[redacted]

Kafka MirrorMaker

MirrorMaker and Mirroring

Tos

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

Mirror Maker

Tos

- ***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

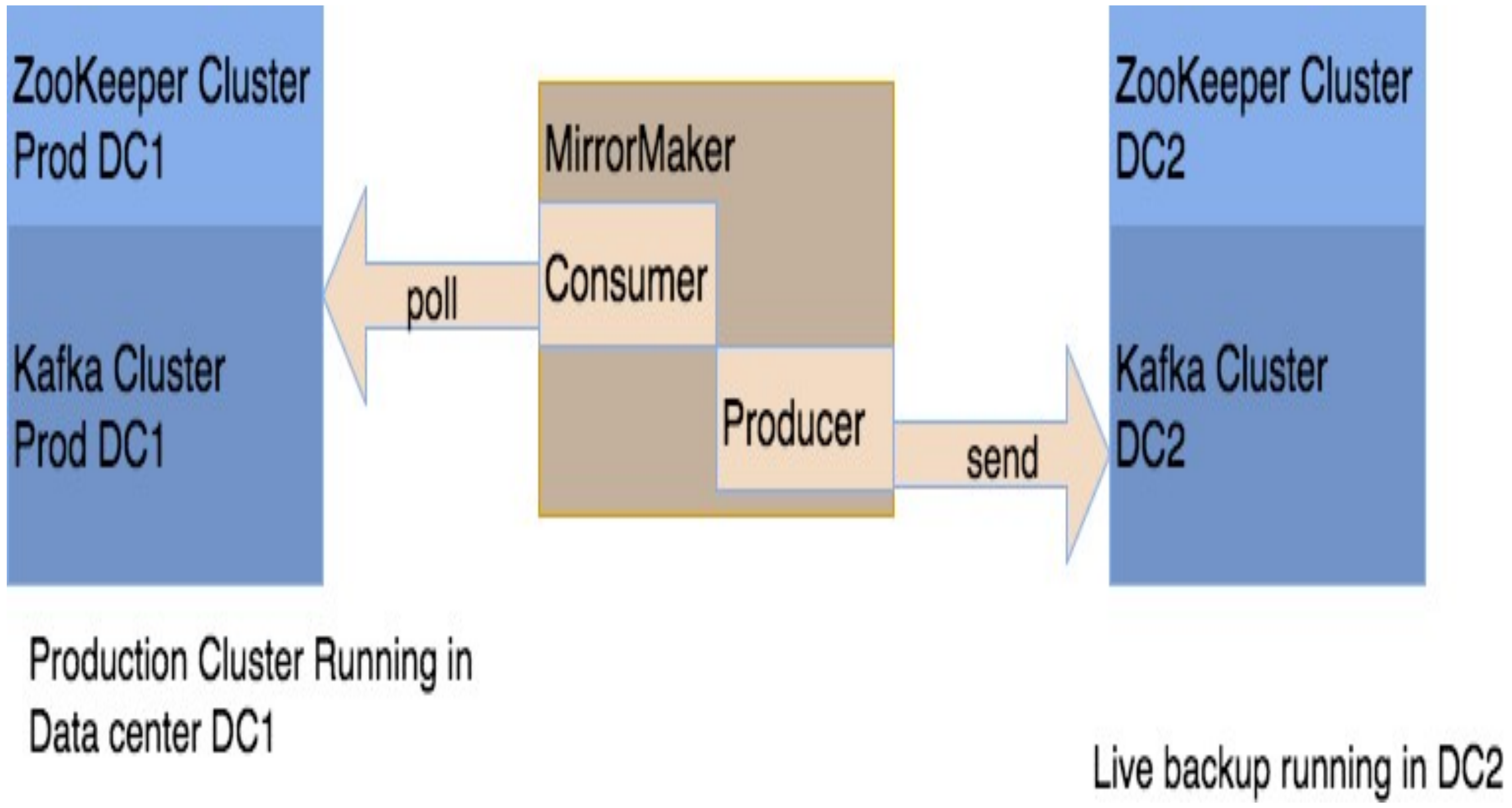
MirrorMaker Use Cases

Tos

- 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



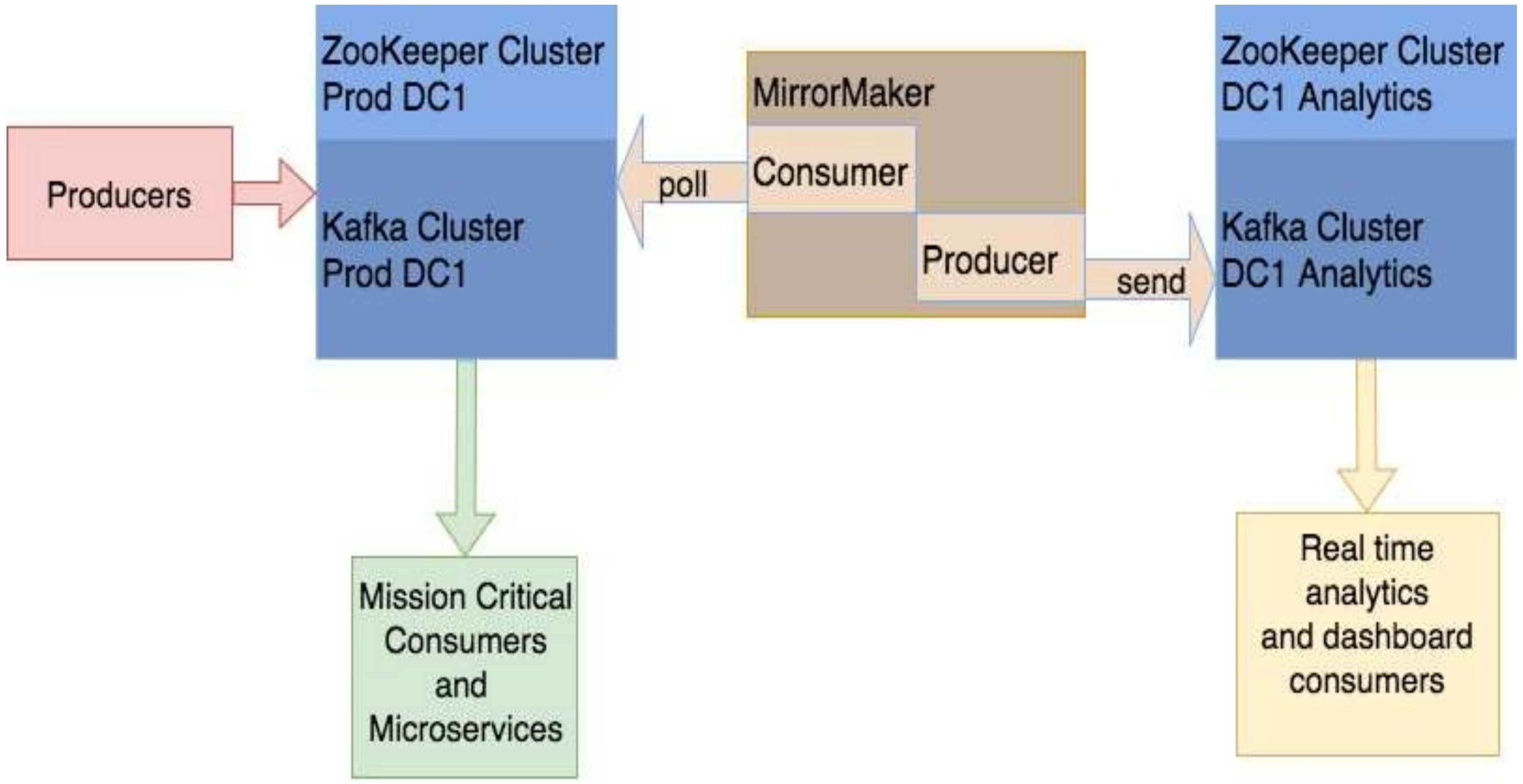
Scale Reads / Scale Consumers

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

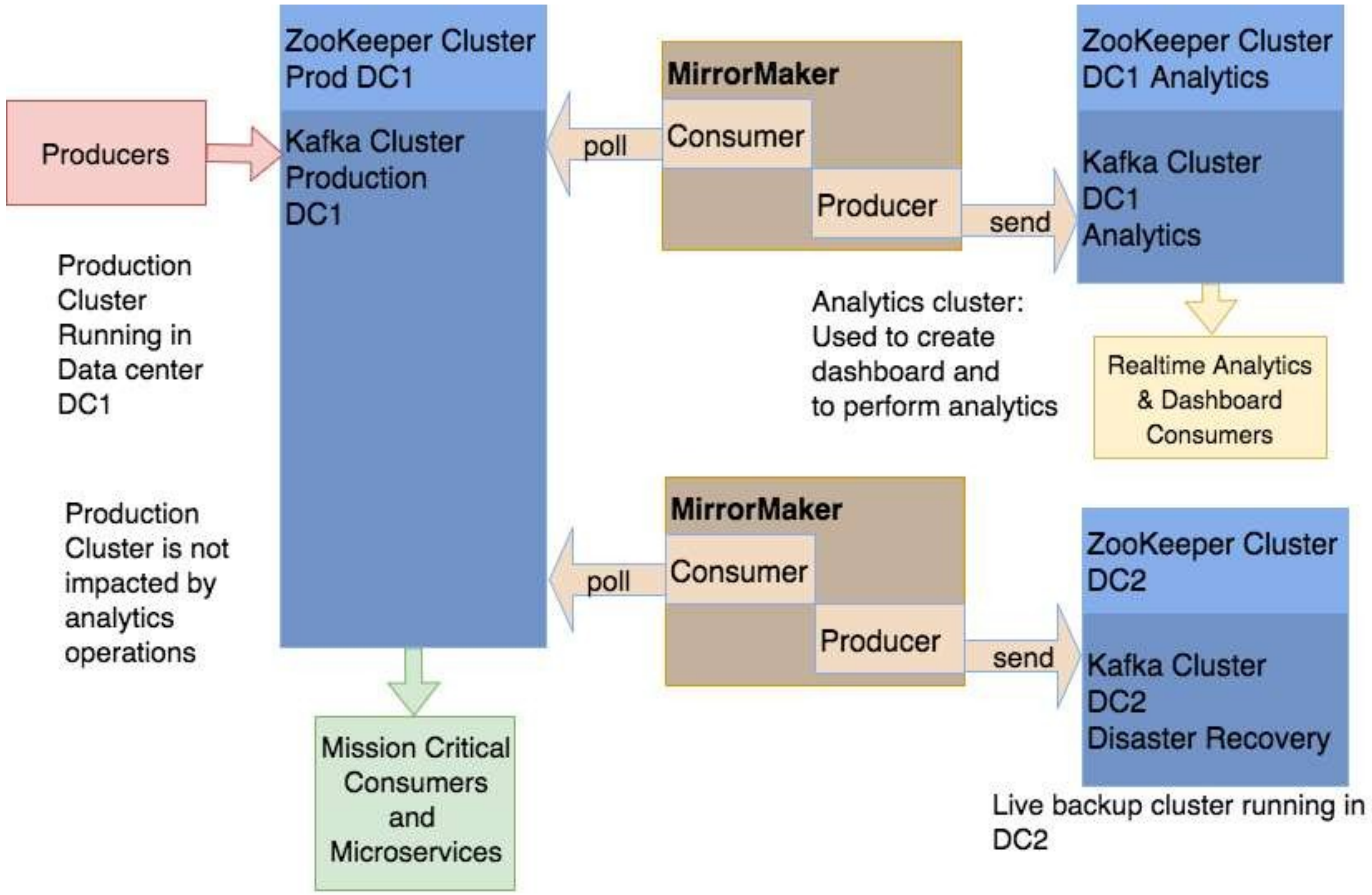
Scale Write, Avoid Impacting Mission Critical Services

Tos


Producers
ZooKeeper Cluster Prod DC1
Kafka Cluster Prod DC1
poll
MirrorMaker
Consumer
Producer
send
ZooKeeper Cluster DC1 Analytics
Kafka Cluster DC1 Analytics
Mission Critical Consumers and Microservices
Real time analytics and dashboard consumers

Many MirrorMakers for different Purposes

Tos



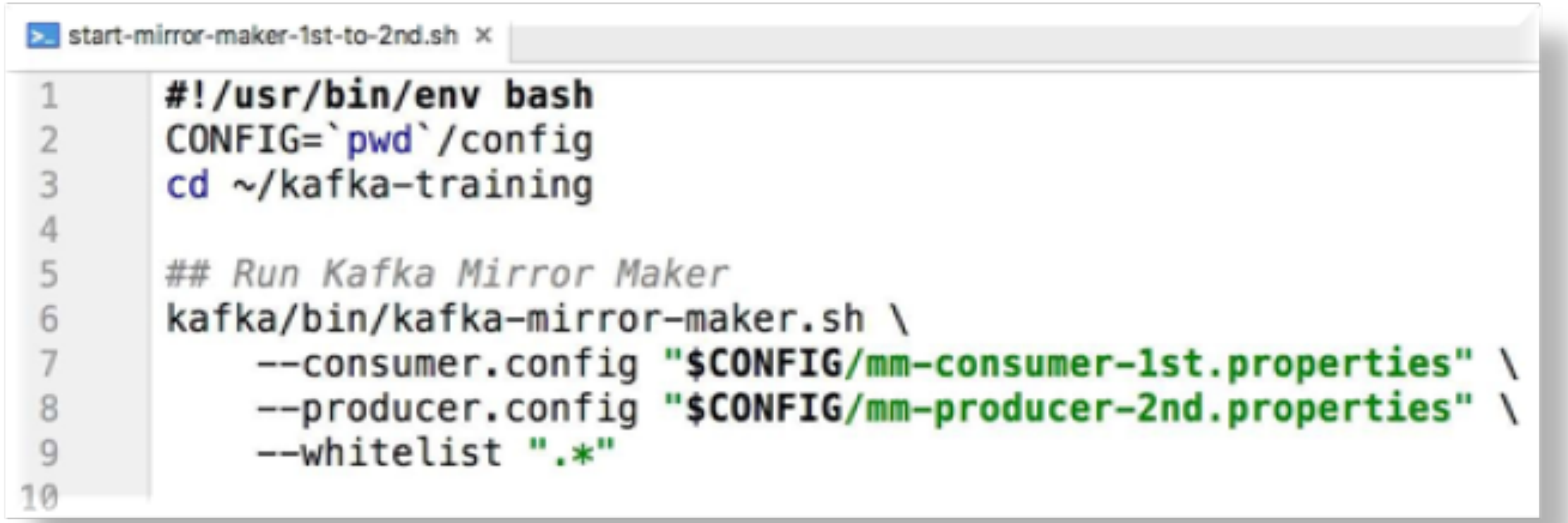
Producers
ZooKeeper Cluster Prod DC1
Kafka Cluster Production DC1
poll
MirrorMaker
Consumer
Producer
send
ZooKeeper Cluster DC1 Analytics
Kafka Cluster DC1 Analytics
Production Cluster Running in Data center DC1
Analytics cluster: Used to create dashboard and to perform analytics
Realtime Analytics & Dashboard Consumers
Production Cluster is not impacted by analytics operations
poll
MirrorMaker
Consumer
Producer
send
ZooKeeper Cluster DC2
Kafka Cluster DC2 Disaster Recovery
Live backup cluster running in DC2
Mission Critical Consumers and Microservices

Mirror Maker Command Line

Tos

- ***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



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

## Run Kafka Mirror Maker
kafka/bin/kafka-mirror-maker.sh \
    --consumer.config "$CONFIG/mm-consumer-1st.properties" \
    --producer.config "$CONFIG/mm-producer-2nd.properties" \
    --whitelist ".*"
```

- 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

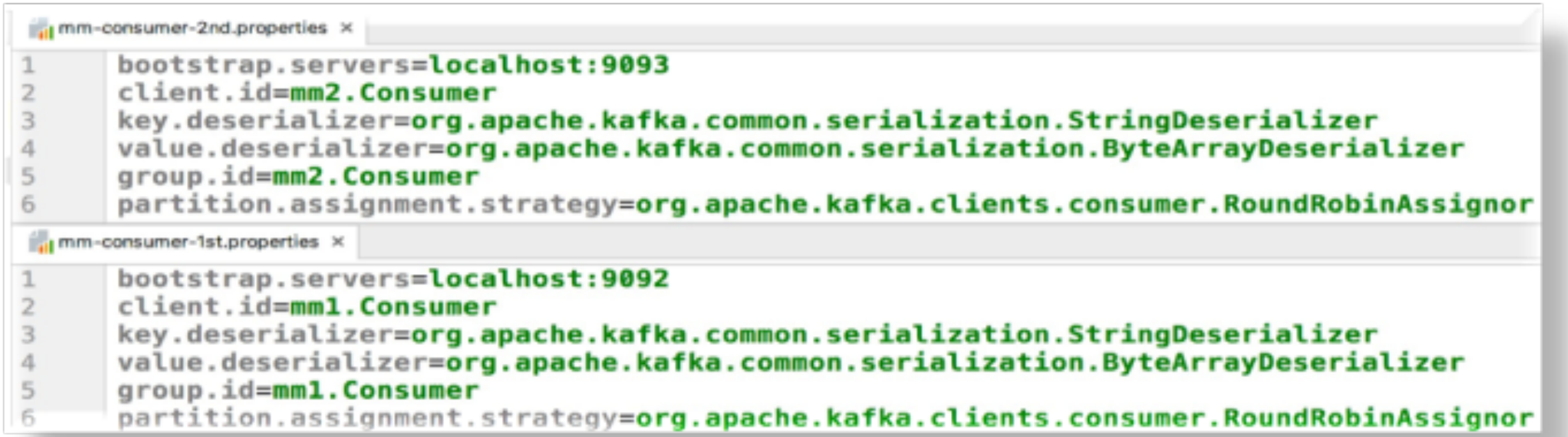
MirrorMaker Review

Tos

- 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

Tos



mm-consumer-2nd.properties

```
bootstrap.servers=localhost:9093
client.id=mm2.Consumer
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer
group.id=mm2.Consumer
partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor
```

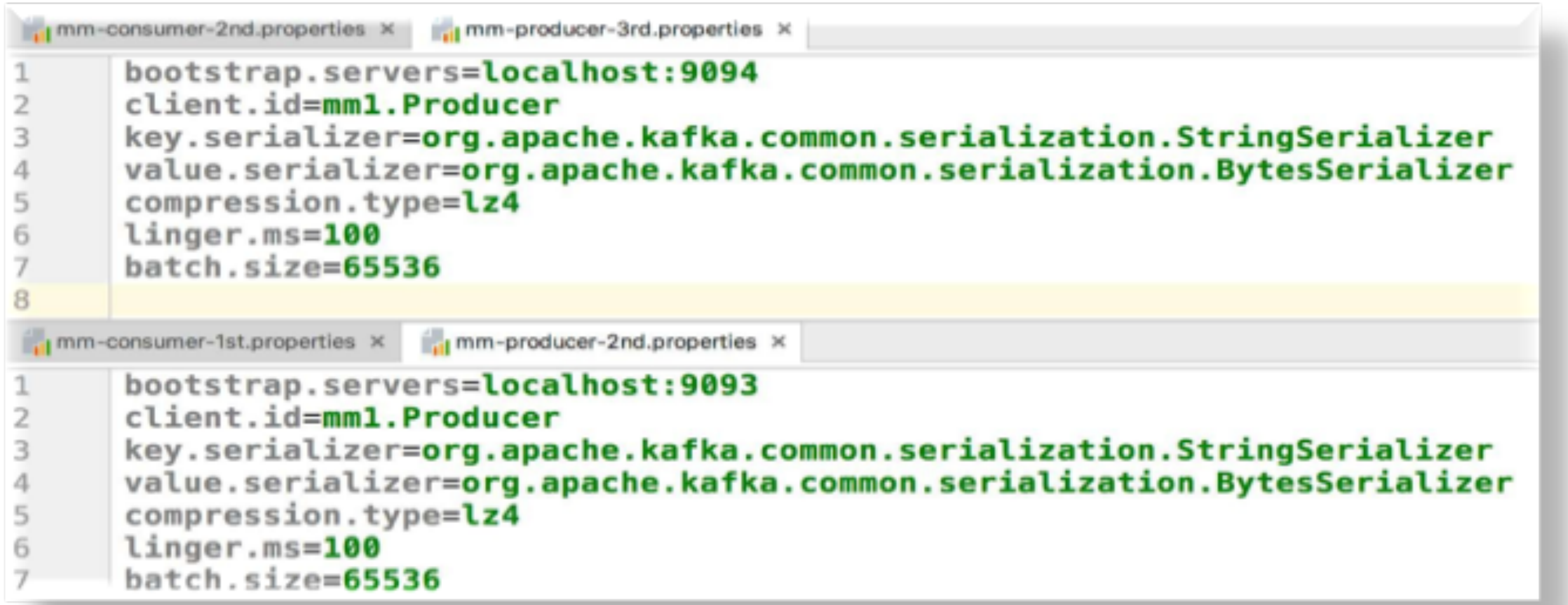
mm-consumer-1st.properties

```
bootstrap.servers=localhost:9092
client.id=mm1.Consumer
key.deserializer=org.apache.kafka.common.serialization.StringDeserializer
value.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer
group.id=mm1.Consumer
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

Tos


mm-consumer-2nd.properties
mm-producer-3rd.properties

```
bootstrap.servers=localhost:9094
client.id=mm1.Producer
key.serializer=org.apache.kafka.common.serialization.StringSerializer
value.serializer=org.apache.kafka.common.serialization.BytesSerializer
compression.type=lz4
linger.ms=100
batch.size=65536

```

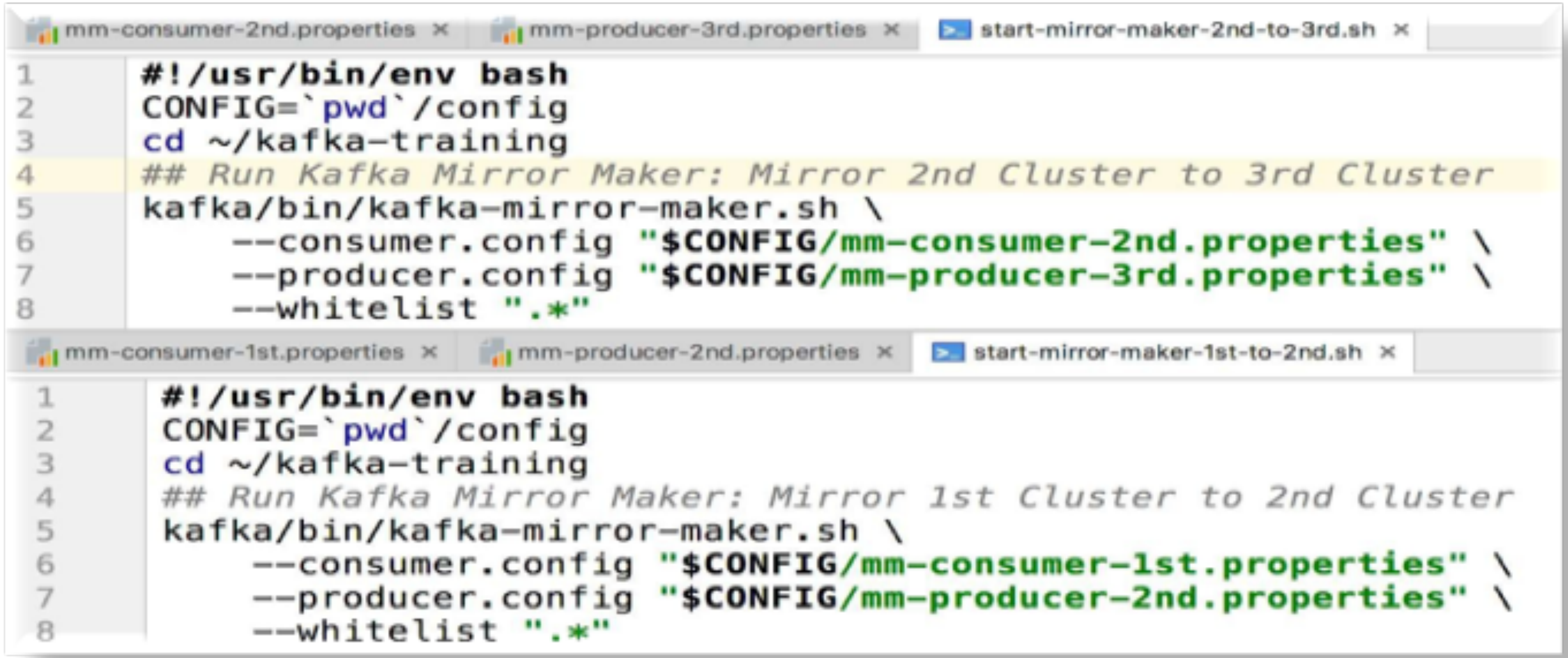
mm-consumer-1st.properties
mm-producer-2nd.properties

```
bootstrap.servers=localhost:9093
client.id=mm1.Producer
key.serializer=org.apache.kafka.common.serialization.StringSerializer
value.serializer=org.apache.kafka.common.serialization.BytesSerializer
compression.type=lz4
linger.ms=100
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

Tos


mm-consumer-2nd.properties
mm-producer-3rd.properties
start-mirror-maker-2nd-to-3rd.sh

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

mm-consumer-1st.properties
mm-producer-2nd.properties
start-mirror-maker-1st-to-2nd.sh

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

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

Tos

Lab: Mirroring data between clusters – MirrorMaker