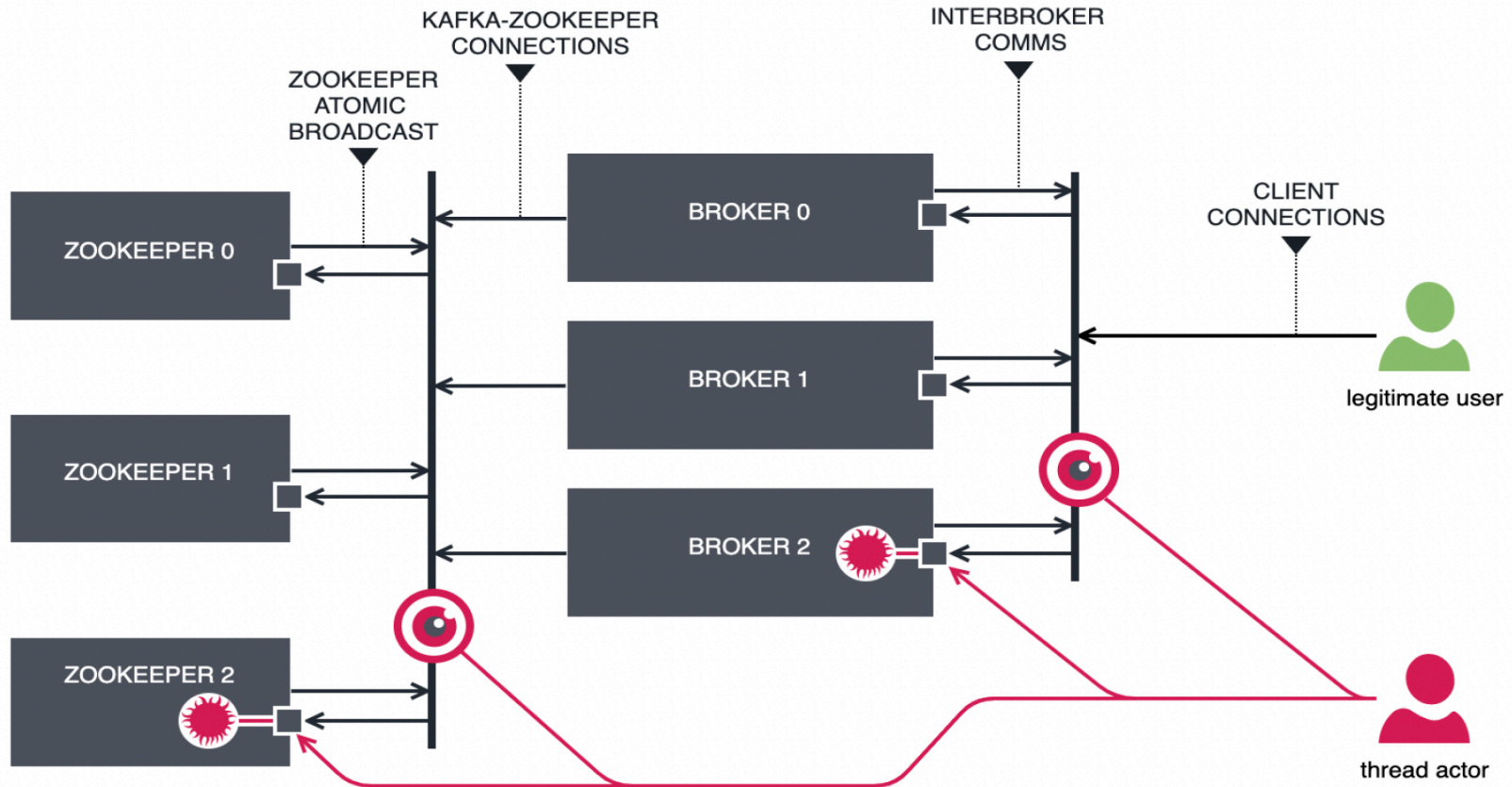


Kafka Cluster and Failover

Kafka Brokers - Cluster

Tos

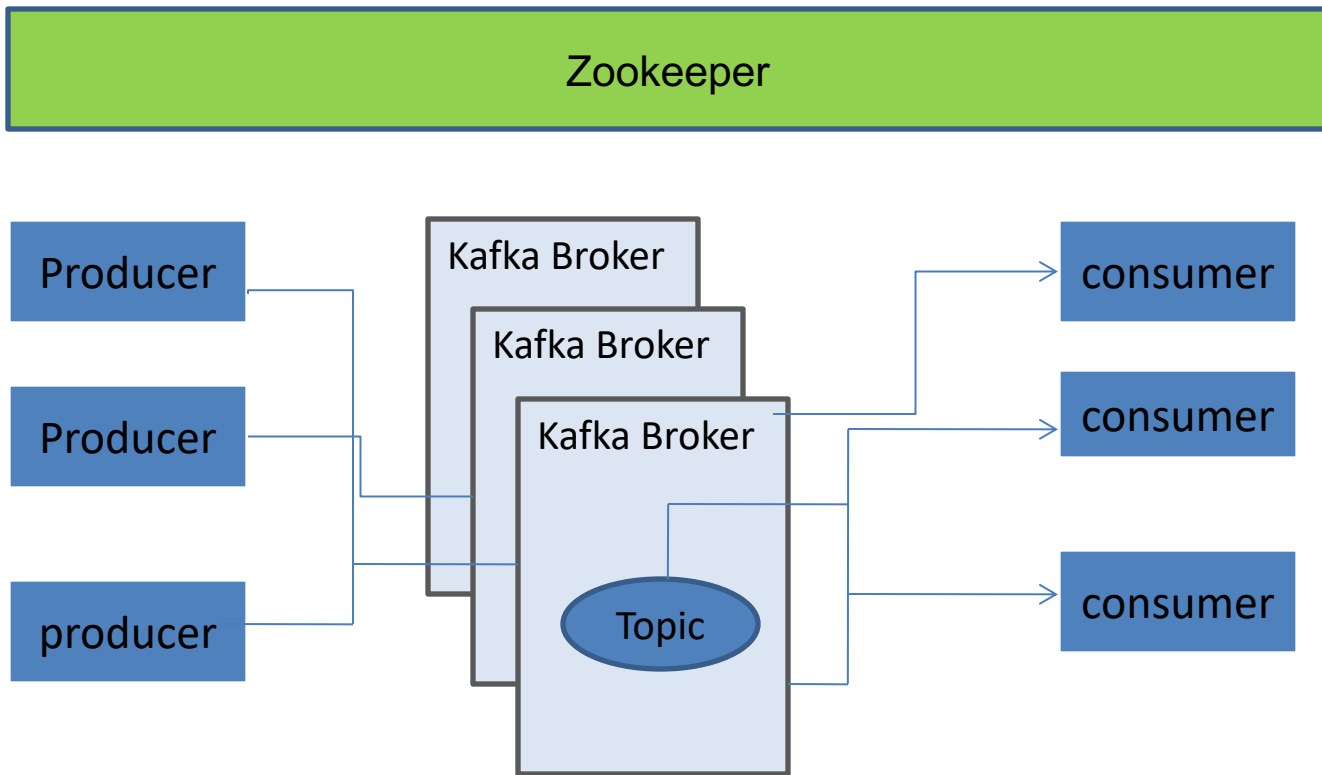


- ❖ Kafka Cluster is made up of multiple Kafka Brokers
- ❖ Each Broker has an ID (number)
- ❖ Brokers contain topic log partitions
- ❖ Connecting to one broker bootstraps client to entire cluster
- ❖ Start with at least three brokers, cluster can have, 10, 100, 1000 brokers if needed

- ❖ Topic ***Partitions*** can be ***replicated***
 - ❖ across ***multiple nodes*** for failover
- ❖ Topic should have a replication factor greater than 1
 - ❖ (2, or 3)
- ❖ ***Failover***
 - ❖ if one Kafka Broker goes down then Kafka Broker with ISR (in-sync replica) can serve data

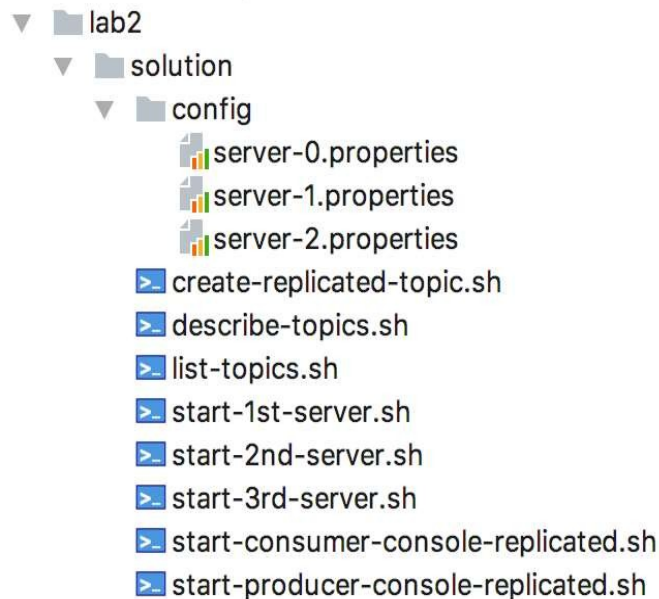
Zoo Keeper does coordination for Kafka cluster

Tos



- ❖ Replication of Kafka Topic Log partitions allows for failure of a rack or AWS availability zone
 - ❖ You need a replication factor of at least 3
- ❖ ***Kafka Replication*** is for ***Failover***
- ❖ ***Replicator/Mirror Maker*** is used for ***Disaster Recovery***
- ❖ Mirror Maker replicates a Kafka cluster to another data-centre or AWS region
 - ❖ Called mirroring since replication happens within a cluster

- ❖ Run many Kafka Brokers
- ❖ Create a replicated topic
- ❖ Demonstrate Pub / Sub
- ❖ Demonstrate load balancing consumers
- ❖ Demonstrate consumer failover
- ❖ Demonstrate broker failover



- ❖ If not already running, start up ZooKeeper
 - ❖ Shutdown Kafka from first lab
- ❖ Copy server properties for three brokers
 - ❖ Modify properties files, Change port, Change Kafka log location
- ❖ Start up many Kafka server instances
- ❖ Create Replicated Topic
- ❖ Use the replicated topic

```
~/kafka-training
[$ ./run-zookeeper.sh
```


Create three new server- n.properties files

- ❖ Copy existing *server.properties* to *server-0.properties*, *server-1.properties*, *server-2.properties*
- ❖ Change *server-0.properties* to use *log.dirs*. “./logs/kafka-logs-0”
- ❖ Change *server-1.properties* to use *port 9093*, *broker id 1*, and *log.dirs* “./logs/kafka-logs-1”
- ❖ Change *server-2.properties* to use *port 9094*, *broker id 2*, and *log.dirs* “./logs/kafka-logs-2”

Modify server-x.properties

Tos

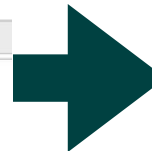
server-0.properties x	
1	broker.id=0
2	port=9092
3	log.dirs=./logs/kafka-0
server-1.properties x	
1	broker.id=1
2	port=9093
3	log.dirs=./logs/kafka-1
server-2.properties x	
1	broker.id=2
2	port=9094
3	log.dirs=./logs/kafka-2

- ❖ Each have different ***broker.id***
- ❖ Each have different ***log.dirs***
- ❖ Each had different ***port***

Create Startup scripts for three Kafka servers

Tos

```
start-1st-server.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3
4  cd ~/kafka-training
5
6  ## Run Kafka
7  kafka/bin/kafka-server-start.sh \
8      "$CONFIG/server-0.properties"
9
start-2nd-server.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  ## Run Kafka
6  kafka/bin/kafka-server-start.sh \
7      "$CONFIG/server-1.properties"
8
9
start-3rd-server.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  ## Run Kafka
6  kafka/bin/kafka-server-start.sh \
7      "$CONFIG/server-2.properties"
8
```



```
start-2nd-server.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  ## Run Kafka
6  kafka/bin/kafka-server-start.sh \
7      "$CONFIG/server-1.properties"
8
9
```



Passing properties files from last step

Run Servers

```
$ ./start-1st-server.sh  
[2017-05-15 11:18:00,168] INFO KafkaConfig values:  
    advertised.host.name = null  
    advertised.listeners = null  
    advertised.port = null
```

```
$ ./start-2nd-server.sh  
[2017-05-15 11:18:24,980] INFO KafkaConfig values:  
    advertised.host.name = null  
    advertised.listeners = null  
    advertised.port = null  
    authorizer.class.name =
```

```
~/kafka-training/lab2/solution  
$ ./start-3rd-server.sh  
[2017-05-15 11:19:04,129] INFO KafkaConfig  
    advertised.host.name = null  
    advertised.listeners = null  
    advertised.port = null  
    authorizer.class.name =
```

Create Kafka replicated topic my-failsafe-topic

Tos

```
create-replicated-topic.sh x
1  #!/usr/bin/env bash
2
3  cd ~/kafka-training
4
5  kafka/bin/kafka-topics.sh --create \ --bootstrap-server localhost:9092
6
7      --replication-factor 3 \
8      --partitions 13 \
9      --topic my-failsafe-topic
10
```

- ❖ **Replication Factor** is set to 3
- ❖ Topic name is **my-failsafe-topic**
- ❖ **Partitions** is 13

```
$ ./create-replicated-topic.sh
Created topic "my-failsafe-topic".
```

```
start-consumer-console-replicated.sh x
1  #!/usr/bin/env bash
2  cd ~/kafka-training
3
4  kafka/bin/kafka-console-consumer.sh \
5      --bootstrap-server localhost:9094,localhost:9092 \
6      --topic my-failsafe-topic \
7      --from-beginning
8
```

- ❖ Pass list of Kafka servers to bootstrap-server
- ❖ We pass two of the three
- ❖ Only one needed, it learns about the rest

Start Kafka Producer

Tos

```
>_ start-producer-console-replicated.sh x
1  #!/usr/bin/env bash
2  cd ~/kafka-training
3
4  kafka/bin/kafka-console-producer.sh \
5  --broker-list localhost:9092,localhost:9093 \
6  --topic my-failsafe-topic
7
```

- ❖ Start producer
- ❖ Pass list of Kafka Brokers

Kafka 1 consumer and 1 producer running

Tos

```
Last login: Mon May 15 11:25:19 on ttys007
```

```
~/kafka-training/lab2/solution
```

```
[$ ./start-producer-console-replicated.sh
```

```
Hi mom
```

```
How are you?
```

```
How are things going?
```

```
Good!
```

```
[
```

```
Last login: Mon May 15 11:19:27 on ttys006
```

```
~/kafka-training/lab2/solution
```

```
[$ ls
```

```
config
```

```
start-2
```

```
create-replicated-topic.sh
```

```
start-3
```

```
list-topics.sh
```

```
start-c
```

```
start-1st-server.sh
```

```
start-p
```

```
~/kafka-training/lab2/solution
```

```
[$ ./start-consumer-console-replicated.sh
```

```
Hi mom
```

```
How are you?
```

```
How are things going?
```

```
Good!
```

```
[
```


Start a second and third consumer

```
$ ./start-producer-console-replicated.sh
Hi mom
How are you?
How are things going?
Good!
message 1
message 2
message 3

Last login: Mon May 15 11:28:21 on ttys011
~/kafka-training/lab2/solution
$ ./start-co

Last login: Mon May 15 11:35:19 on ttys007
~/kafka-training/lab2/solution
$ ./start-consumer-console-replicated.sh
Good!
How are things going?
How are you?
Hi mom
message 1
message 2
message 3

Last login: Mon May 15 11:35:35 on ttys011
~/kafka-training/lab2/solution
$ ./start-consumer-console-replicated.sh
Good!
How are things going?
How are you?
Hi mom
message 1
message 2
message 3
```

- ❖ Acts like pub/sub
- ❖ Each consumer in its own group
- ❖ Message goes to each
- ❖ How do we load share?

Running consumers in same group

Tos

> start-consumer-console-replicated.sh x

1 **#!/usr/bin/env bash**

2 **cd ~/kafka-training**

3

4 **kafka/bin/kafka-console-consumer.sh **

5 **--bootstrap-server localhost:9094,localhost:9092 **

6 **--topic my-failsafe-topic **

7 **--consumer-property group.id=mygroup**

8 **--from-beginning**

9

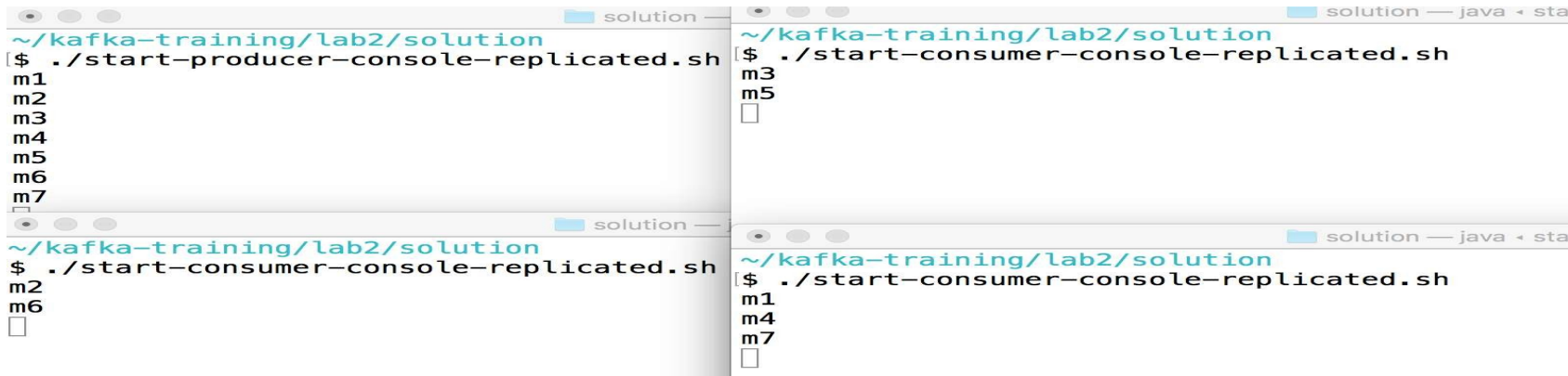
❖ Modify start consumer script

❖ Add the consumers to a group called mygroup

❖ Now they will share load

Start up three consumers again

Tos



The image displays four terminal windows arranged in a 2x2 grid, all showing the directory `~/kafka-training/lab2/solution`. The top-left window shows the execution of `./start-producer-console-replicated.sh`, which outputs messages `m1` through `m7`. The top-right window shows `./start-consumer-console-replicated.sh` outputting `m3` and `m5`. The bottom-left window shows `./start-consumer-console-replicated.sh` outputting `m2` and `m6`. The bottom-right window shows `./start-consumer-console-replicated.sh` outputting `m1`, `m4`, and `m7`. Each window has a cursor at the end of the last line of output.

```
~/kafka-training/lab2/solution
[$] ./start-producer-console-replicated.sh
m1
m2
m3
m4
m5
m6
m7

~/kafka-training/lab2/solution
[$] ./start-consumer-console-replicated.sh
m3
m5

~/kafka-training/lab2/solution
[$] ./start-consumer-console-replicated.sh
m2
m6

~/kafka-training/lab2/solution
[$] ./start-consumer-console-replicated.sh
m1
m4
m7
```

- ❖ Start up producer and three consumers
- ❖ Send 7 messages
- ❖ Notice how messages are spread among 3 consumers

Consumer Failover

Tos

```
~/kafka-training/lab2/solution
$ ./start-producer-console-replicated.sh
m1
m2
m3
m4
m5
m6
m10
m7
m12
m8
m13
m9
m10
m11
m12
m13
m14
```

- ❖ Kill one consumer
- ❖ Send seven more messages
- ❖ Load is spread to remaining consumers
- ❖ Failover WORK!

Create Kafka Describe Topic

Tos

```
describe-topics.sh x
1  #!/usr/bin/env bash
2
3  cd ~/kafka-training
4
5  # List existing topics
6  kafka/bin/kafka-topics.sh --describe \
7    --topic my-failsafe-topic \
8    --bootstrap-server localhost:9092
9
```

- ❖ —describe will show list partitions, ISRs, and partition leadership

Use Describe Topics

```
[ $ ./describe-topics.sh
Topic:my-failsafe-topic PartitionCount:13 ReplicationFactor:3 Configs:
Topic: my-failsafe-topic Partition: 0 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
Topic: my-failsafe-topic Partition: 1 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
Topic: my-failsafe-topic Partition: 2 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
Topic: my-failsafe-topic Partition: 3 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
Topic: my-failsafe-topic Partition: 4 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
Topic: my-failsafe-topic Partition: 5 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
Topic: my-failsafe-topic Partition: 6 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
Topic: my-failsafe-topic Partition: 7 Leader: 0 Replicas: 0,1,2 Isr: 0,1,2
Topic: my-failsafe-topic Partition: 8 Leader: 1 Replicas: 1,2,0 Isr: 1,2,0
Topic: my-failsafe-topic Partition: 9 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0
Topic: my-failsafe-topic Partition: 10 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1
Topic: my-failsafe-topic Partition: 11 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2
Topic: my-failsafe-topic Partition: 12 Leader: 2 Replicas: 2,0,1 Isr: 2,0,1
```

- ❖ Lists which broker owns (leader of) which partition
- ❖ Lists Replicas and ISR (replicas that are up to date)
- ❖ Notice there are 13 topics

Test Broker Failover: Kill 1st server

Tos

Kill the first server

```
~/kafka-training/lab2/solution
```

```
$ kill `ps aux | grep java | grep server-0.properties | tr -s " " | cut -d " " -f2`
```

use Kafka topic describe to see that a new leader was elected!

```
$ ./describe-topics.sh
```

```
Topic:my-failsafe-topic PartitionCount:13 ReplicationFactor:3 Configs:
Topic: my-failsafe-topic Partition: 0 Leader: 2 Replicas: 2,0,1 Isr: 2,1
Topic: my-failsafe-topic Partition: 1 Leader: 1 Replicas: 0,1,2 Isr: 1,2
Topic: my-failsafe-topic Partition: 2 Leader: 1 Replicas: 1,2,0 Isr: 1,2
Topic: my-failsafe-topic Partition: 3 Leader: 2 Replicas: 2,1,0 Isr: 2,1
Topic: my-failsafe-topic Partition: 4 Leader: 2 Replicas: 0,2,1 Isr: 2,1
Topic: my-failsafe-topic Partition: 5 Leader: 1 Replicas: 1,0,2 Isr: 1,2
Topic: my-failsafe-topic Partition: 6 Leader: 2 Replicas: 2,0,1 Isr: 2,1
Topic: my-failsafe-topic Partition: 7 Leader: 1 Replicas: 0,1,2 Isr: 1,2
Topic: my-failsafe-topic Partition: 8 Leader: 1 Replicas: 1,2,0 Isr: 1,2
Topic: my-failsafe-topic Partition: 9 Leader: 2 Replicas: 2,1,0 Isr: 2,1
Topic: my-failsafe-topic Partition: 10 Leader: 2 Replicas: 0,2,1 Isr: 2,1
Topic: my-failsafe-topic Partition: 11 Leader: 1 Replicas: 1,0,2 Isr: 1,2
Topic: my-failsafe-topic Partition: 12 Leader: 2 Replicas: 2,0,1 Isr: 2,1
```

Show Broker Failover Worked

Tos

```
~/kafka-training/lab2/solution
$ ./start-producer-console-replicated.sh
m1
m2
m3
m4
m5
m6
m10
m12
m13
m9
[2017-05-15 12:00:58,462] WARN Auto-commit
ta='', my-failsafe-topic-3=OffsetAndMetad
ffset=1, metadata='', my-failsafe-topic-1
etAndMetadata{offset=1, metadata=''}, my-f
fe-topic-5=OffsetAndMetadata{offset=2, met
triable exception. You should retry commit
Coordinator)
m16
[]
```

- ❖ Send two more messages from the producer
- ❖ Notice that the consumer gets the messages
- ❖ Broker Failover WORKS

- ❖ Why did the three consumers not load share the messages at first?
- ❖ How did we demonstrate failover for consumers?
- ❖ How did we demonstrate failover for producers?
- ❖ What tool and option did we use to show ownership of partitions and the ISRs?

Lab : Kafka cluster