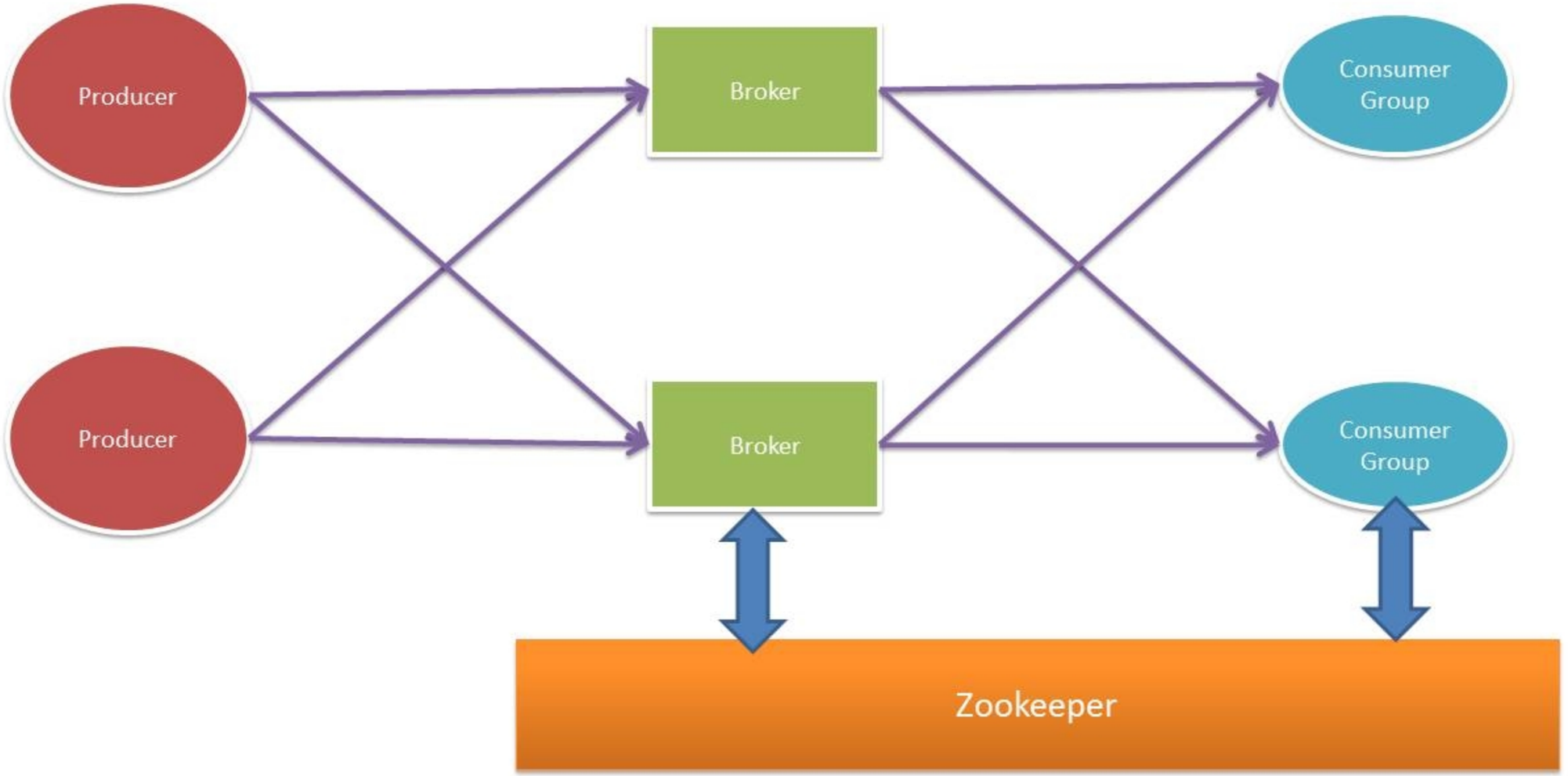
Welcome to the World of Distributed Messaging Queue

Apache Kafka

- Kafka is a distributed messaging queue
- Support replication
- Partitioned the data
- No single point of failure
- High/Read write performance
- Persist data into disk
- Support the data retention

Architecture – Kafka 0.8.X



Architecture

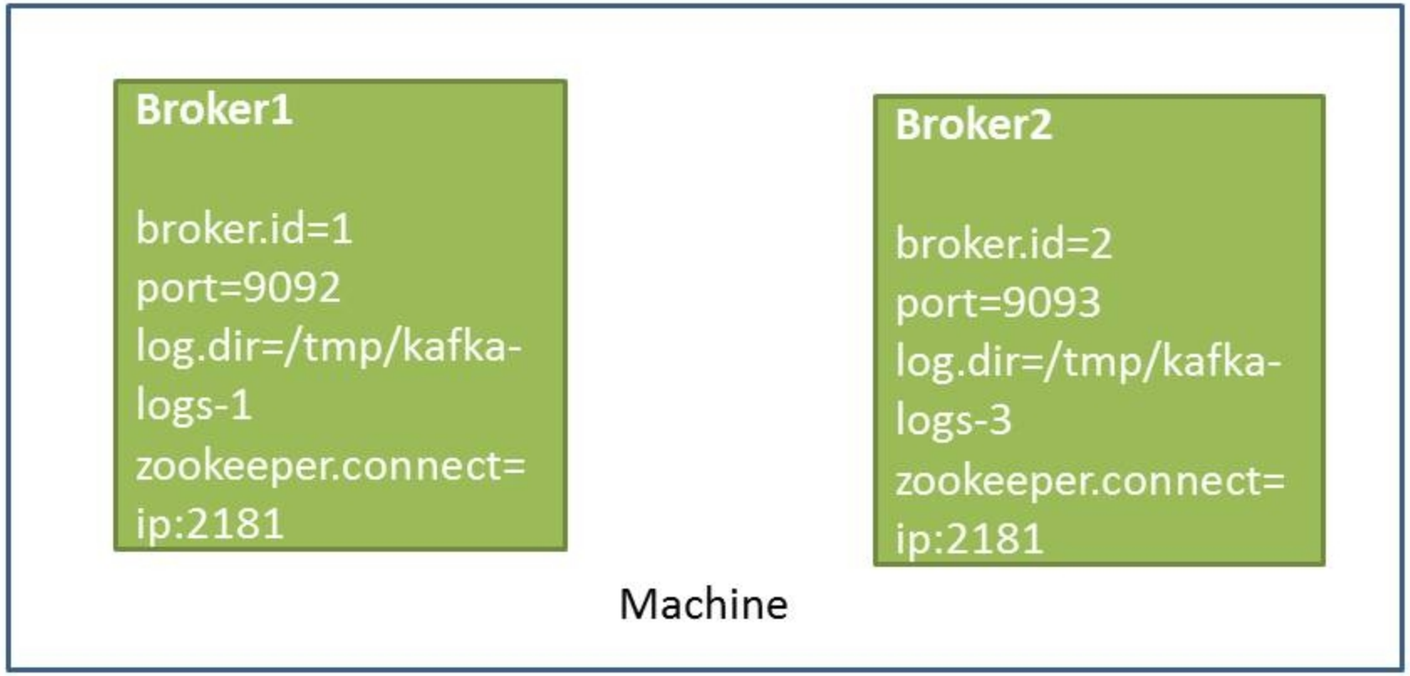
- Kafka v0.7.X: **Brokers, Producer** and **consumers** all used the zookeeper.
- Kafka v0.8.X: **Brokers** and **consumers** both used the zookeeper.
- Kafka v0.9.X: Only **Brokers** will used the zookeeper (Not Released)

Broker

- Broker is a server part of Kafka.
- Brokers are peers, there is no the master broker.
- Brokers can run on multiple nodes, but you can also run the multiple brokers on each node.
- Each broker has own IP and port for client communication.
- Configuration:
  - broker.id = 1 (Should be unique for each broker, it must be positive integer)
  - Port = 9092 (The port at which broker listen the client connection)
  - log.dir = /tmp/kafka-log (Location where broker store the data into disk)
  - zookeeper.connect = 127.0.0.1:2181 (IP's of zookeeper for coordination between broker)

Broker Deployment

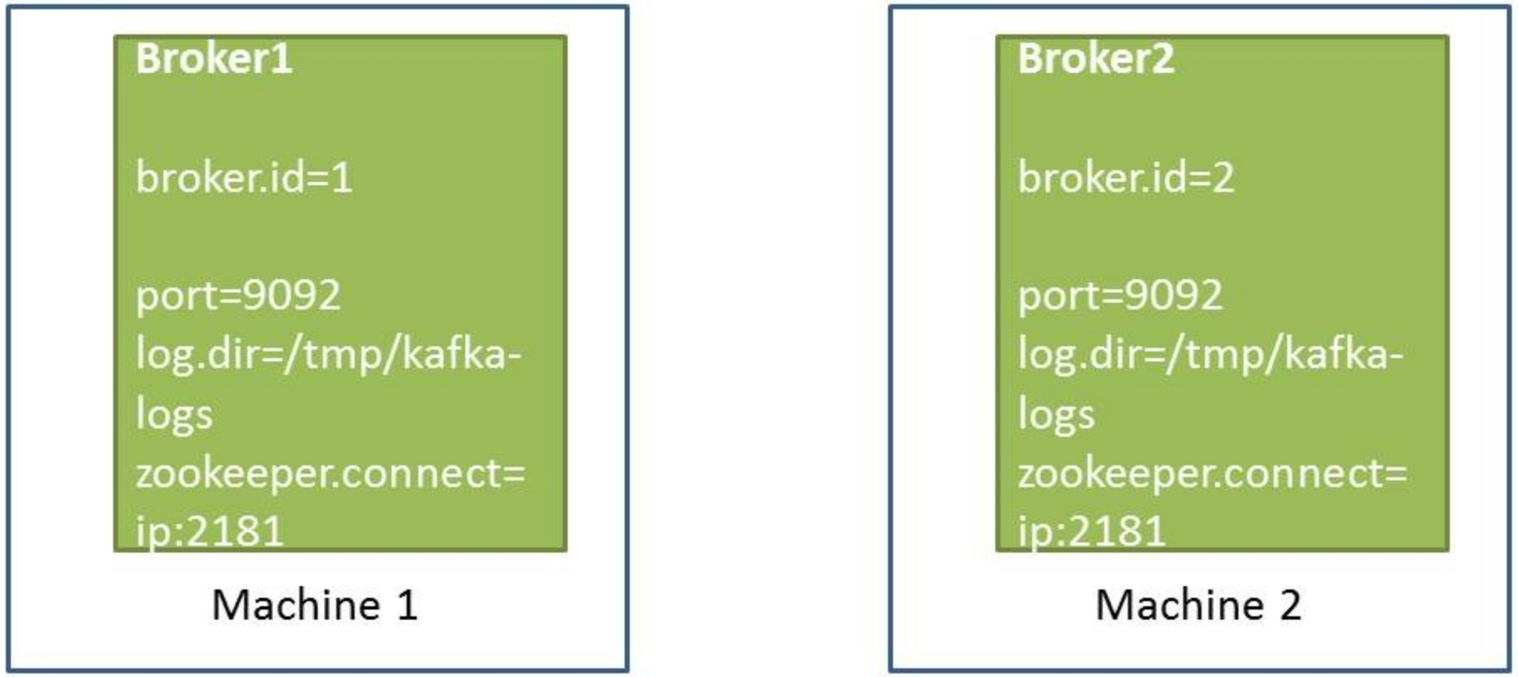
- Multiple Brokers on one machine



1. We need to modify the following properties for deploying the multiple brokers on same machine:
   broker.id
   port
   log.dir
2. The value of **zookeeper.connect** must be same on all brokers else both will form a different cluster

Broker Deployment

- One Broker on each machine



1. We need to modify the following properties for deploying one broker on each machine:

broker.id

2. Not require to change the below properties.

port

log.dir

2. The value of **zookeeper.connect** must be same on all brokers else both will form a different cluster

How data is stored?

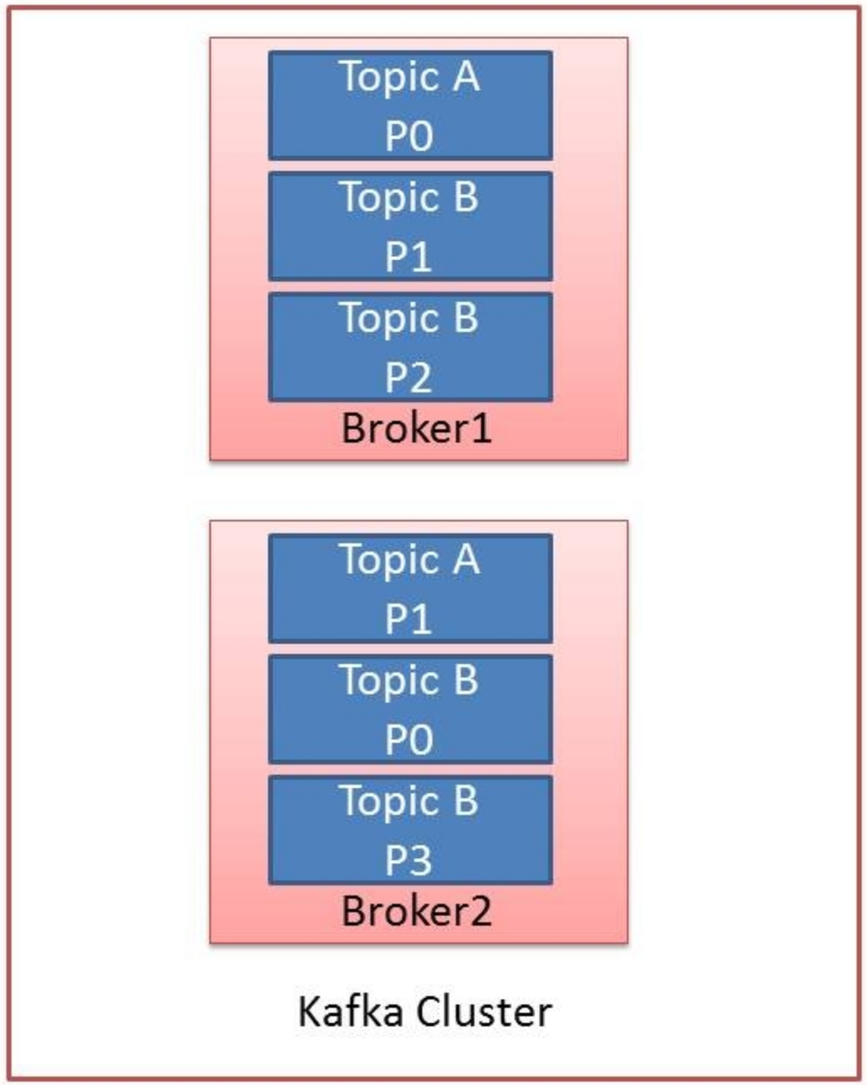
- Data is stored in topic
- A Topic is feed name or Queue name where the data is published
- A broker can handle multiple topics
- Topics are split into partitions

Why topic are split into partitions?

Why topic are split into partition?

- Assign the data of one topic to multiple nodes
- Help as to support high read/write rate
  - As we all know each disk has some limitation
  - We can write data on multiple disks at the same time
  - It help as to scale our cluster
- Property **num.partitions** define the default number of partition would create for each topic
- Also, we can define the partition for a topic at the time of creation
- Partition are replicated

Distribution of partitions

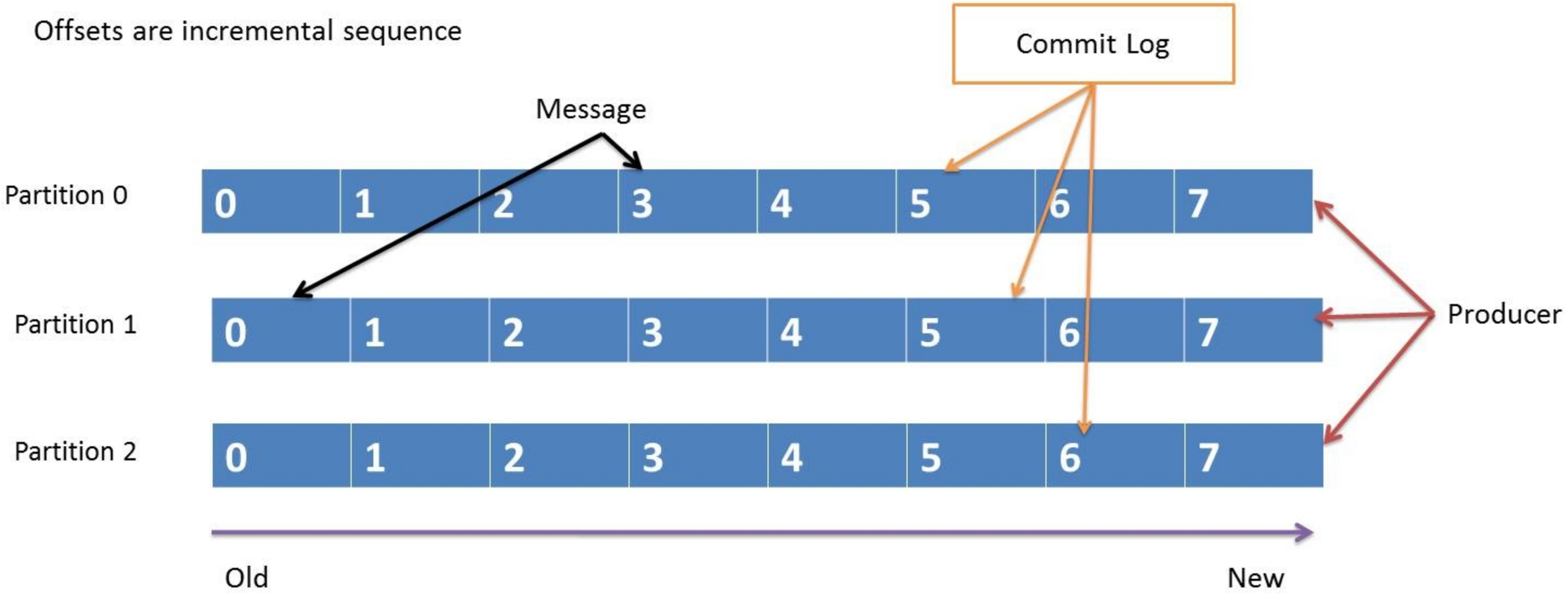


1. Topic A is created with 2 partitions
2. One partition is assign to broker A and second one assign to broker B
3. Topic B is created with 4 partitions
4. Two partitions are assign to broker A and remaining two are assign to broker B

Partition

- Each partition is an ordered, immutable sequence of messages that is continually appended to—a commit log.
- The messages in the partitions are each assigned a sequential id number called the *offset* that uniquely identifies each message within the partition.

Partition



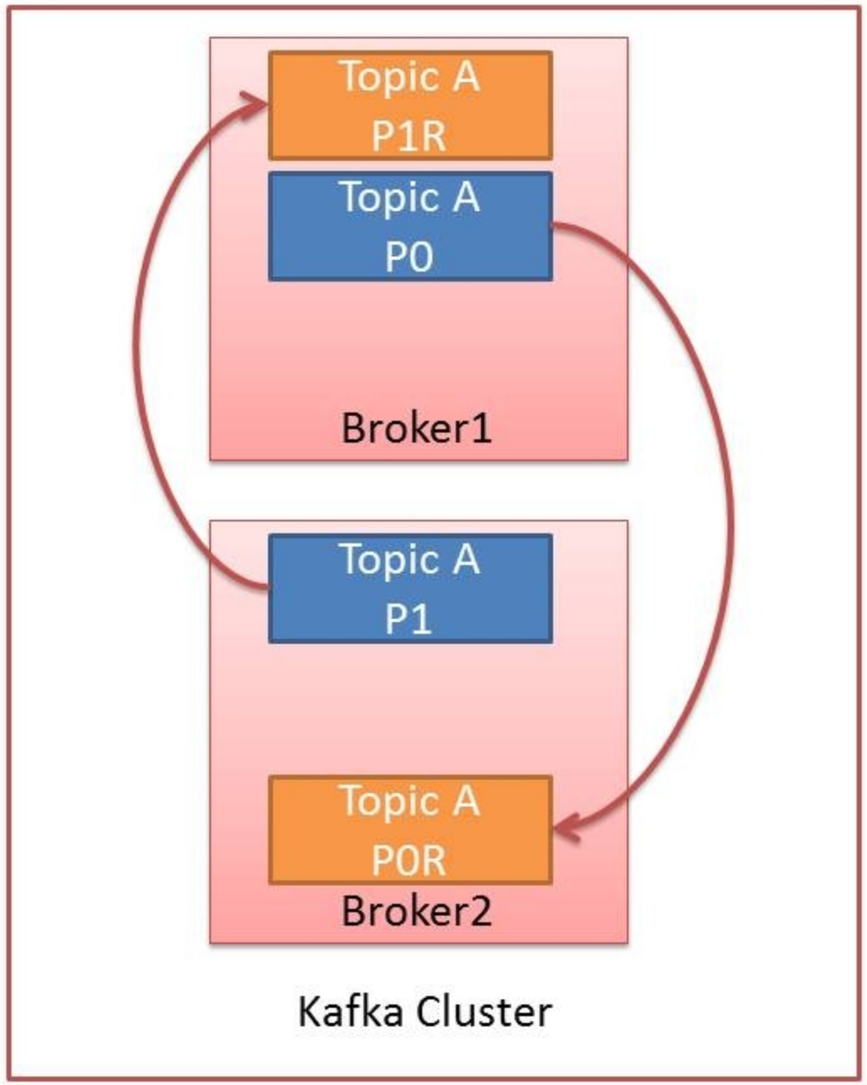
Replication

- Partition are replicated
- Partition and replica never assign to same node
- Each partition in Kafka has single leader and one or more followers
- All the read and write will happens to leader node
- Leader will copy the data to followers
- The commit log of leader and followers are identical
- It help as to support failover

Replication

- Property **default.replication.factor** define the default number of replica would create for each partition
- Also, we can define the number of replica of each partition at the time of topic creation

Replication Distribution



1. Partition and replica are never assign to same node
2. What will happens? If replication factor is greater than number of broker

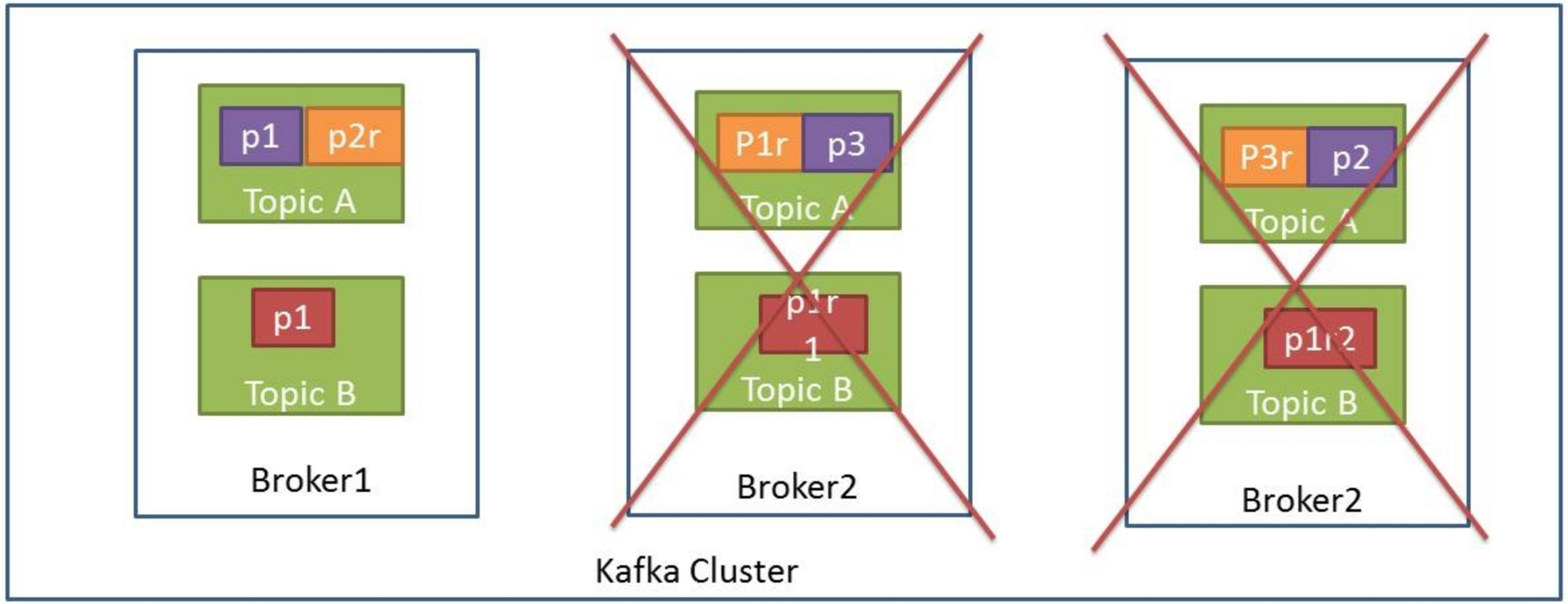
Replication

- What will happens? If replication factor is greater than number of available brokers
- Answer: Topic creation failed

How many failure Kafka Cluster can support?

- For a topic with replication factor N, we can tolerate up to N-1 server failures without losing any messages committed to the log.

How many failure Kafka Cluster can support?



Topic A

- Replication 2
- Partition 3

Topic B

- Replication 3
- Partition 1

Multiple Broker on one machine – Hands on

- Download the Kafka setup on any location on your machine
  - **http://download.nextag.com/apache/kafka/0.8.2.2/kafka_2.9.1-0.8.2.2.tgz**
  - http://apache.cs.utah.edu/kafka/0.9.0.0/kafka_2.10-0.9.0.0.tgz
  - Extract the downloaded setup
- Download the JDK 1.7 on any location on you machine
  - http://download.oracle.com/otn-pub/java/jdk/7u71-b14/jdk-7u71-linux-x64.tar.gz
  - Extract the downloaded setup
- Set JAVA_HOME environment variable
- Add the bin folder of java in $PATH variable
- Set ZOOKEEPER_HOME environment variable

Multiple Broker on one machine – Hands on

- Go to zookeeper home directory
  - > cd $ZOOKEEPER_HOME
- Now, go to zookeeper conf folder
  - > cd conf
- Run the following command
  - > mv zoo_sample.cfg zoo.cfg
- Edit the following property in zoo.cfg file
  - > data.dir=/tmp/zookeeper
- Run the below command from Zookeeper home directory
  - > ./bin/zkServer.sh start
- Check the status of Zookeeper node
  - > ./bin/zkServer.sh status

Multiple Broker on one machine – Hands on

- Go to home directory of first Kafka node
- Edit the **server.properties** configuration file
  - broker.id=0
  - port=9092
  - log.dirs=/tmp/kafka-log-1
  - zookeeper.connect=127.0.0.1:2181
- Start the first broker node
  - > bin/kafka-server-start.sh config/server.properties

Multiple Broker on one machine – Hands on

- Go to home directory of second Kafka node
- Edit the **server.properties** configuration file
  - broker.id=1
  - port=9093
  - log.dirs=/tmp/kafka-log-2
  - zookeeper.connect=127.0.0.1:2181
- Start the second broker node
  - > bin/kafka-server-start.sh config/server.properties

Topic, Partition and Replication Hands-On

- Command to create topic in Kafka
  - Partition=1, Replication=1
    - Go to home directory of Kafka and run below command
    - > bin/kafka-topics.sh --create --zookeeper 127.0.01:2181 --**replication-factor** 1 --**partitions** 1 --topic **test-replication-1**
    - Partition of topic **test-replication-1** will be assign to one machine in a cluster
  - Partition=1, Replication=2
    - > bin/kafka-topics.sh --create --zookeeper 127.0.01:2181 --**replication-factor** 2 --**partitions** 1 --topic **test-replication-2**
    - Leader partition of topic **test-replication-2** will be assign to one node while the replica will be assign to other node

Topic, Partition and Replication Hands-On

- Command to create topic in Kafka
  - Partition=4, Replication=2
    - > bin/kafka-topics.sh --create --zookeeper 127.0.01:2181 --**replication-factor** 2 --**partitions** 4 --topic **test-p4-replication-2**
    - Two replica and two leader node is assign to each node
- Instead of manually creating topics you can also configure your brokers to auto-create topics when data on non-existent topic is published.
  - auto.create.topics.enable=true
  - It will create the topic with the default replication factor and number of partitions.

Topic, Partition and Replication Hands-On

- Describe command
  - This command is used to collect the metadata of a topic
  - > bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic test-p4-replication-2
  - Output of above command
    - Topic:test-p4-replication-2 PartitionCount:4 ReplicationFactor:2
    - Topic: test-p4-replication-2 Partition: 0 Leader: 1 Replicas: 1,0 Isr: 1,0
    - Topic: test-p4-replication-2 Partition: 1 Leader: 0 Replicas: 0,1 Isr: 0,1
    - Topic: test-p4-replication-2 Partition: 2 Leader: 1 Replicas: 1,0 Isr: 1,0
    - Topic: test-p4-replication-2 Partition: 3 Leader: 0 Replicas: 0,1 Isr: 0,1

Topic, Partition and Replication Hands-On

- Let's consider one node in a cluster goes down
  - Now, again run describe command:
  - > bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic test-p4-replication-2
  - Output of above command
    - Topic:test-p4-replication-2 PartitionCount:4 ReplicationFactor:2
    - Topic: test-p4-replication-2 Partition: 0 **Leader: 0** Replicas: 1,0 Isr: 0
    - Topic: test-p4-replication-2 Partition: 1 **Leader: 0** Replicas: 0,1 Isr: 0
    - Topic: test-p4-replication-2 Partition: 2 **Leader: 0** Replicas: 1,0 Isr: 0
    - Topic: test-p4-replication-2 Partition: 3 **Leader: 0** Replicas: 0,1 Isr: 0

Topic, Partition and Replication Hands-On

- Create a topic which has replication factor greater than number of brokers:
  - > bin/kafka-topics.sh --create --zookeeper 127.0.01:2181 --replication-factor 3 --partitions 4 --topic test-p4-replication-3
  - What will happen?
  - Topic creation will failed with below exception:
    - **Error while executing topic command replication factor: 3 larger than available brokers: 2**

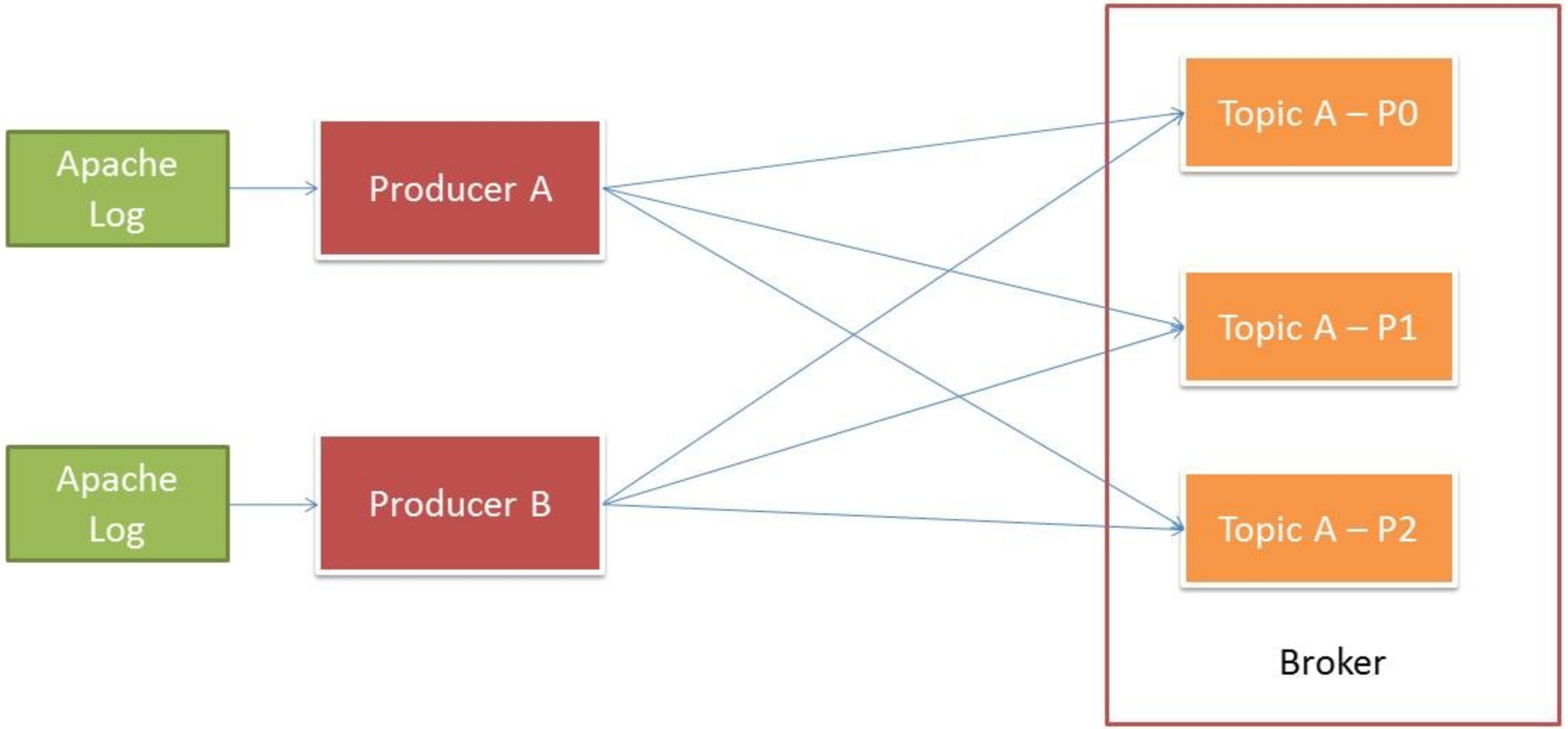
Producer

- The producer push data directly to the broker that is the **leader for the partition**.
- Is Producer use zookeeper?
  - No (After Kafka v 0.8)
- How producer identify the destination leader partition?
  - Producer collect the metadata from brokers about which servers are alive and where the leaders for the partitions of a topic are assigned

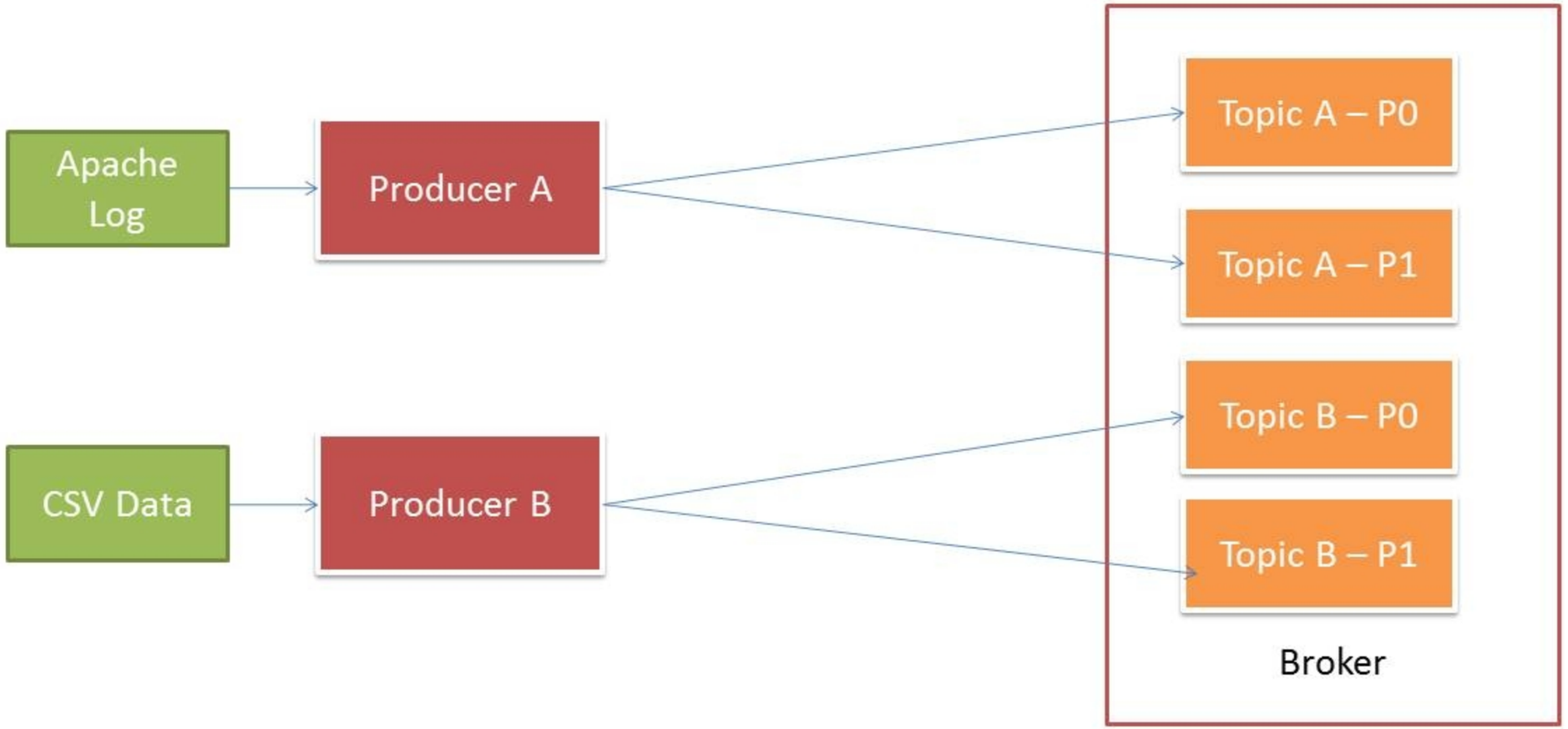
Producer

- metadata.broker.list = host1:port1,host2:port2
  - The producer will only use it for getting metadata (topics, partitions and replicas)
  - The socket connections for sending the actual data will be established based on the broker information returned in the metadata.

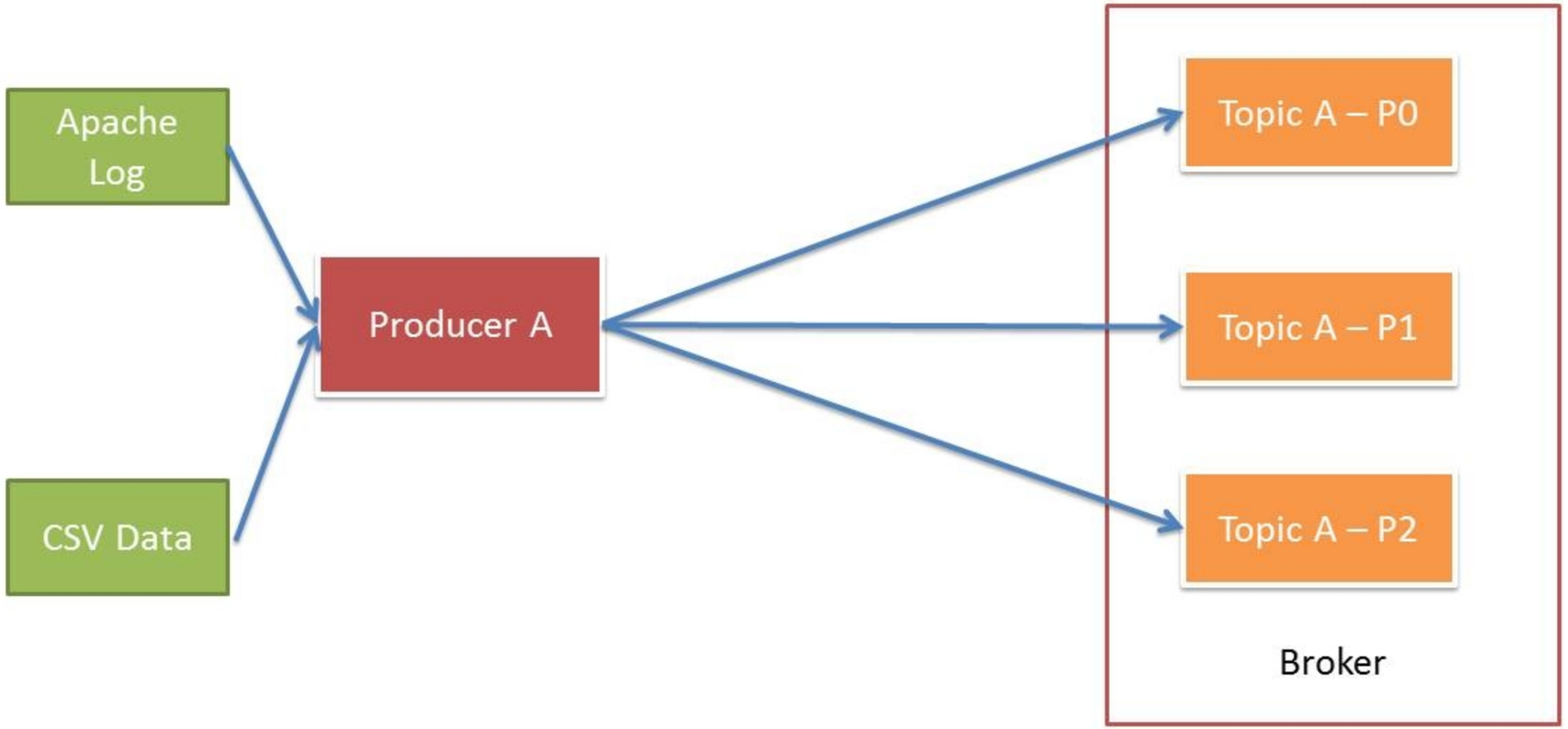
Producer



Producer



Producer



Producer

1. How producer distribute the data on partitions?
2. Is all the partitions contain equal amount of data?

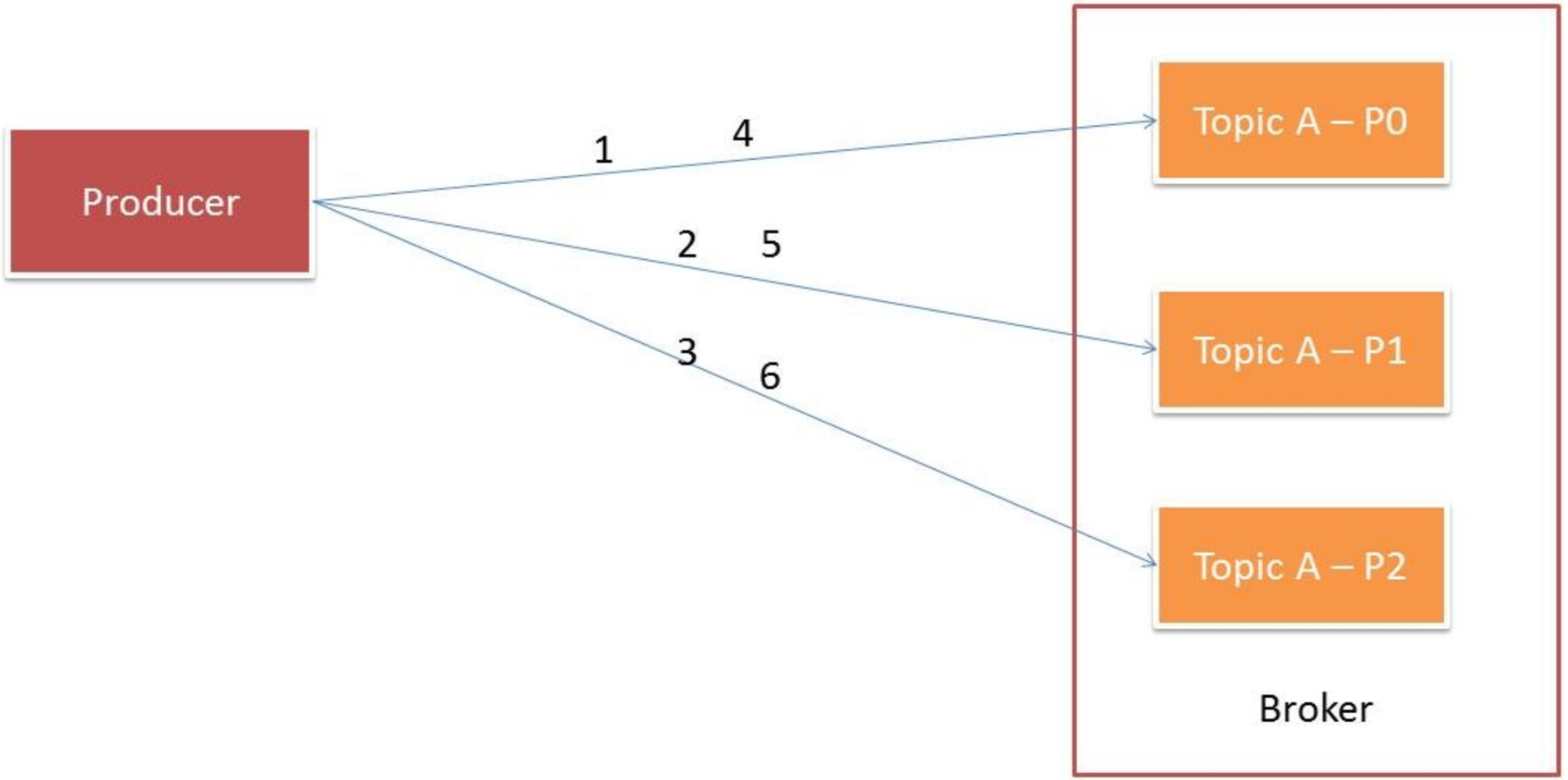
Producer Architecture

- How producer distribute the data on partitions?
  - By default, producer push data randomly to any partition
- Is all the partitions contain equal amount of data?
  - No

Producer – Custom Partition

- We can define our own data partition mechanism
- Most commonly used partition mechanism are round robin partition
- Advantage:
  - All the partition contains equals amount of data
- Configuration
  - partitioner.class = com.abc.partition.RoundRobinPartition

Producer – RoundRobinPartition

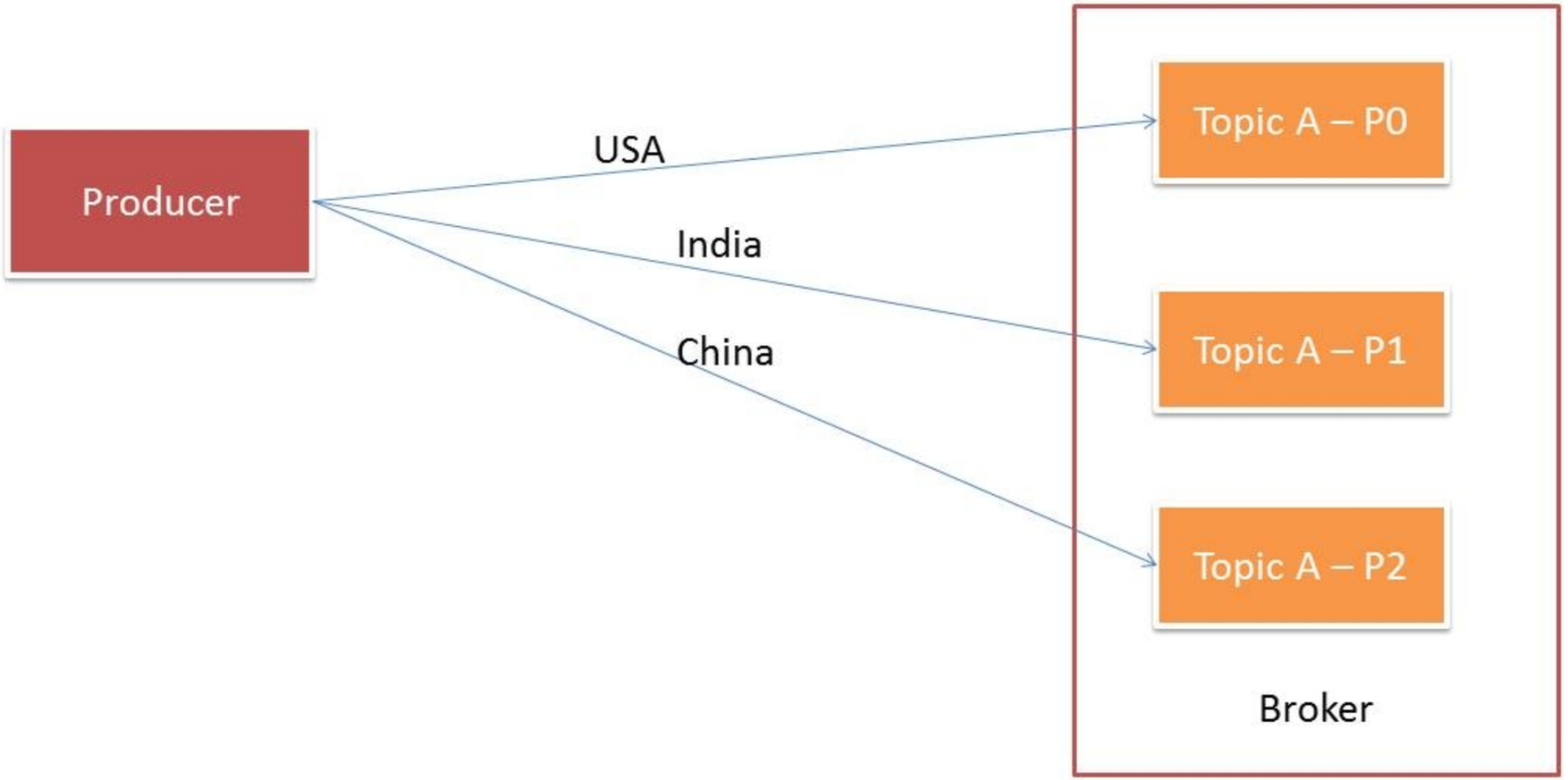


Distribute data uniformly between all partitions

Producer – Custom Partition

- We can also partition our data on the basis of field value.
- For example:
  - If the key chosen was a country then all data for a given country would be sent to the same partition.

Producer – Field Based Partition



DataFlair Web Services Pvt Ltd

+91-8451097879
info@data-flair.com
http://data-flair.com