the Producer config property "acks" (ProducerConfig.ACKS_CONFIG) to all.

Kafka Broker Config for Min.Insync.Replicas

Next let's set the the min.insync.replicas to three. This will force at least three in-sync replicas (ISRs) have to respond for our producer to get an ack from the Kafka Broker. NOTE: We have three brokers in this lab, thus all three have to be up for the Producer to work.

Ensure min.insync.replicas is set to three in all of the broker config files (server-0.properties, server-1.properties and server-2.properties).

ACTION EDIT config/server-0.properties and min.insync.replicas=3

ACTION EDIT config/server-1.properties and min.insync.replicas=3

ACTION EDIT config/server-2.properties and min.insync.replicas=3

Run this lab. Run it. Run Servers. Run Producer. Kill 1 Broker.

If not already, startup ZooKeeper. Now startup three Kafka brokers (or ensure they are running) using scripts described earlier. From the IDE run StockPriceKafkaProducer class. From the terminal kill one of the Kafka Brokers.

Now look at the logs for the StockPriceKafkaProducer, you should see Caused by: org.apache.kafka.common.errors.NotEnoughReplicasException. Note that the Messages are rejected since there are fewer in-sync replicas than required. Repeat this with only 2 min.insync.replicas set. (Change config and restart brokers and restart producer). Observe the behavior of using 2 for min.insync.replicas vs. three.

ACTION RUN brokers and zookeeper if needed.

ACTION RUN StockPriceKafkaProducer from the IDE

ACTION KILL 1 of the Kafka Brokers

ACTION OBSERVE Producer terminal messages

ACTION EDIT config/server-0.properties and min.insync.replicas=2

ACTION EDIT config/server-1.properties and min.insync.replicas=2

ACTION EDIT config/server-2.properties and min.insync.replicas=2

ACTION RESTART brokers

ACTION KILL 1 of the Kafka Brokers

ACTION OBSERVE Producer terminal messages

Why did the send fail?

The producer config ProducerConfig.ACKS_CONFIG (acks config for producer) was set to “all”. This settings expects leader to only give successful ack after all followers ack the send was written to their log. The Broker config min.insync.replicas set to 3. At least three in-sync replicas must respond before send is considered successful. Since we took one broker out and only had three to begin with, it forces the send to fail since the send can not get three acks from ISRs.

Modify durability to leader only

Change StockPriceKafkaProducer acks config to 1 `props.put(ProducerConfig.ACKS_CONFIG, "1"` , i.e., leader sends ack after write to log. From the IDE run StockPriceKafkaProducer again. From the terminal kill one of the Kafka Brokers. Notice that the StockPriceKafkaProducer now runs normally.

ACTION EDIT StockPriceKafkaProducer and set Producer config acks to "1".

ACTION RUN all brokers and zookeeper if needed.

ACTION RUN StockPriceKafkaProducer from the IDE

ACTION KILL 1 of the Kafka Brokers

ACTION OBSERVE Producer terminal messages or lack thereof

Why did the send not fail for acks 1?

Setting the Producer config ProducerConfig.ACKS_CONFIG (acks config for producer) to “1”. This setting expects leader to only give successful ack after it writes to its log. Replicas still get replication but leader does not wait for replication to send ack. Broker Config min.insync.replicas is still set to 3, but this config only gets looked at if acks=“all”.

Running describe topics before and after stoping broker

Try the last steps again. Stop a server while producer is running. Then run describe-topics.sh. Then Rerun server you stopped and run describe-topics.sh again. Observe the changes to ISRs and partition to broker ownership.

bin/describe-topics.sh

```
#!/usr/bin/env bash

cd ~/kafka-training

# List existing topics
kafka/bin/kafka-topics.sh --describe \
    --topic stock-prices \
    --zookeeper localhost:2181
```

The script bin/describe-topics.sh calls kafka-topics.sh to describe the topic layout with regards to brokers and partitions.

ACTION REPEAT Last experiment, but this time run bin/describe-topics.sh and observe.

Retry with acks = 0

Run the last example again (servers, and producer), but this time set acks to 0. Run all three brokers then take one away. Then take another broker away. Try Run describe-topics. Take all of the brokers down and continue to run the producer. What do you think happens? When you are done, change acks back to acks=all.

ACTION EDIT StockPriceKafkaProducer and set Producer config acks to "0".

ACTION RUN all brokers and zookeeper if needed.

ACTION RUN StockPriceKafkaProducer from the IDE

ACTION REPEAT Last experiment, but this time run bin/describe-topics.sh and observe.

Lab Review

Look at the following listing.

Describe topic listing

```
$ bin/describe-topics.sh
Topic:stock-prices    PartitionCount:3    ReplicationFactor:3
Configs:min.insync.replicas=2
    Topic: stock-prices    Partition: 0    Leader: 2    Replicas: 1,2,0    Isr: 2
    Topic: stock-prices    Partition: 1    Leader: 2    Replicas: 2,0,1    Isr: 2
    Topic: stock-prices    Partition: 2    Leader: 2    Replicas: 0,1,2    Isr: 2
```

- How would you describe the above?
- How many servers are likely running out of the three?
- Would the producer still run with acks=all? Why or Why not?
- Would the producer still run with acks=1? Why or Why not?
- Would the producer still run with acks=0? Why or Why not?
- Which broker is the leader of partition 1?

- How would you describe the above? Two servers are down. Broker 0 and Broker 1.
- How many servers are likely running out of the three? Just the third broker.
- Would the producer still run with acks=all? Why or Why not? No. Only one server is running.

- Would the producer still run with acks=1? Why or Why not? No. Yes. The 3rd server owns the partitions.
- Would the producer still run with acks=0? Why or Why not? Yes.
- Which broker is the leader of partition 1? The third server one with broker id 2.