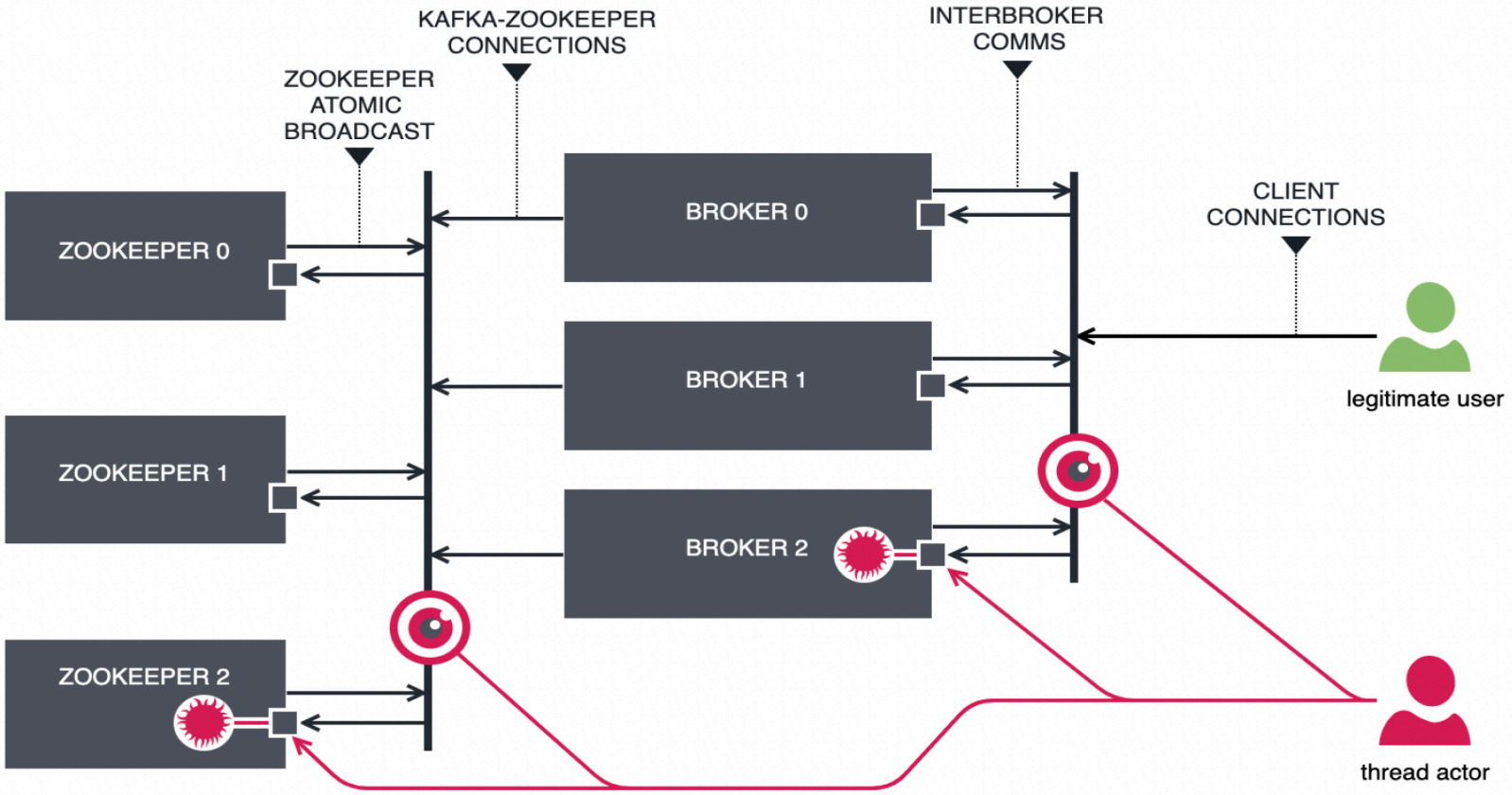
# Kafka Cluster  and Failover

# Kafka Brokers - Cluster

Tos



# Kafka Brokers

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

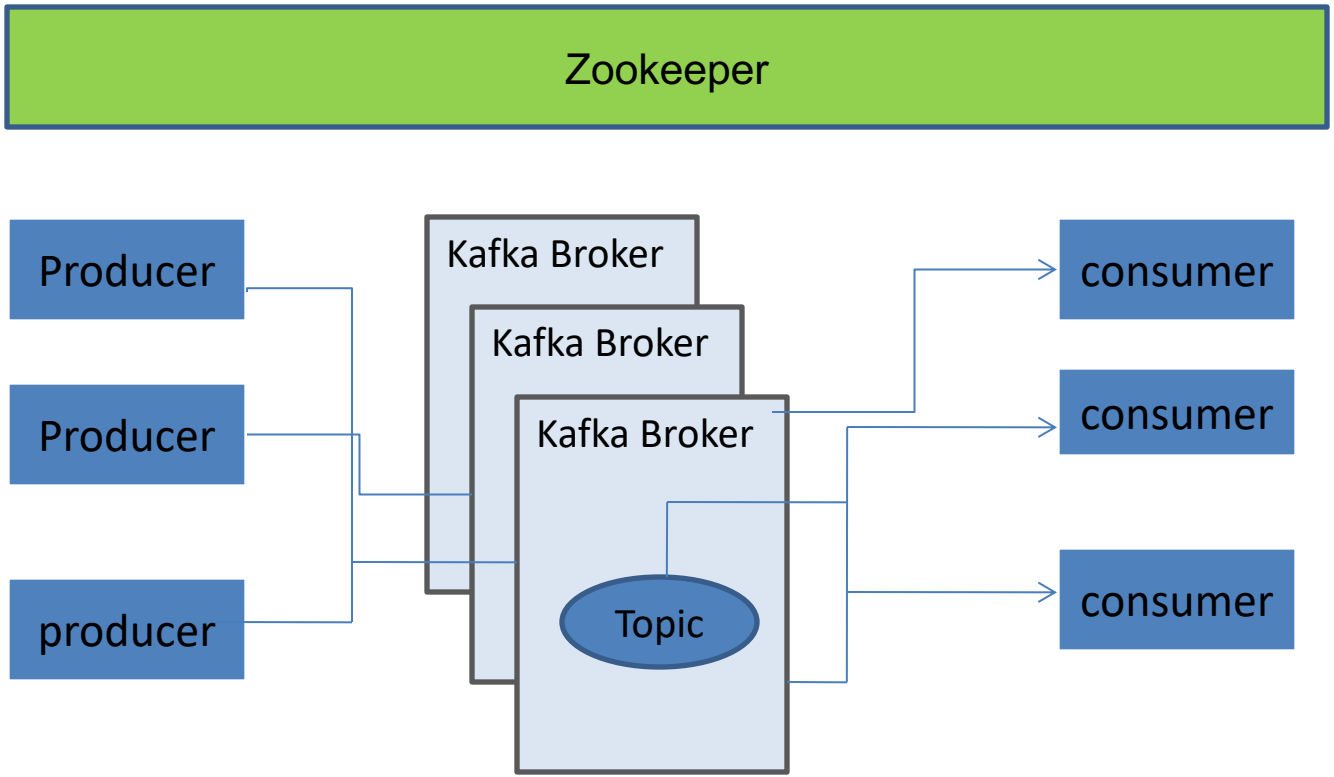
# Kafka Cluster, Failover, ISRs

Tos

- 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

# Zookeeper does coordination for Kafka cluster

Tos



# Failover vs. Disaster Recovery

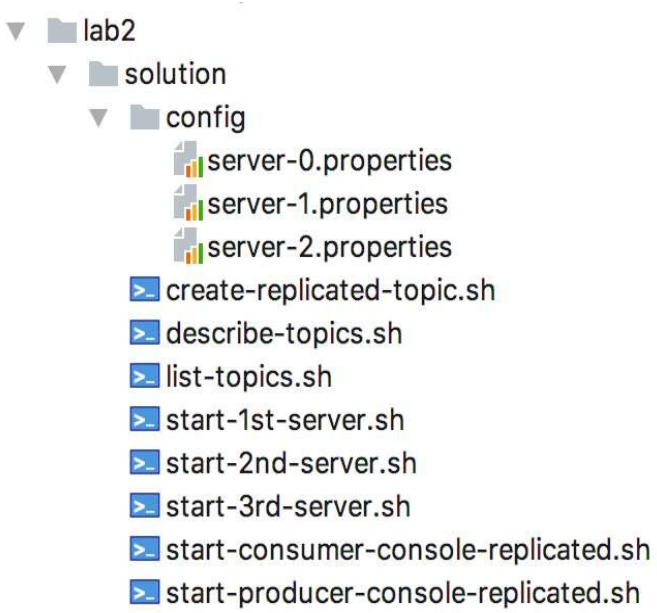
Tos

- 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

# Objectives

Tos

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



# Running many nodes

Tos

- 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

# Create three new server- n.properties files

Tos

- 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

```
1  broker.id=0
2  port=9092
3  log.dirs=./logs/kafka-0
```

server-1.properties

```
1  broker.id=1
2  port=9093
3  log.dirs=./logs/kafka-1
```

server-2.properties

```
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

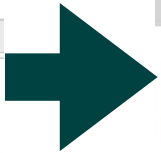
```
#!/usr/bin/env bash
CONFIG=`pwd`/config

cd ~/kafka-training

## Run Kafka
kafka/bin/kafka-server-start.sh \
    "$CONFIG/server-0.properties"
```

start-2nd-server.sh

```
#!/usr/bin/env bash
CONFIG=`pwd`/config
cd ~/kafka-training

## Run Kafka
kafka/bin/kafka-server-start.sh \
    "$CONFIG/server-1.properties"
```

start-3rd-server.sh

```
#!/usr/bin/env bash
CONFIG=`pwd`/config
cd ~/kafka-training

## Run Kafka
kafka/bin/kafka-server-start.sh \
    "$CONFIG/server-2.properties"
```



start-2nd-server.sh

```
#!/usr/bin/env bash
CONFIG=`pwd`/config
cd ~/kafka-training

## Run Kafka
kafka/bin/kafka-server-start.sh \
    "$CONFIG/server-1.properties"
```

- Passing properties files from last step

# Run Servers

Tos

```
$ ./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

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

cd ~/kafka-training

kafka/bin/kafka-topics.sh --create \ --bootstrap-server localhost:9092

    --replication-factor 3 \
    --partitions 13 \
    --topic my-failsafe-topic

```

- ***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 Kafka Consumer

Tos

start-consumer-console-replicated.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/kafka-console-consumer.sh \
    --bootstrap-server localhost:9094,localhost:9092 \
    --topic my-failsafe-topic \
    --from-beginning

```

- 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

```
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

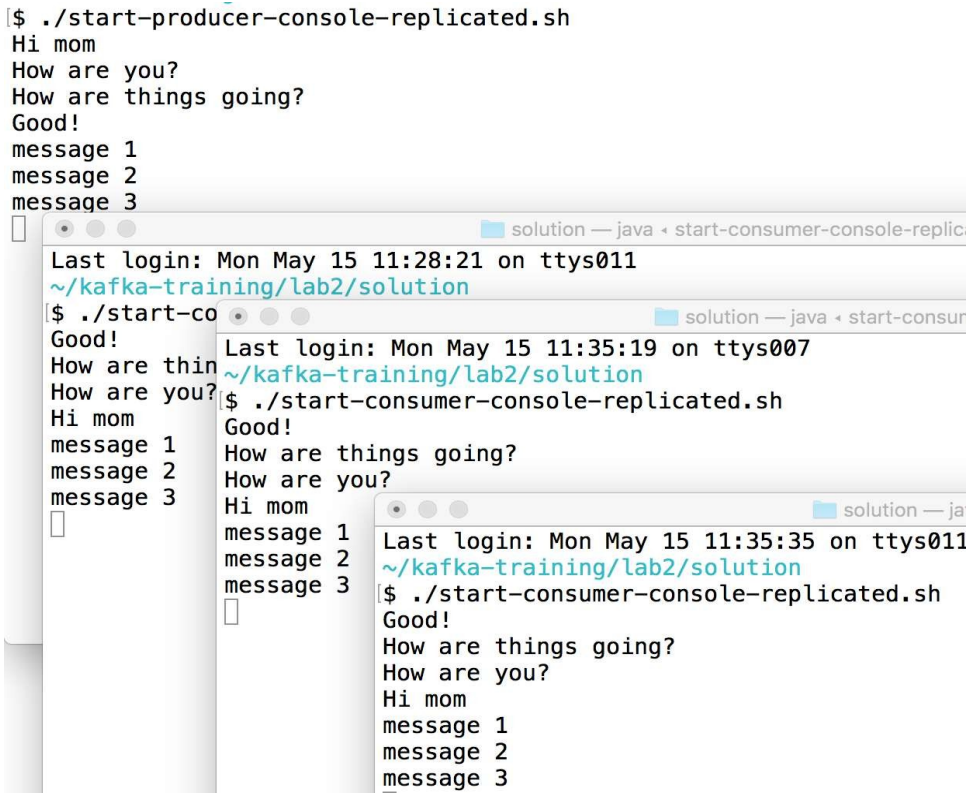
```
Last login: Mon May 15 11:25:19 on ttys007
~/kafka-training/lab2/solution
$ ./start-producer-console-replicated.sh
Hi mom
How are you?
How are things going?
Good!
```

solution — java ◂ sta

```
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

Tos




- 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

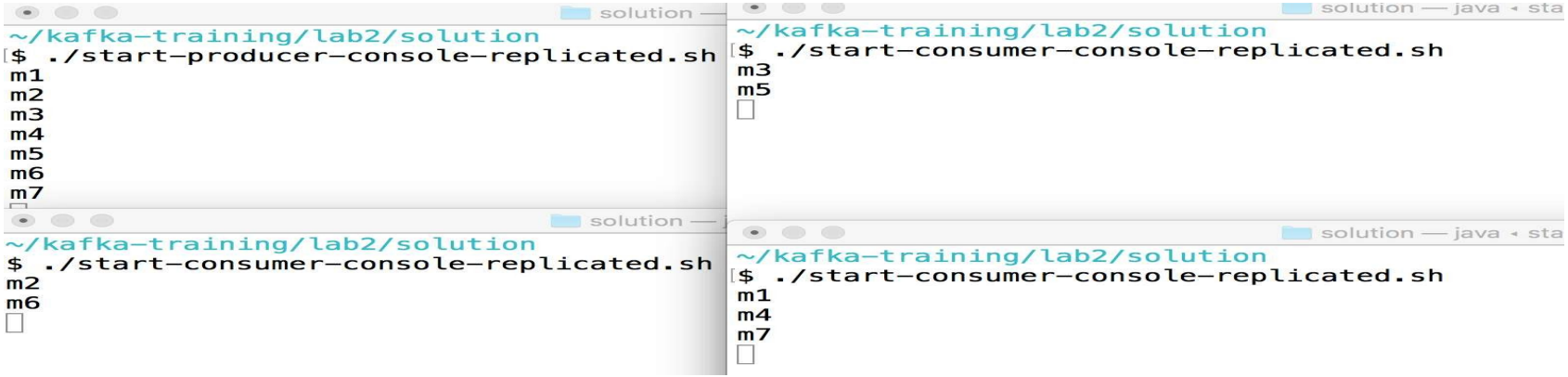
start-consumer-console-replicated.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/kafka-console-consumer.sh \
    --bootstrap-server localhost:9094,localhost:9092 \
    --topic my-failsafe-topic \
    --consumer-property group.id=mygroup
    --from-beginning

```

- Modify start consumer script
- Add the consumers to a group called mygroup
- Now they will share load

# Start up three consumers again

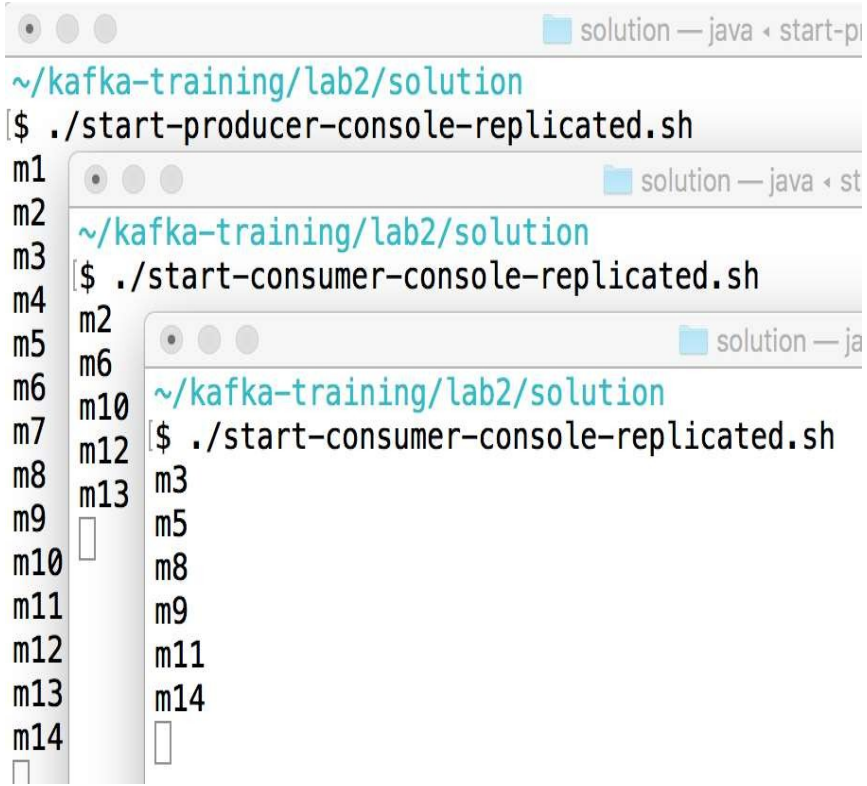
Tos




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

# Consumer Failover

Tos




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

# Create Kafka Describe Topic

Tos

describe-topics.sh

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

cd ~/kafka-training

# List existing topics
kafka/bin/kafka-topics.sh --describe \
    --topic my-failsafe-topic \
        --bootstrap-server localhost:9092
```

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

# Use Describe Topics

Tos

```
$ ./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

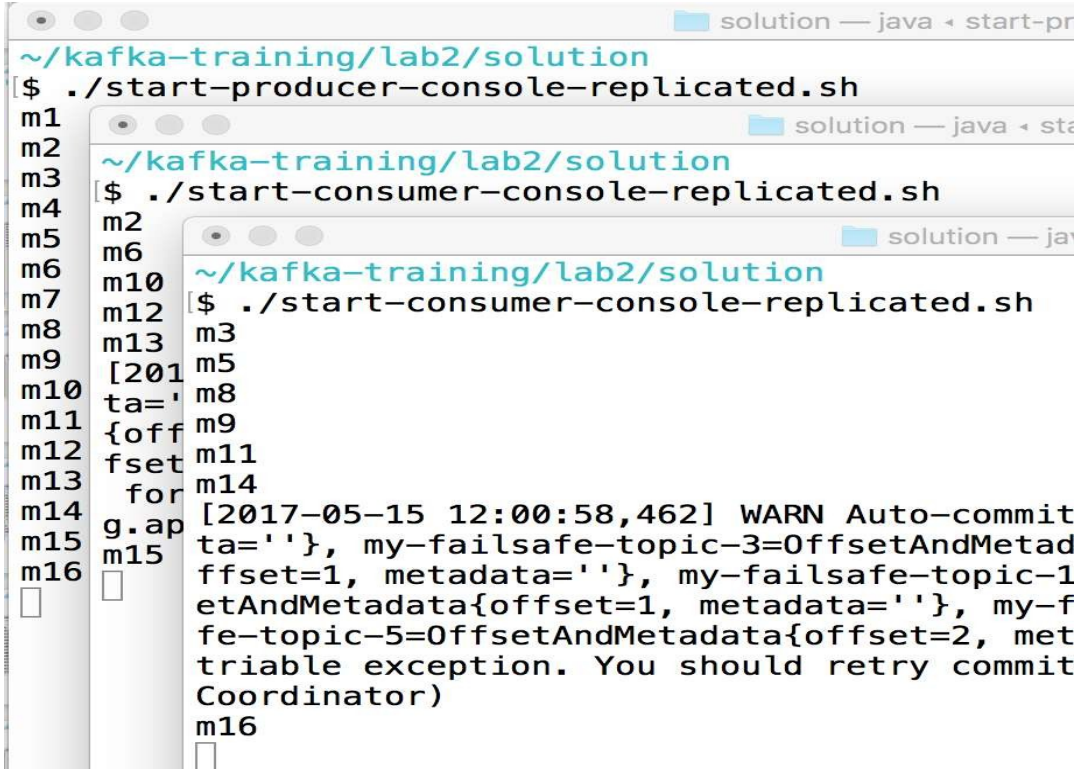
```
~/kafka-training/lab2/solution
$ kill `ps aux | grep java | grep server-0.properties | tr -s " " | cut -d " " -f2`
```

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




- Send two more messages from the producer
- Notice that the consumergets the messages
- Broker Failover WORKS

# Kafka Cluster Review

Tos

- 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