Kafka architecture

INTRODUCTION TO KAFKA

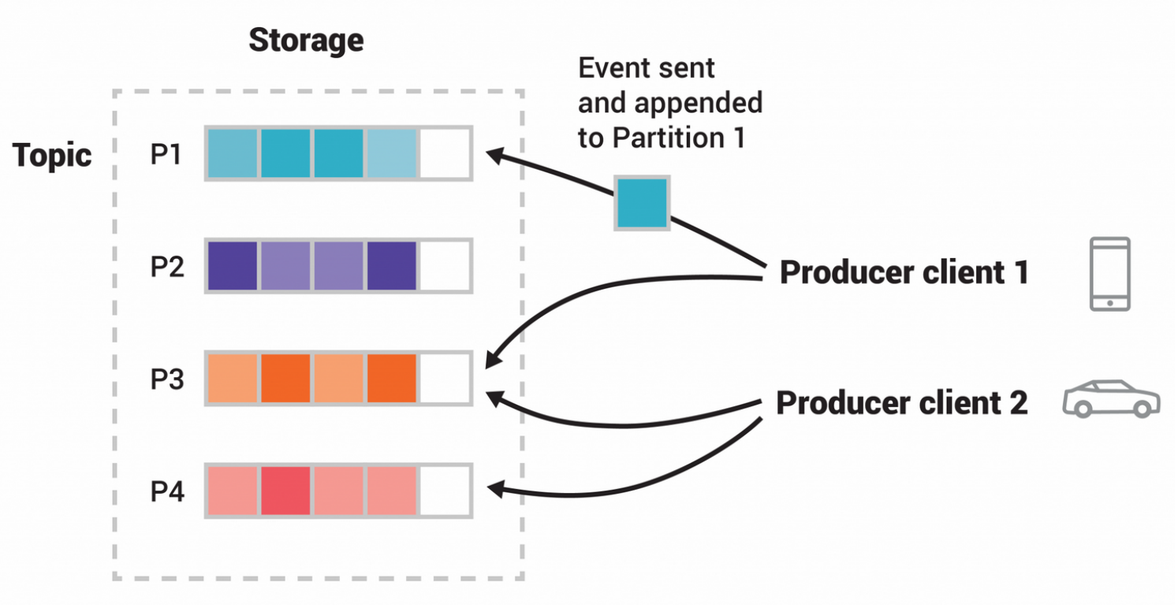
Mike Metzger
Data Engineering Consultant

Kafka components

- Kafka server
  - A cluster of one or more computers
  - Stores data
  - Manages communications
  - Can also integrate with other systems (databases, logs, etc))
- Kafka clients
  - Read via Kafka consumer
  - Write via Kafka producer
  - Process data as required

Kafka server

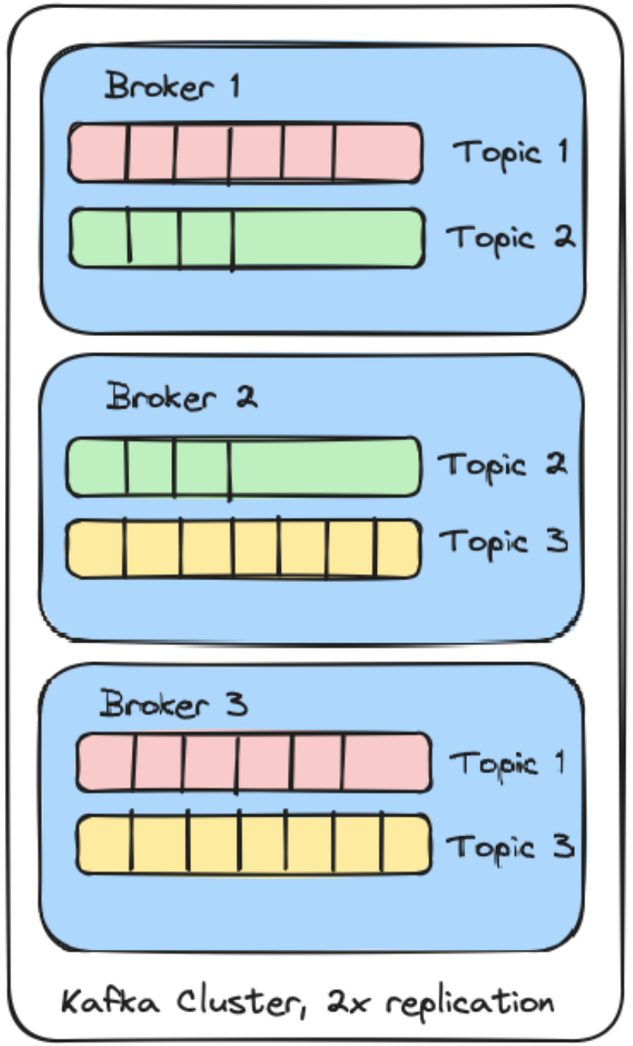
- Storage, also known as the Kafka Broker
  - Data written by producers is stored and organized via topics
  - Topics are partitioned, or stored in separate pieces
  - Messages / events stored in a given partition based on an event id
  - Messages are retrieved in same order as written



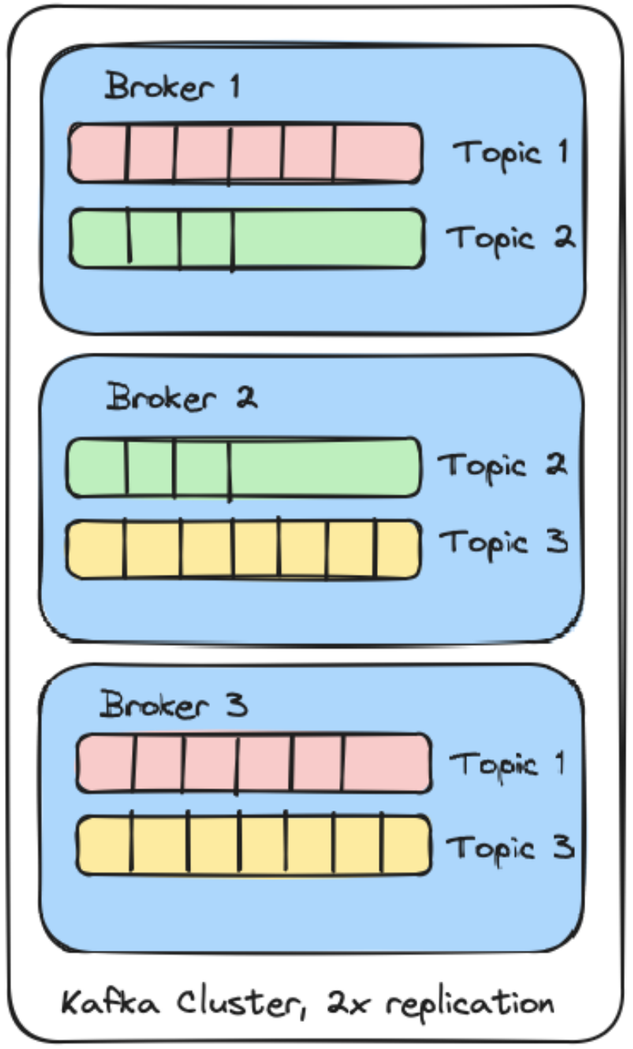
[1] Image sourced from https://kafka.apache.org/intro

Partitions & replication

- Kafka is fault-tolerant
- If one system goes offline, others can provide the requested data
- Max number of failures - 1 == replication factor
- Max replication factor == number of servers
- Copying partitions are how replication is handled

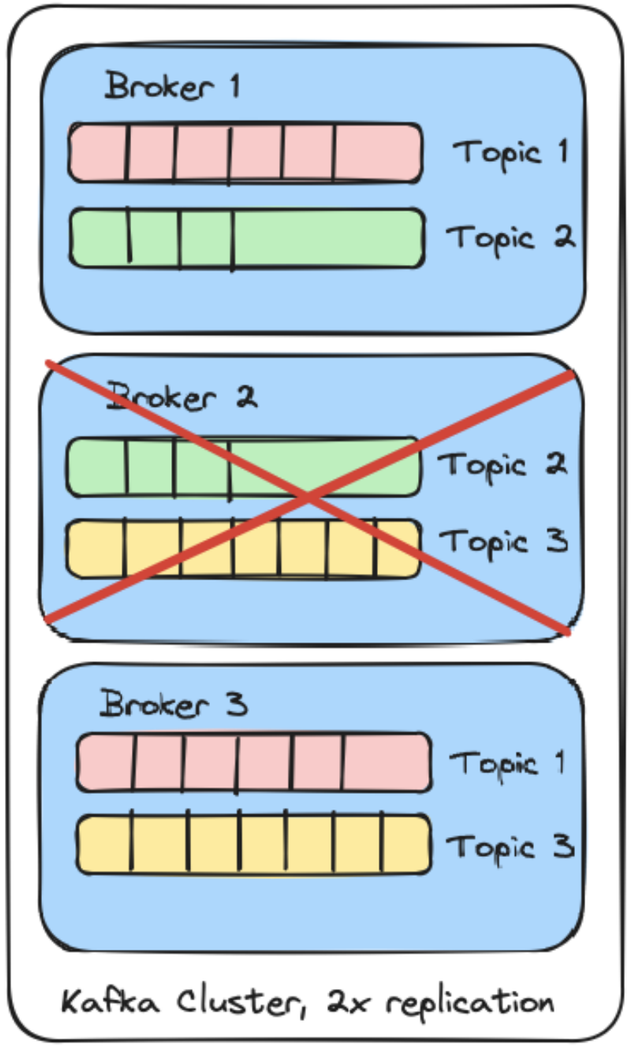


Example



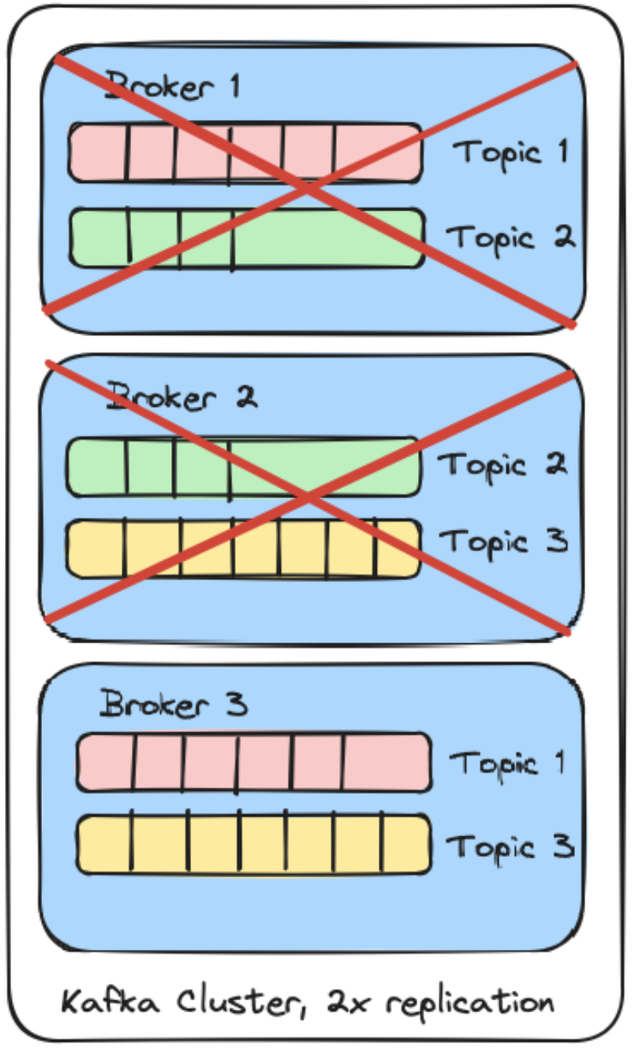
- Kafka cluster with 3 brokers
- 3 topics defined
- 2x replication == can lose 1 system
- Broker 1 has copies of Topic 1 & Topic 2
- Broker 2 has copies of Topic 2 & Topic 3
- Broker 3 has copies of Topic 1 & Topic 3
- Each topic is shared across 2 brokers within the cluster

Example with 1 failure



- Broker 2 fails
- Replication factor 2x
  - 2 - 1 == Can handle 1 failed system
- Each topic still has at least one copy on the cluster

Example with 2 failures



- 2 failed brokers
- Replication factor 2x
- More failures than supported (1 system)
- Topic 2 is no longer available in the cluster
  - Topics 1 and 3 are still available, so not a complete failure

Let's practice!

INTRODUCTION TO KAFKA

Creating and managing Kafka clusters

INTRODUCTION TO KAFKA

Mike Metzger
Data Engineering Consultant

What is ZooKeeper?

- ZooKeeper is a framework to manage information & provide services necessary for running distributed systems
- Primarily used by developers to create distributed applications
  - Users interact with ZooKeeper
- Example applications
  - Kafka
  - Hadoop
  - Neo4j

APACHE ZooKeeper™

What does ZooKeeper do?

- ZooKeeper provides services necessary to run distributed applications
  - Handling configuration
  - System naming
  - Synchronization across systems
  - Services required by a group of systems
- Designed as a framework to prevent individual distributed applications from implementing custom versions of services.
  - Like a common connector, such as a power plug or hose nozzle

ZooKeeper and Kafka

- Kafka uses ZooKeeper for cluster management
  - Newer versions of Kafka can use KRaft
- Two files used
  - `config/zookeeper.properties`
  - `config/server.properties`

```
# zookeeper.properties

# The directory where the
# snapshot is stored.
dataDir=/tmp/zookeeper

# The port at which the clients
# will connect
clientPort=2181

...
```

config/server.properties

- Handles specific Kafka configuration
  - Broker details
  - Network configuration
  - Event storage location
  - Basic topic configurations, including replication

```
...
# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs
# The default number of log partitions per topic.
num.partitions=1
```

Starting a Kafka cluster

- Kafka clusters are started in two pieces

```
$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
...
INFO   ______                   _                                                (org.apac
INFO  |___  /                  | |                                               (org.apac
INFO     / /    ___     ___    | | __   ___     ___    _ __      ___    _ __     (org.apac
INFO    / /    / _ \   / _ \   | |/ /  / _ \   / _ \  | '_ \    / _ \  | '__|  (org.apache
INFO   / /__  | (_) | | (_) |  |   <  |  __/ |  __/  | |_) |  |  __/  | |      (org.apach
INFO  /_____|  \___/   \___/   |_|\_\  \___|  \___|  | .__/    \___|  |_|  (org.apache.zo
INFO                                                 | |                        (org.apa
INFO                                                 |_|                        (org.apa
INFO  (org.apache.zookeeper.server.ZooKeeperServer)
```

Starting a Kafka cluster (continued)

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

```
INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.DataPlaneAcceptor)
INFO Kafka version: 3.7.0 (org.apache.kafka.common.utils.AppInfoParser)
INFO Kafka commitId: 2ae524ed625438c5 (org.apache.kafka.common.utils.AppInfoParser)
INFO Kafka startTimeMs: 1717502877829 (org.apache.kafka.common.utils.AppInfoParser)
INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
INFO [zk-broker-0-to-controller-forwarding-channel-manager]: Recorded new controller
     from now on will use node 815f25786085:9092 (id: 0 ra
```

Stopping a Kafka cluster

- `bin/kafka-server-stop.sh`
- `bin/zookeeper-server-stop.sh`
- Note the reverse order of shutdown

Let's practice!

INTRODUCTION TO KAFKA

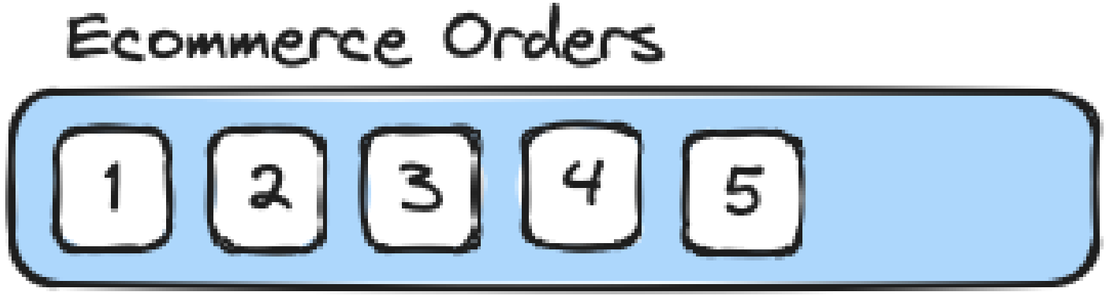
Kafka topics

INTRODUCTION TO KAFKA

Mike Metzger
Data Engineering Consultant

What is a topic?

- Topics are logical groupings of events
  - Similar to a table in a relational database
  - Event log
  - Messages are immutable
  - Messages in a topic can be read or written, but not modified
  - Messages can be removed based on age



Creating topics

- Multiple ways to create topics
- We'll use the `bin/kafka-topics.sh` script

```
bin/kafka-topics.sh
    --bootstrap-server <server>
    --topic <topicname>
    --create
```

- Example

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092 \
--topic orders --create
```

```
Created topic orders
```

Other variations

- Optional arguments for `bin/kafka-topics.sh --create`
  - `--replication-factor <x>`
    - Define the replication factor of the topic
  - `--partitions <x>`
    - Specify the number of partitions manually
- Example

```
$ bin\kafka-topics.sh --bootstrap-server localhost:9092 \
--topic orders --create --replication-factor 3 \
--partitions 3
```

```
Created topic orders
```

--describe

- `bin/kafka-topics.sh --bootstrap-server <server> --topic <topicname> --describe`
  - Get details about the topic configuration
- Example

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092 \
   --topic orders --describe
```

```
Topic: orders-new TopicId: <topicid> PartitionCount: 3 ReplicationFactor: 1
  Configs:
    Topic: orders-new    Partition: 0    Leader: 0    Replicas: 0    Isr: 0
    Topic: orders-new    Partition: 1    Leader: 0    Replicas: 0    Isr: 0
    Topic: orders-new    Partition: 2    Leader: 0    Replicas: 0    Isr: 0
```

Removing topics

- `bin/kafka-topics.sh --bootstrap-server <server> --topic <topicname> --delete`
  - Removes a topic from Kafka server
  - Removing the topic also removes all messages in the topic
- Example

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092
  --topic orders --delete
```

```
$
```

Let's practice!

INTRODUCTION TO KAFKA

Kafka troubleshooting

INTRODUCTION TO KAFKA

Mike Metzger
Data Engineering Consultant

"Helpful" behaviors

- Kafka tries to be helpful to users
- Sometimes causes issues instead
- By default, a topic does not need to exist before writing
- Consider what happens if you misspell a topic name (ie, `ordrs` instead of `orders`)

[1] Image courtesy Dex Ezekiel on Unsplash

"Helpful" example

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

```
orders
```

```
$ echo "Test message" | bin/kafka-console-producer.sh
   --bootstrap-server localhost:9092 \
   --topic ordrs
```

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

```
orders
ordrs
```

Connectivity issues

- Kafka is a networked service
- Any networking issues can cause problems with Kafka communications
- Look at the output of commands to determine potential issues

`bin\kafka-topics.sh --bootstrap-server localhost:9092 --list`

```
WARN [AdminClient clientId=adminclient-1]
  Connection to node -1 (localhost/127.0.0.1:9092) could not be established.
  Node may not be available. (org.apache.kafka.clients.NetworkClient)
```

- Check Kafka service is running (`ps ax | grep kafka` , `netstat -tlnp | grep 9092`)
- Check for firewall issues
- Verify correct port / IP

Other common problems

- Consumers:
  - Use `--from-beginning` if you need older messages
  - Remember `--max-messages` to only read a specific message quantity
- All tools:
  - Remember to include the `--bootstrap-server`
  - Most error messages are pretty clear and point you to likely solutions
  - Tools also have a `--help` option to get further details

Let's practice!

INTRODUCTION TO KAFKA

Congratulations!

INTRODUCTION TO KAFKA

Mike Metzger
Data Engineering Consultant

Review

- Kafka components
  - Producers
  - Consumers
  - Topics
- Kafka architecture
  - ZooKeeper
  - Kafka servers / brokers
- Troubleshooting

Next steps

- Review kafka documentation
- Running `--help` on various Kafka commands
- Working with Kafka libraries for programming languages
  - Python
  - Rust
  - etc

[1] https://kafka.apache.org

Great work!

INTRODUCTION TO KAFKA