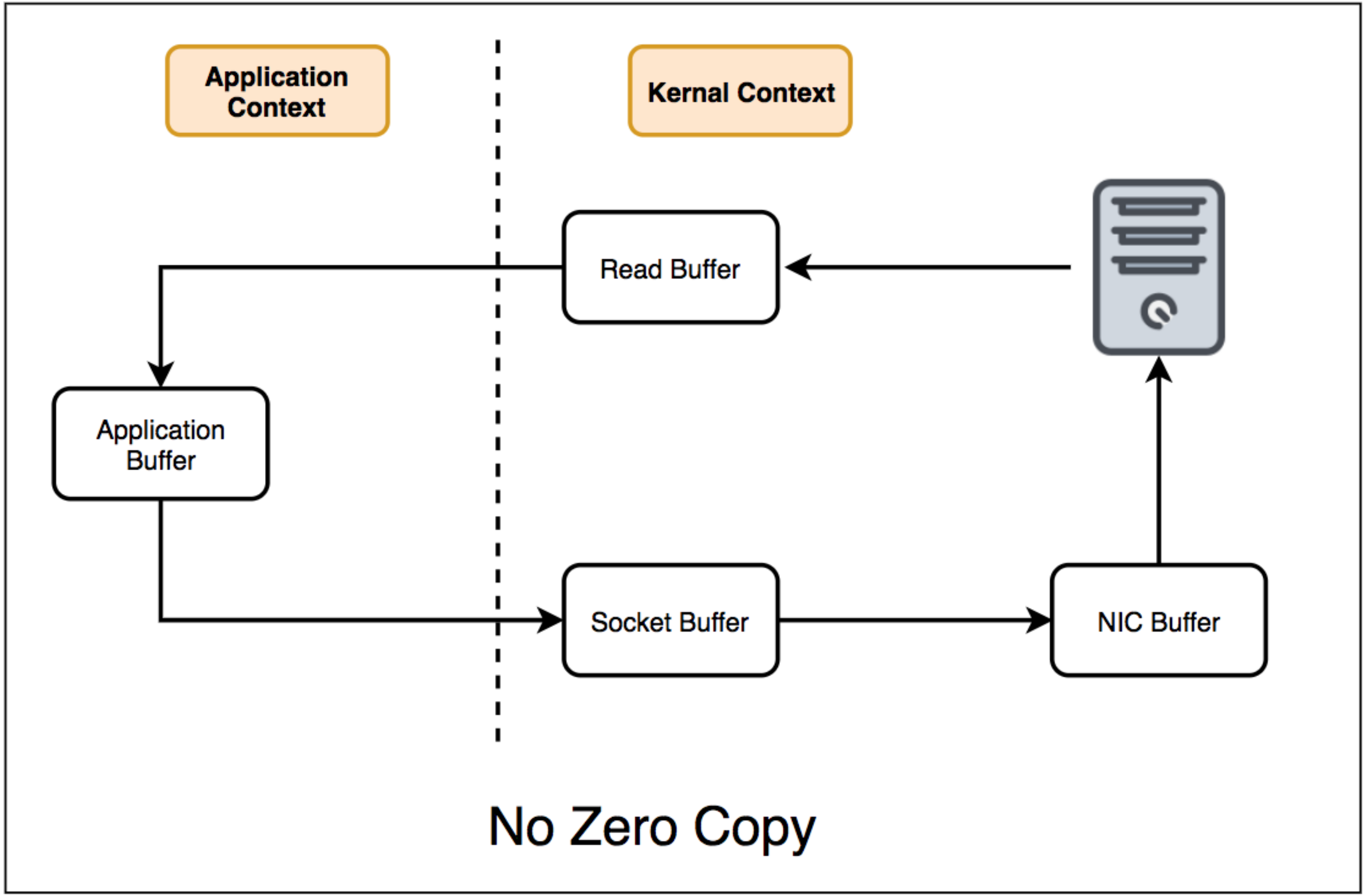
Tos

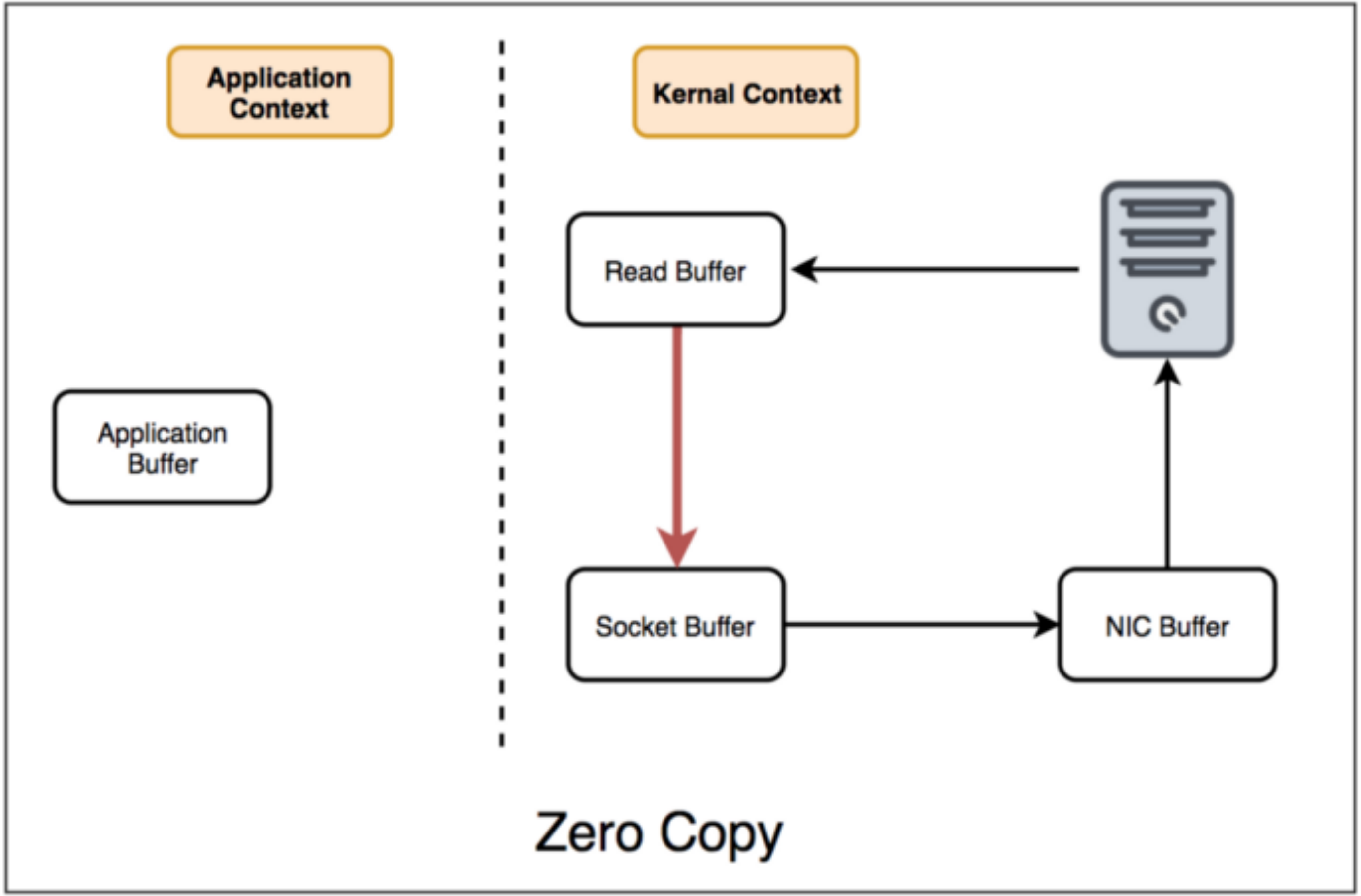
Kafka Topics Architecture

No Zero Copy


Application Context
Kernal Context
Read Buffer
Application Buffer
Socket Buffer
NIC Buffer
No Zero Copy

Zero Copy

Tos



Kafka versus MOM

Kafka vs JMS vs RabbitMQ Messaging

- Is Kafka a Queue or a Pub/Sub/Topic?
  - Yes
- Kafka is like a Queue per consumer group
  - Kafka is a queue system per consumer in consumer group so load balancing like JMS, RabbitMQ queue
- Kafka is like Topics in JMS, RabbitMQ, MOM
  - Topic/pub/sub by offering Consumer Groups which act like subscriptions
  - Broadcast to multiple consumer groups
- MOM = JMS, ActiveMQ, RabbitMQ, IBM MQ Series, Tibco, etc.

Kafka vs MOM

- By design, Kafka is better suited for scale than traditional MOM systems due to partition topic log
    - Load divided among Consumers for read by partition
    - Handles parallel consumers better than traditional MOM
- Also by moving location (partition offset) in log to client/consumer side of equation instead of the broker, less tracking required by Broker and more flexible consumers
- Kafka written with mechanical sympathy, modern hardware, cloud in mind
    - Disks are faster
    - Servers have tons of system memory
    - Easier to spin up servers for scale out

Topics, Logs, Partitions

Tos

- Kafka ***Topic*** is a stream of records
- ***Topics*** stored in log
- ***Log*** broken up into ***partitions*** and ***segments***
- ***Topic*** is a category or stream name or feed
- Topics are pub/sub
  - Can have zero or many subscribers - consumer groups
- ***Topics*** are broken up and spread by partitions for speed and size

Topic Partitions

- ***Topics*** are broken up into ***partitions***
- ***Partitions*** decided usually by key of record
  - Key of record determines which partition
- ***Partitions*** are used to scale Kafka across many servers
  - Record sent to correct partition by key
- ***Partitions*** are used to facilitate parallel consumers
  - Records are consumed in parallel up to the number of partitions
- Order guaranteed per partition
- Partitions can be ***replicated*** to multiple brokers

Topic Partition Log

Tos

- ***Order*** is maintained only in a single ***partition***
  - ***Partition*** is ordered, immutable sequence of records that is continually appended to—a structured commit ***log***
- ***Records*** in partitions are assigned ***sequential id*** number called the *offset*
- ***Offset*** identifies each record within the partition
- ***Topic Partitions*** allow Kafka log to scale beyond a size that will fit on a single server
  - Topic partition must fit on servers that host it
  - topic can span many partitions hosted on many servers

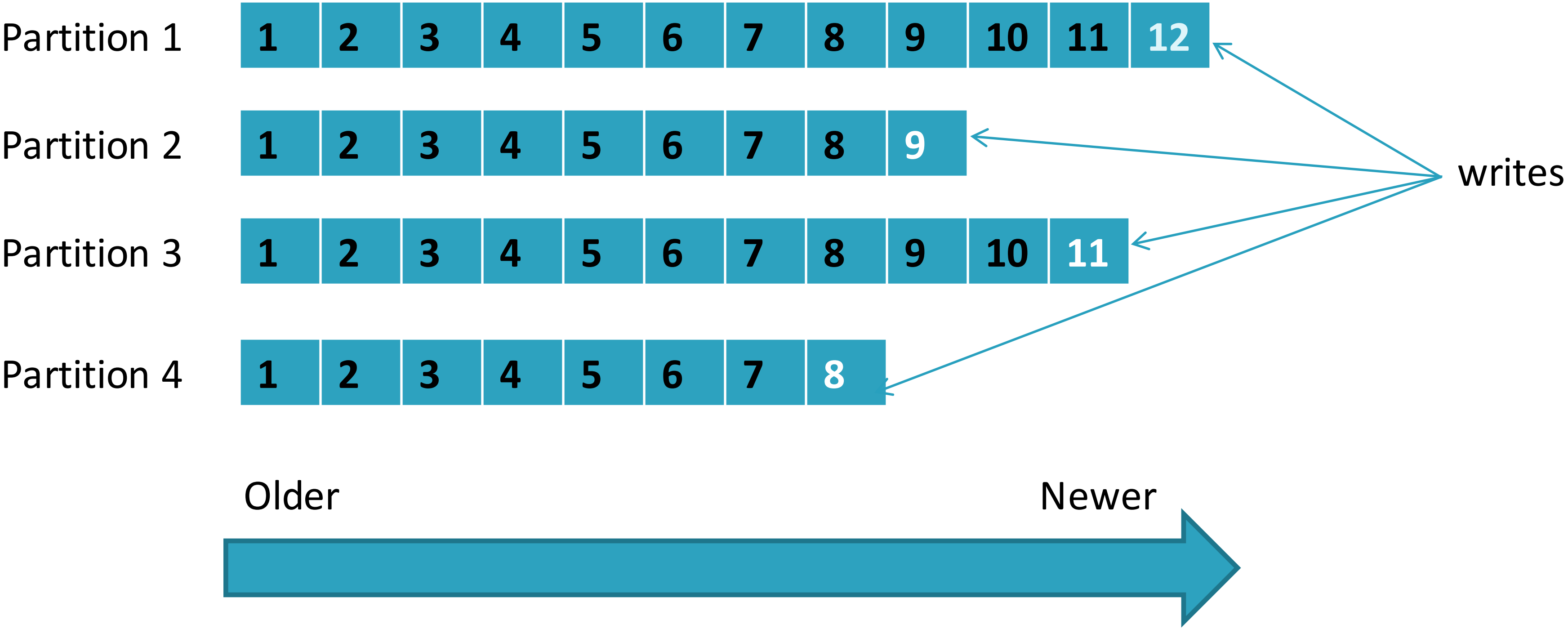
Topic Parallelism and consumers

Tos

- Topic Partitions are unit of ***parallelism***
  partition can only be used by one consumer in group at a time
- Consumers can run in their own process or their own thread
- If a consumer stops, Kafka spreads partitions across remaining consumer in group
- #of Consumers you can run per Consumer Group limited by #of Partitions
- Consumers getting assigned partition aids in efficient message consumption tracking

Kafka Topic Partitions Layout

Tos



Partition 1	1	2	3	4	5	6	7	8	9	10	11	12
Partition 2	1	2	3	4	5	6	7	8	9			
Partition 3	1	2	3	4	5	6	7	8	9	10	11	
Partition 4	1	2	3	4	5	6	7	8				

writes

Older

Newer

Kafka Scale and Speed

- How can Kafka scale if multiple producers and consumers read/write to same Kafka Topic log?
- Writes fast: Sequential writes to file system are ***fast*** (700 MB or more a second)
- Scales writes and reads by ***sharding:***
    - Topic logs into ***Partitions*** (parts of a Topic log)
    - Topics logs can be split into multiple Partitions ***different machines/different disks***
    - Multiple Producers can write to different Partitions of the same Topic
    - Multiple Consumers Groups can read from different partitions

Replication: Kafka Partition Distribution

Tos

- Each partition has ***leader server*** and zero or more ***follower servers***
  - ***Leader*** handles all read and write requests for partition
  - ***Followers*** replicate leader, and take over if leader dies
  - Used for parallel consumer handling within a group
- Partitions of log are distributed over the servers in the Kafka cluster with each server handling data and requests for a share of partitions
- Each partition can be replicated across a configurable number of Kafka servers - Used for fault tolerance

Replication: Kafka Partition leader

Tos

- One node/partition's replicas is chosen as ***leader***

- Leader handles all reads and writes of Records for partition

- Writes to partition are ***replicated*** to ***followers*** (node/partition pair)

- An ***follower*** that is ***in-sync*** is called an ***ISR (in-sync replica)***

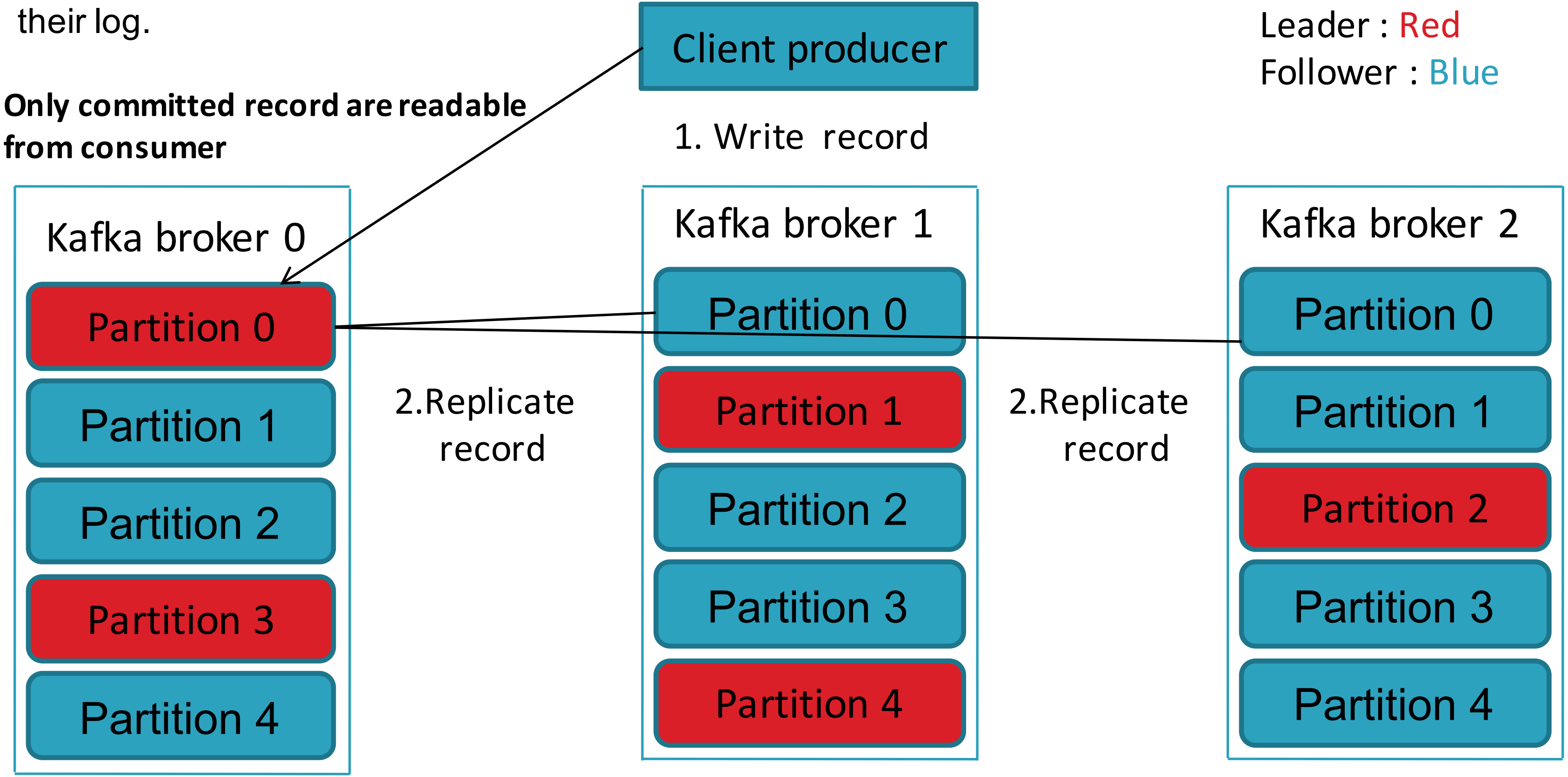
- If a partition leader fails, one ISR is chosen as new leader

Kafka Replication to Partition 0

Tos

Record is considered “committed”.
When all ISRs for partition wrote to
their log.

Only committed record are readable from consumer



Client producer

1. Write record

Leader : Red
Follower : Blue

Kafka broker 0
Partition 0
Partition 1
Partition 2
Partition 3
Partition 4

2.Replicate record

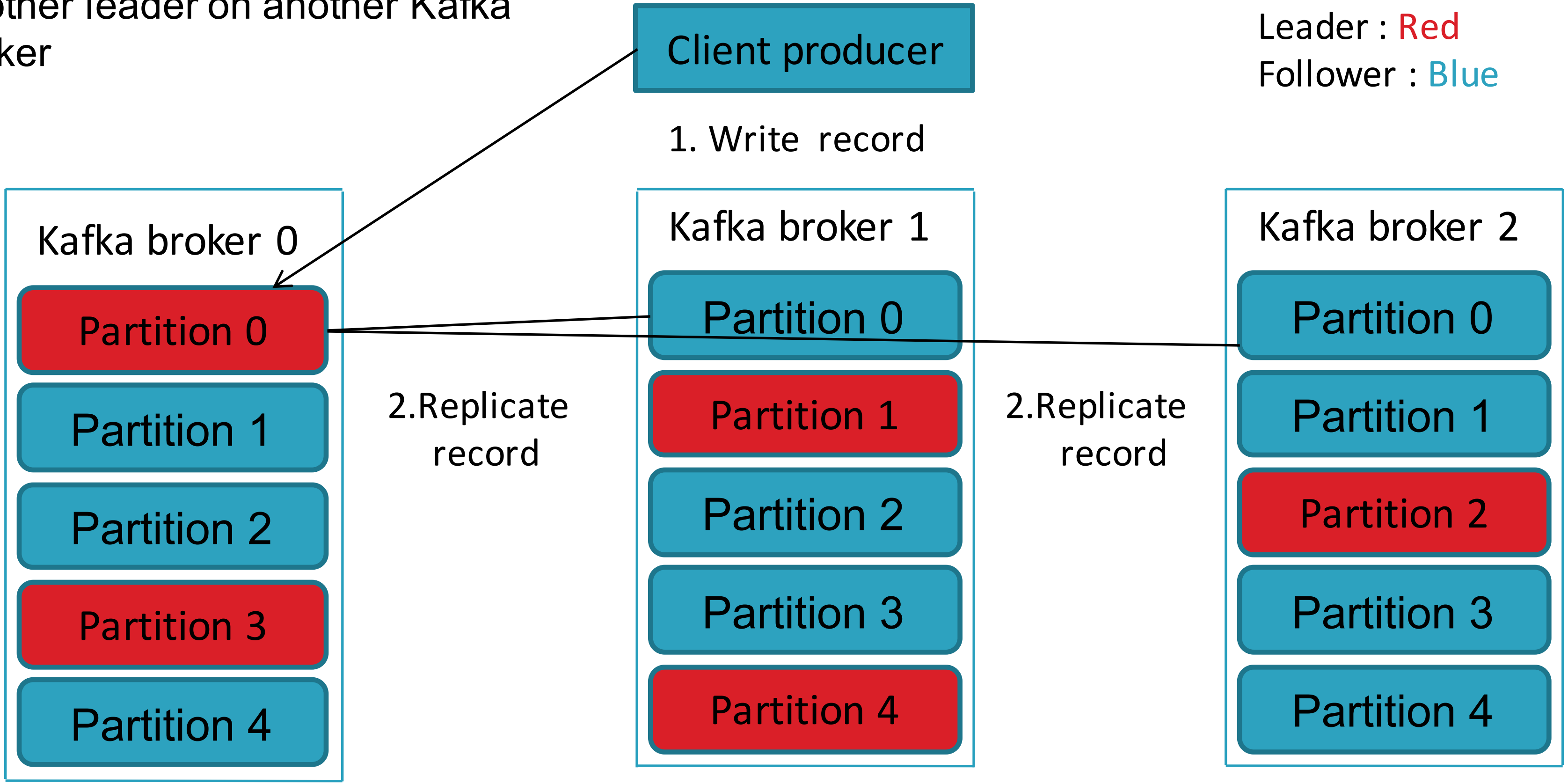
Kafka broker 1
Partition 0
Partition 1
Partition 2
Partition 3
Partition 4

2.Replicate record

Kafka broker 2
Partition 0
Partition 1
Partition 2
Partition 3
Partition 4

Kafka Replication to Partition 1

Another partition can be owned by another leader on another Kafka broker

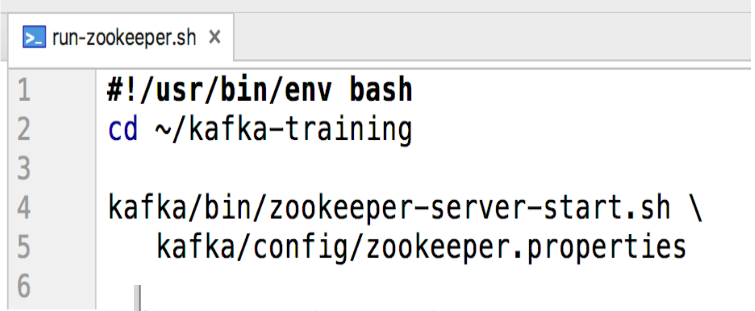


Leader : Red
Follower : Blue

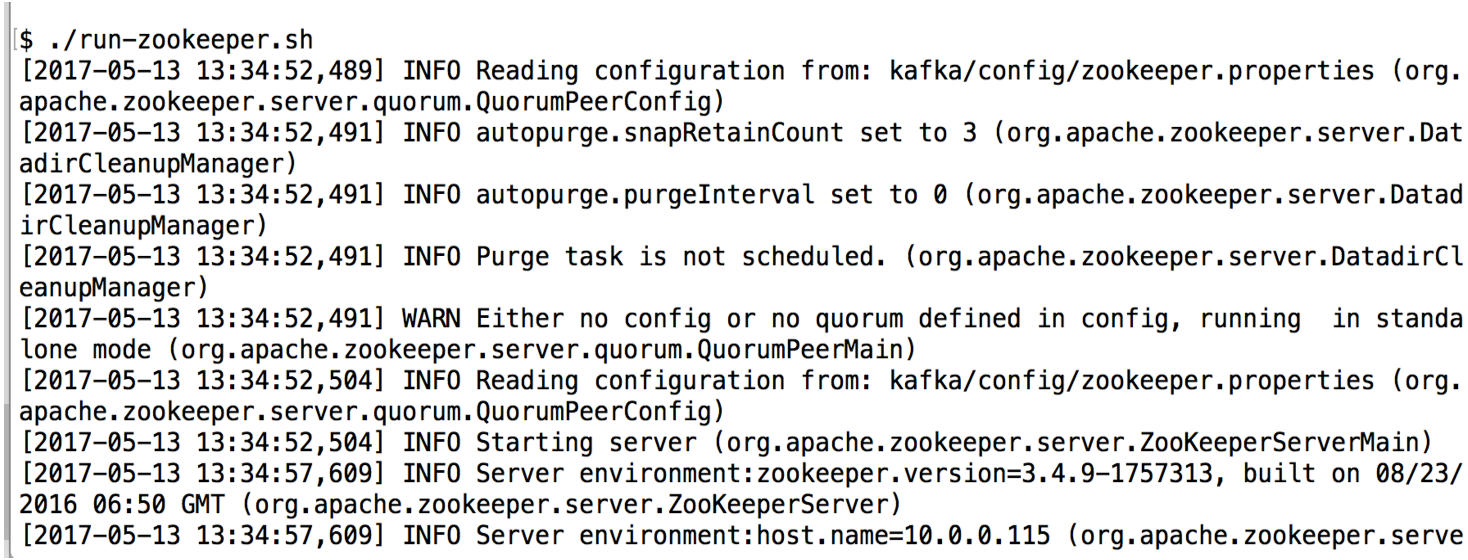
Run Kafka

- Run Zoo Keeper start up script
- Run Kafka Server/Broker start up script
- Create Kafka Topic from command line

Run ZooKeeper

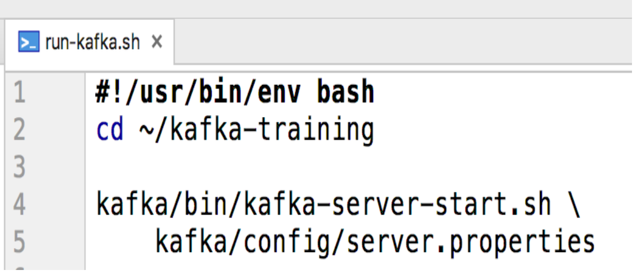


```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/zookeeper-server-start.sh \
   kafka/config/zookeeper.properties

```

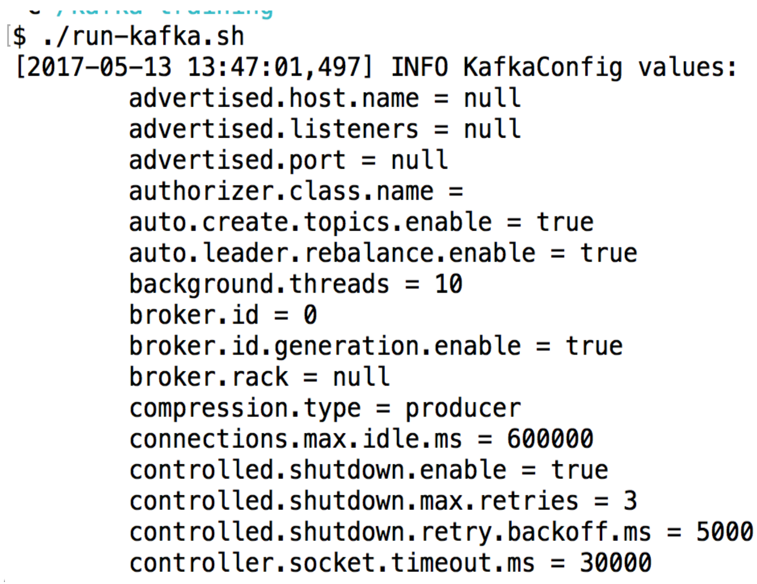


```
$ ./run-zookeeper.sh
[2017-05-13 13:34:52,489] INFO Reading configuration from: kafka/config/zookeeper.properties (org.
apache.zookeeper.server.quorum.QuorumPeerConfig)
[2017-05-13 13:34:52,491] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.Dat
adirCleanupManager)
[2017-05-13 13:34:52,491] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.Datad
irCleanupManager)
[2017-05-13 13:34:52,491] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCl
eanupManager)
[2017-05-13 13:34:52,491] WARN Either no config or no quorum defined in config, running  in standa
lone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2017-05-13 13:34:52,504] INFO Reading configuration from: kafka/config/zookeeper.properties (org.
apache.zookeeper.server.quorum.QuorumPeerConfig)
[2017-05-13 13:34:52,504] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2017-05-13 13:34:57,609] INFO Server environment:zookeeper.version=3.4.9-1757313, built on 08/23/
2016 06:50 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2017-05-13 13:34:57,609] INFO Server environment:host.name=10.0.0.115 (org.apache.zookeeper.serve
```

Run Kafka Server

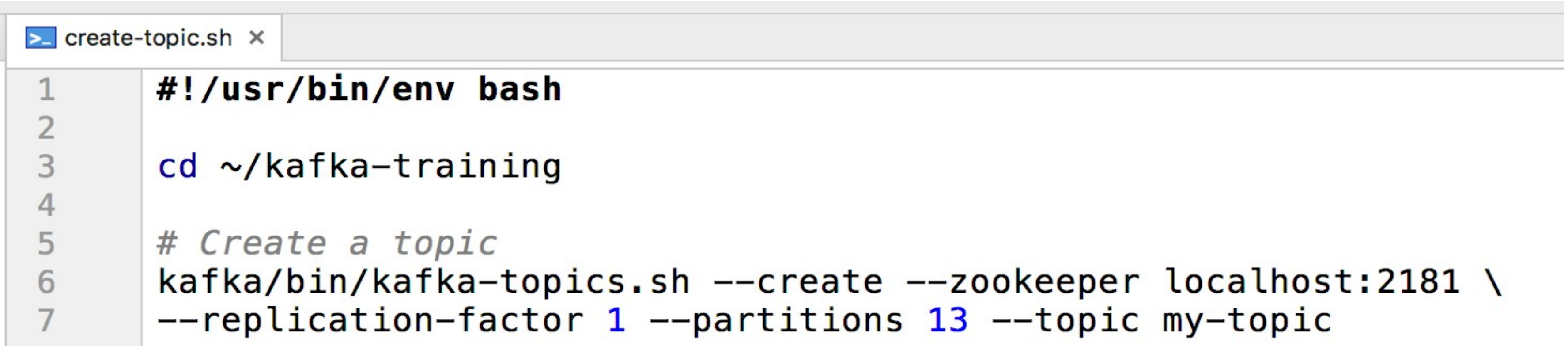


```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/kafka-server-start.sh \
    kafka/config/server.properties
```

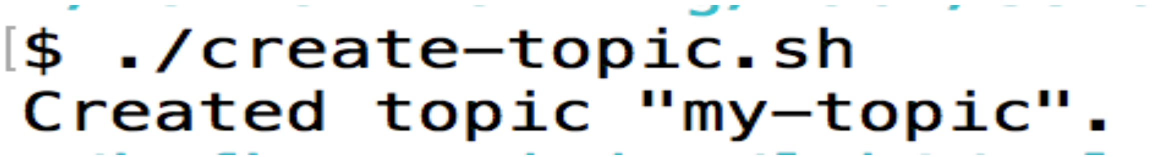


```
~ /kafka-training
[$ ./run-kafka.sh
[2017-05-13 13:47:01,497] INFO KafkaConfig values:
        advertised.host.name = null
        advertised.listeners = null
        advertised.port = null
        authorizer.class.name =
        auto.create.topics.enable = true
        auto.leader.rebalance.enable = true
        background.threads = 10
        broker.id = 0
        broker.id.generation.enable = true
        broker.rack = null
        compression.type = producer
        connections.max.idle.ms = 600000
        controlled.shutdown.enable = true
        controlled.shutdown.max.retries = 3
        controlled.shutdown.retry.backoff.ms = 5000
        controller.socket.timeout.ms = 30000
```

Create Kafka Topic

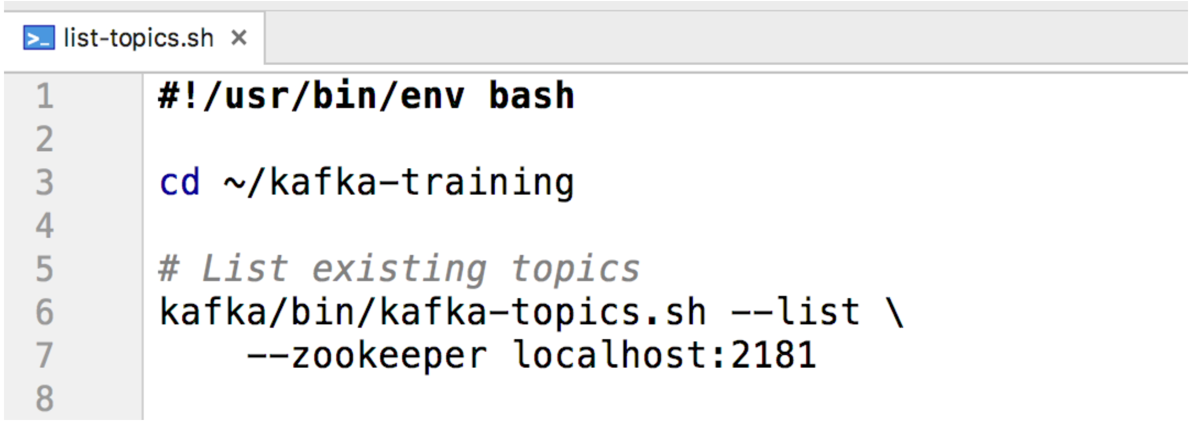


```
#!/usr/bin/env bash

cd ~/kafka-training

# Create a topic
kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 \
--replication-factor 1 --partitions 13 --topic my-topic
```

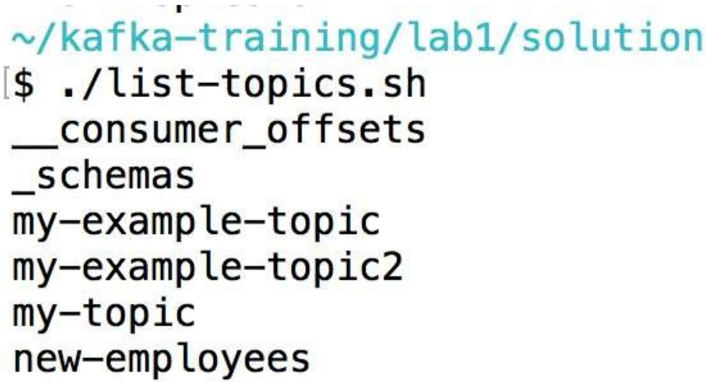


```
[$ ./create-topic.sh
 Created topic "my-topic".
```

List Topics



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

cd ~/kafka-training

# List existing topics
kafka/bin/kafka-topics.sh --list \
    --zookeeper localhost:2181

```



```
~/kafka-training/lab1/solution
$ ./list-topics.sh
__consumer_offsets
_schemas
my-example-topic
my-example-topic2
my-topic
new-employees
```

[redacted]

Lab - Basic Kafka Operations - CLI (Topic) – 30 Mins