

Kafka MirrorMaker

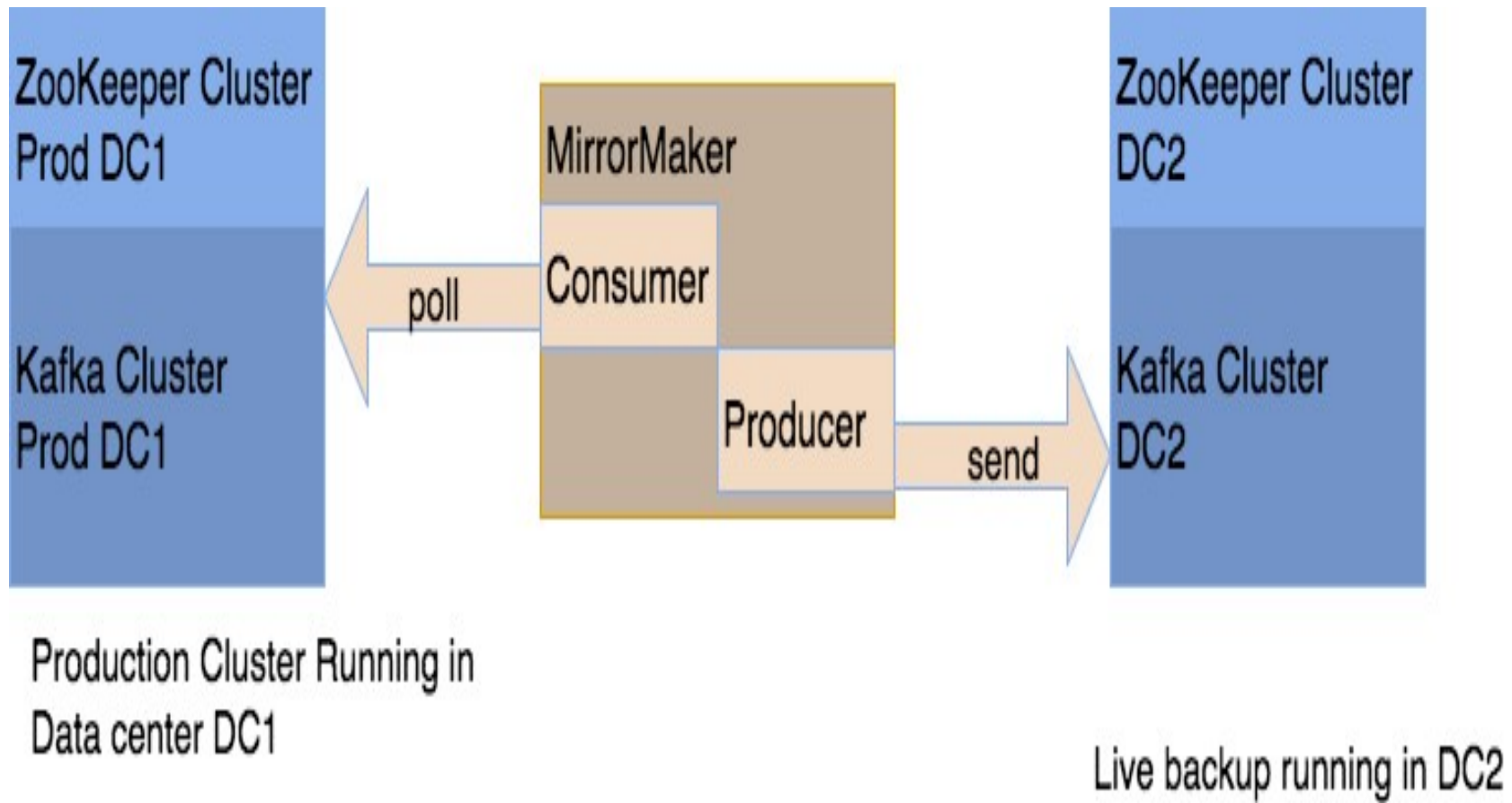
- ❖ Mirroring is replication between clusters – called ***mirroring*** to not confuse with ***replication***
- ❖ replication uses cluster involving brokers, partition leaders, partition followers, ISR and ZooKeeper
- ❖ mirroring is just a consumer/producer pair in two clusters
- ❖ ***MirrorMaker*** is used for replicating one clusters data to another cluster

- ❖ ***MirrorMaker*** acts like a ***consumer*** to a ***source cluster***
- ❖ ***MirrorMaker*** acts like a ***producer*** to a ***destination cluster***
- ❖ ***Data read from source topics in source cluster and written to same named topics in destination cluster***
- ❖ Source and destination clusters are independent and not coupled
 - ❖ Topics can be configured differently, have different offsets
 - ❖ e.g., different partition count and different replication factors

- ❖ Provide a replica to another datacenter or AWS region
- ❖ ***Mirroring*** used for ***disaster recovery***
 - ❖ datacenter or region goes down
 - ❖ cluster is used for normal fault-tolerance
- ❖ ***Mirroring*** can also be used for ***increased throughput***
 - ❖ scale consumers
 - ❖ scales reads

Disaster Recovery

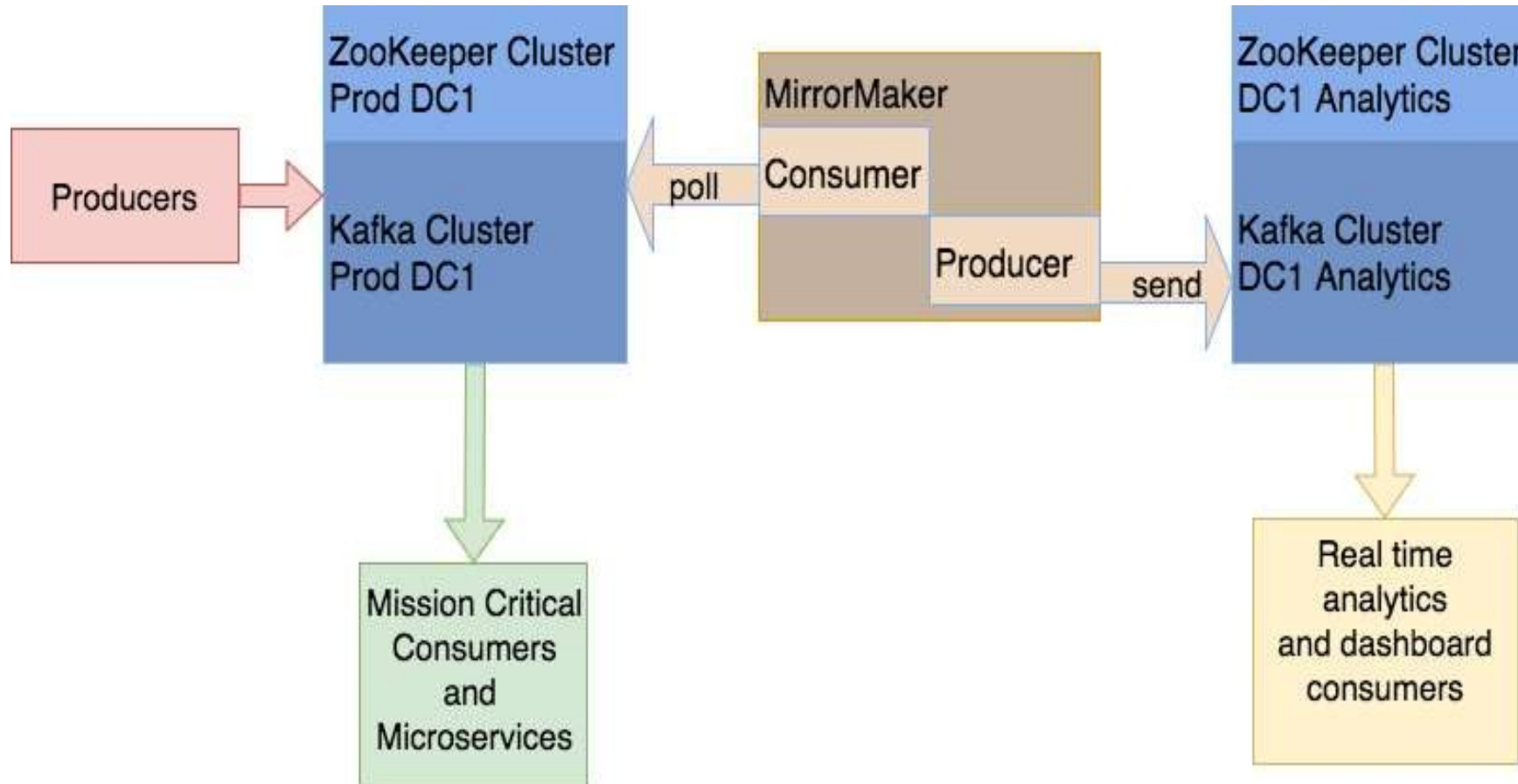
Tos



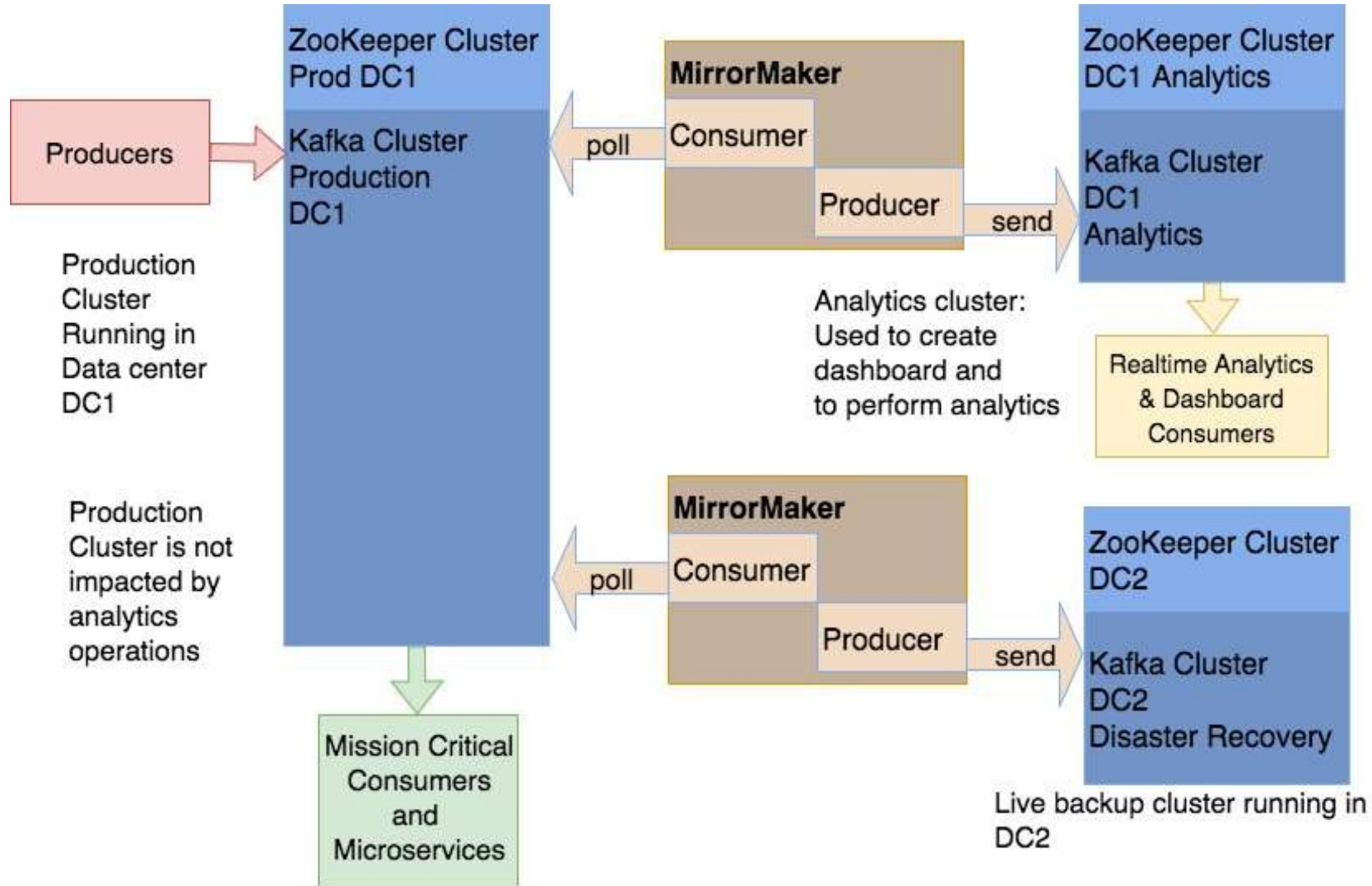
- ❖ You can use MirrorMaker to scale reads
- ❖ You could move non-mission critical consumers to another cluster and replicate to this other cluster
- ❖ Other cluster can replay log or do read intensive log operations and analytics **w/o impacting Production**
- ❖ Production cluster **to serve mission critical services**
- ❖ Analytics cluster could be doing real time dash boards and analytics

Scale Write, Avoid Impacting Mission Critical Services

Tos



Many MirrorMakers for different Purposes



- ❖ ***kafka-mirror-maker.sh***
- ❖ ***--whitelist*** specifies regex for topics to mirror
 - ❖ 'stock-prices|stocks' selects two topics
 - ❖ '*' selects all topics
- ❖ ***--blacklist*** —whitelist regex for topics to exclude
- ❖ Using mirroring with broker config ***auto.create.topics.enable=true*** on destination cluster makes auto replication with no config possible (***--whitelist "*"***)

Mirror Maker Command Line

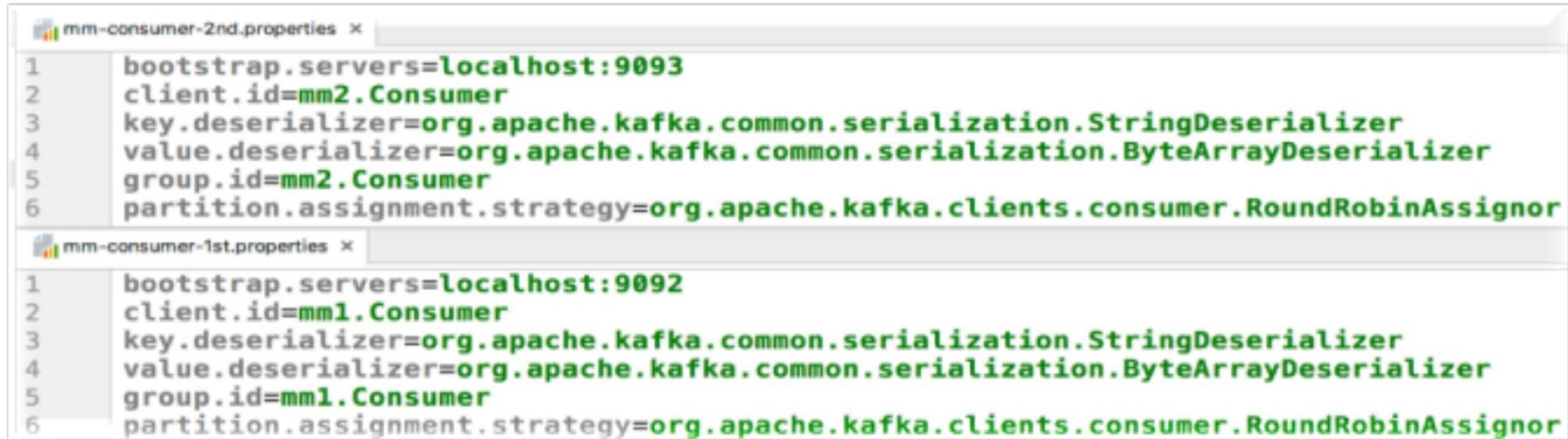
Tos

```
start-mirror-maker-1st-to-2nd.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4
5  ## Run Kafka Mirror Maker
6  kafka/bin/kafka-mirror-maker.sh \
7      --consumer.config "$CONFIG/mm-consumer-1st.properties" \
8      --producer.config "$CONFIG/mm-producer-2nd.properties" \
9      --whitelist ".*"
10
```

- ❖ Pass consumer properties to read from 1st cluster
- ❖ Pass producer properties to write to 2nd cluster
- ❖ Specify that you want to replicate all topics via whitelist regex

- ❖ What is the difference between failover and disaster recovery?
- ❖ What are two use cases where you would use MirrorMaker?
- ❖ Why might you want to separate a production microservice messages from a more ad hoc analytics system?
- ❖ If you had to run a nightly job that tallied analytics to all of the calls to a 24/7 production microservice for the last month would you run that in the production

Mirror Maker Consumer Config



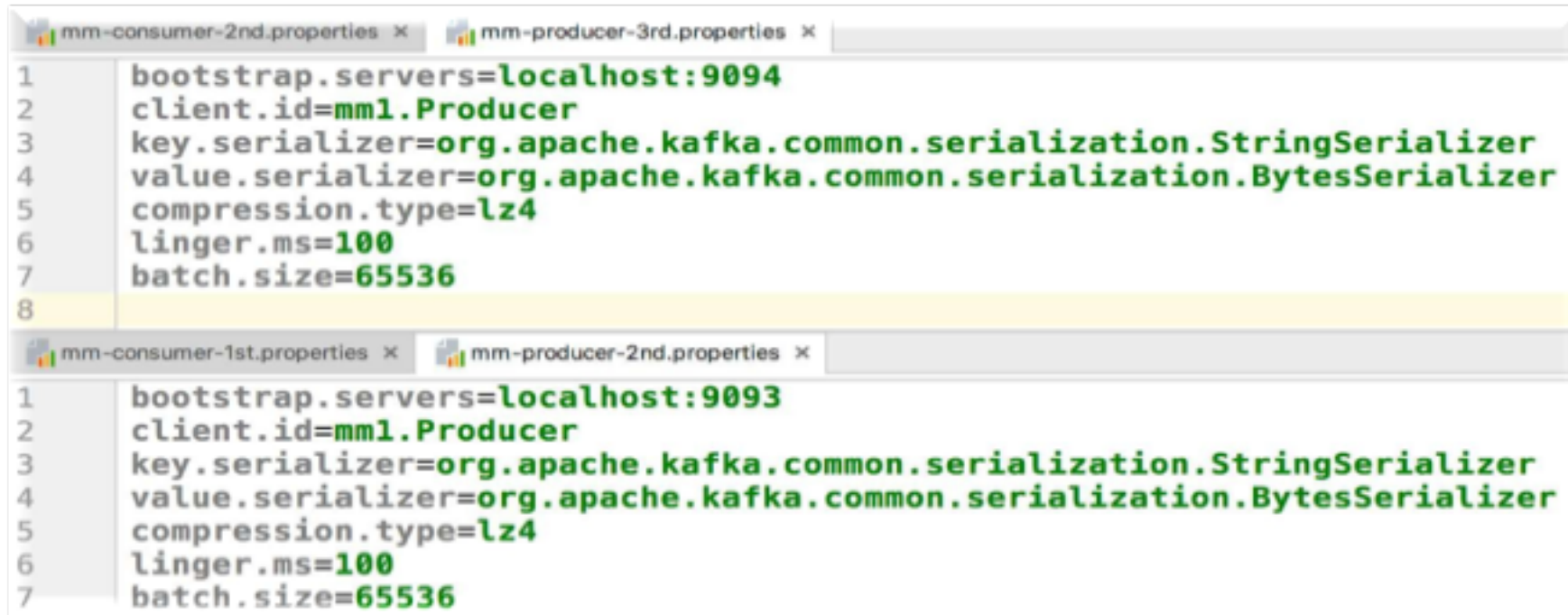
The image shows a screenshot of two text editors side-by-side, displaying Kafka consumer configuration properties for Mirror Maker. The top editor, titled 'mm-consumer-2nd.properties', contains six lines of configuration for a consumer that connects to localhost:9093. The bottom editor, titled 'mm-consumer-1st.properties', contains six lines of configuration for a consumer that connects to localhost:9092. Both configurations specify the same serializers, group ID, and partition assignment strategy.

```
mm-consumer-2nd.properties x
1 bootstrap.servers=localhost:9093
2 client.id=mm2.Consumer
3 key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
4 value.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer
5 group.id=mm2.Consumer
6 partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor

mm-consumer-1st.properties x
1 bootstrap.servers=localhost:9092
2 client.id=mm1.Consumer
3 key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
4 value.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer
5 group.id=mm1.Consumer
6 partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor
```

- ❖ Two Consumer for 2 different MirrorMakers
- ❖ One consumes 2nd Cluster (9093)
- ❖ One consumes 1st Cluster (9092)
- ❖ Notice we use ByteArrayDeserializer because we want MirrorMaker treating payload as opaque

Mirror Maker Producer Config



The image shows a screenshot of two text editors side-by-side. The top editor is titled 'mm-consumer-2nd.properties' and 'mm-producer-3rd.properties'. It contains the following configuration:

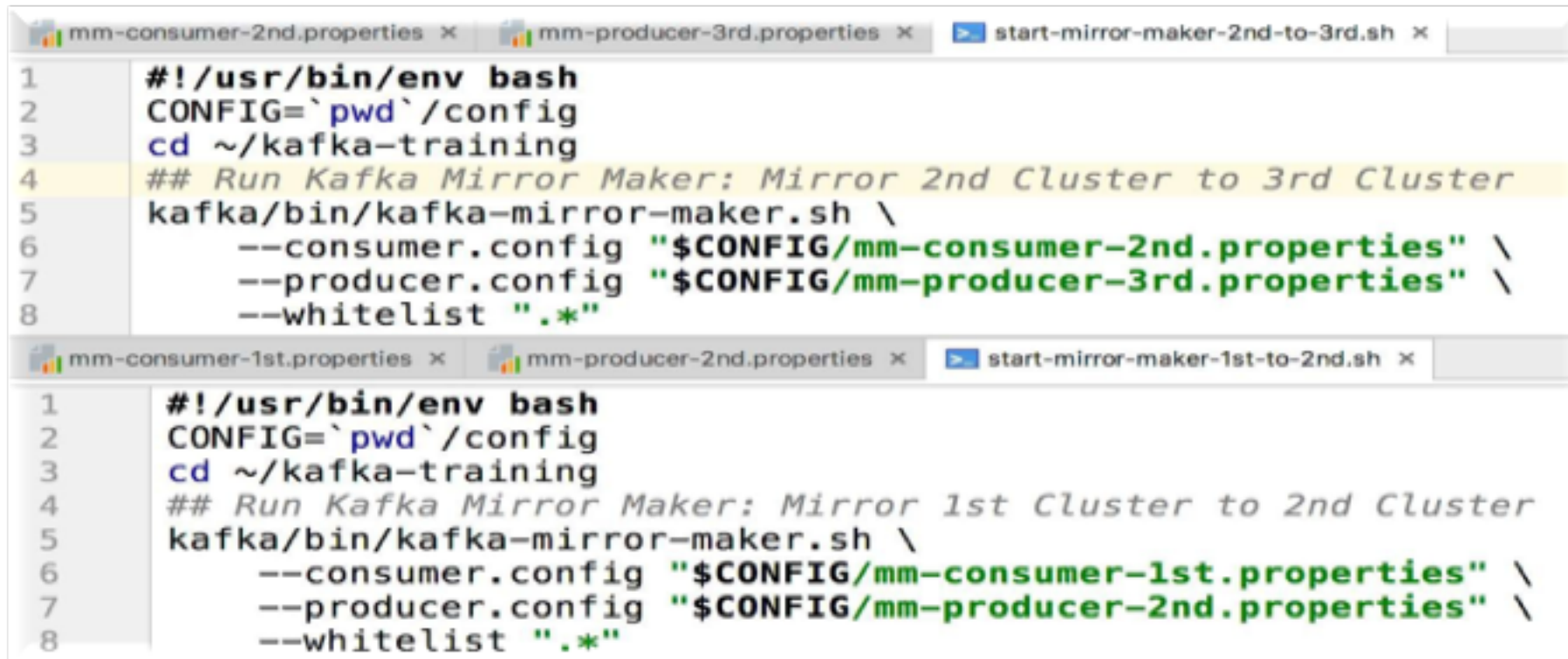
```
1 bootstrap.servers=localhost:9094
2 client.id=mm1.Producer
3 key.serializer=org.apache.kafka.common.serialization.StringSerializer
4 value.serializer=org.apache.kafka.common.serialization.BytesSerializer
5 compression.type=lz4
6 linger.ms=100
7 batch.size=65536
8
```

The bottom editor is titled 'mm-consumer-1st.properties' and 'mm-producer-2nd.properties'. It contains the following configuration:

```
1 bootstrap.servers=localhost:9093
2 client.id=mm1.Producer
3 key.serializer=org.apache.kafka.common.serialization.StringSerializer
4 value.serializer=org.apache.kafka.common.serialization.BytesSerializer
5 compression.type=lz4
6 linger.ms=100
7 batch.size=65536
```

- ❖ Two Consumer for 2 different MirrorMakers
- ❖ One produces to 3rd Cluster
- ❖ One produces to 2nd Cluster
- ❖ Notice we use BytesSerializer because we want MirrorMaker treating payload as opaque

Mirror Maker Start Scripts



The image shows two terminal windows side-by-side. The top window is titled 'start-mirror-maker-2nd-to-3rd.sh' and contains a script to start Mirror Maker from the 2nd cluster to the 3rd cluster. The bottom window is titled 'start-mirror-maker-1st-to-2nd.sh' and contains a script to start Mirror Maker from the 1st cluster to the 2nd cluster. Both scripts use the 'kafka-mirror-maker.sh' command with specific consumer and producer configuration files and a whitelist.

```
mm-consumer-2nd.properties x mm-producer-3rd.properties x start-mirror-maker-2nd-to-3rd.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4  ## Run Kafka Mirror Maker: Mirror 2nd Cluster to 3rd Cluster
5  kafka/bin/kafka-mirror-maker.sh \
6      --consumer.config "$CONFIG/mm-consumer-2nd.properties" \
7      --producer.config "$CONFIG/mm-producer-3rd.properties" \
8      --whitelist ".*"

mm-consumer-1st.properties x mm-producer-2nd.properties x start-mirror-maker-1st-to-2nd.sh x
1  #!/usr/bin/env bash
2  CONFIG=`pwd`/config
3  cd ~/kafka-training
4  ## Run Kafka Mirror Maker: Mirror 1st Cluster to 2nd Cluster
5  kafka/bin/kafka-mirror-maker.sh \
6      --consumer.config "$CONFIG/mm-consumer-1st.properties" \
7      --producer.config "$CONFIG/mm-producer-2nd.properties" \
8      --whitelist ".*"
```

- ❖ Mirror 2nd Cluster to 3rd using Producer and Consumer config
- ❖ Mirror 1st Cluster to 2nd using Producer and Consumer config

Lab: Mirroring data between clusters – MirrorMaker