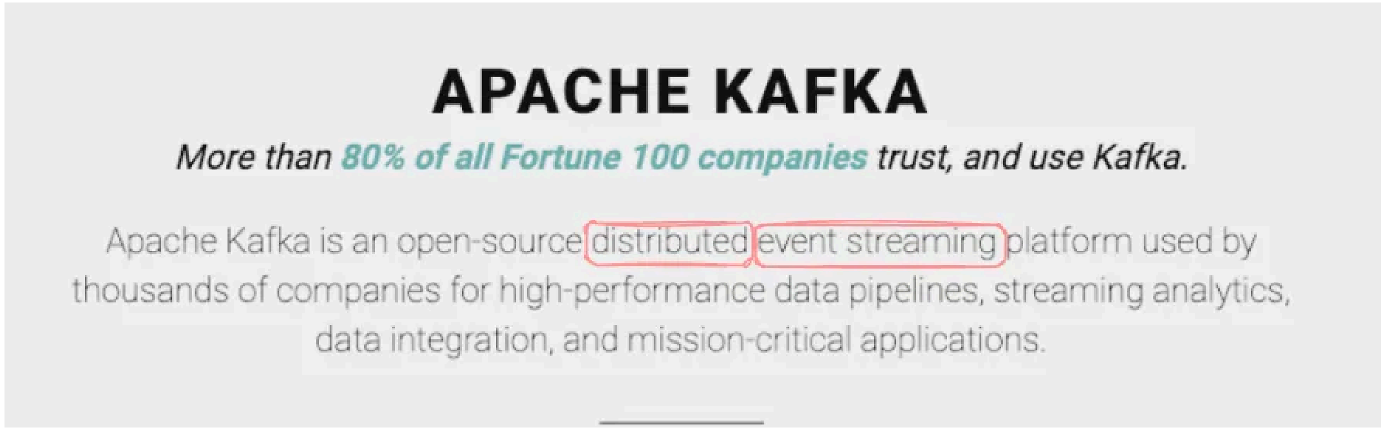
What is kafka?

https://kafka.apache.org/


APACHE KAFKA

More than 80% of all Fortune 100 companies trust, and use Kafka.

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

What is distributed

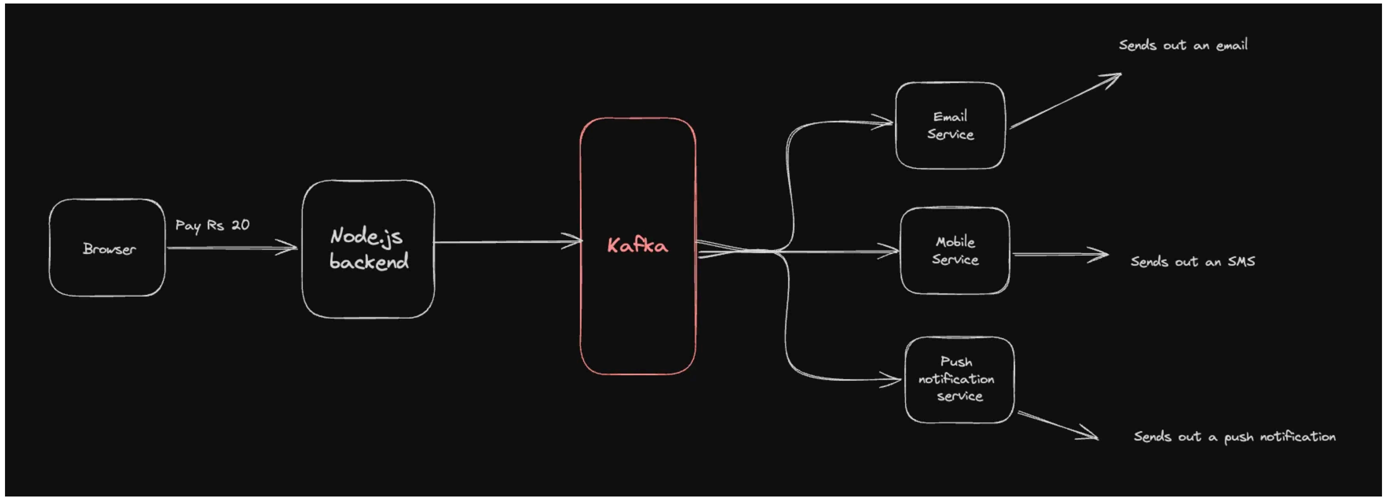
You can scale kafka horizontally by adding more nodes that run your kafka `brokers`

Event streaming

If you want to build a system where one process `produces` events that can be consumed by multiple `consumers`

Examples of apps

Payment notifications



Jargon

Cluster and broker

A group of machines running kafka are known as a kafka cluster

Each individual machine is called a broker

Producers

As the name suggests, producers are used to `publish` data to a topic

Consumers

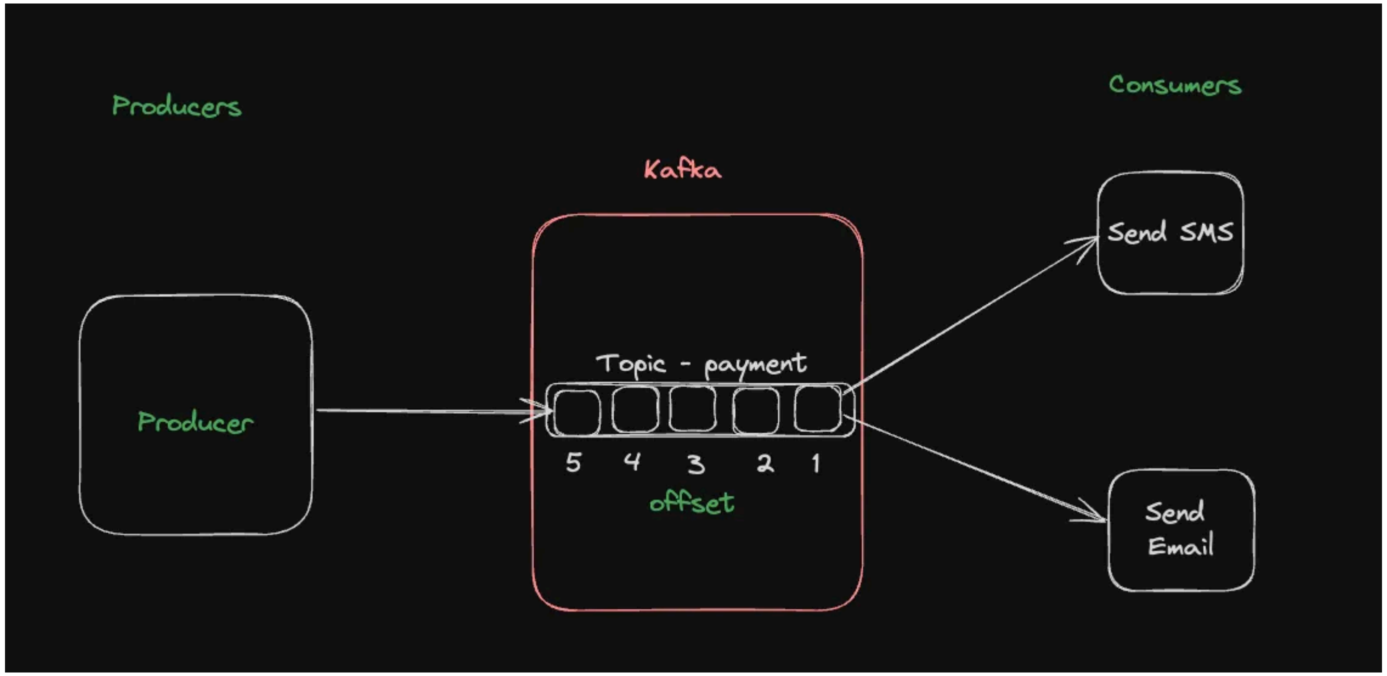
As the name suggests, consumers consume from a topic

Topics

A topic is a logical channel to which producers send messages and from which consumers read messages.

Offsets

Consumers keep track of their position in the topic by maintaining offsets, which represent the position of the last consumed message. Kafka can manage offsets automatically or allow consumers to manage them manually.



Retention

Kafka topics have configurable retention policies, determining how long data is stored before being deleted. This allows for both real-time processing and historical data replay.

Start kafka locally

Ref - https://kafka.apache.org/quickstart

Using docker

```
docker run -p 9092:9092 apache/kafka:3.7.1
```

Get shell access to container

```
docker ps
docker exec -it container_id /bin/bash
cd /opt/kafka/bin
```

Create a topic

```
./kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
```

Publish to the topic

```
./kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
```

Consuming from the topic

```
./kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-ser
```

Kafka in a Node.js process

Ref https://www.npmjs.com/package/kafkajs

- Initialise project

```
npm init -y
npx tsc --init
```

- Update package.json

```
"rootDir": "./src",
"outDir": "./dist"
```

- Add `src/index.ts`

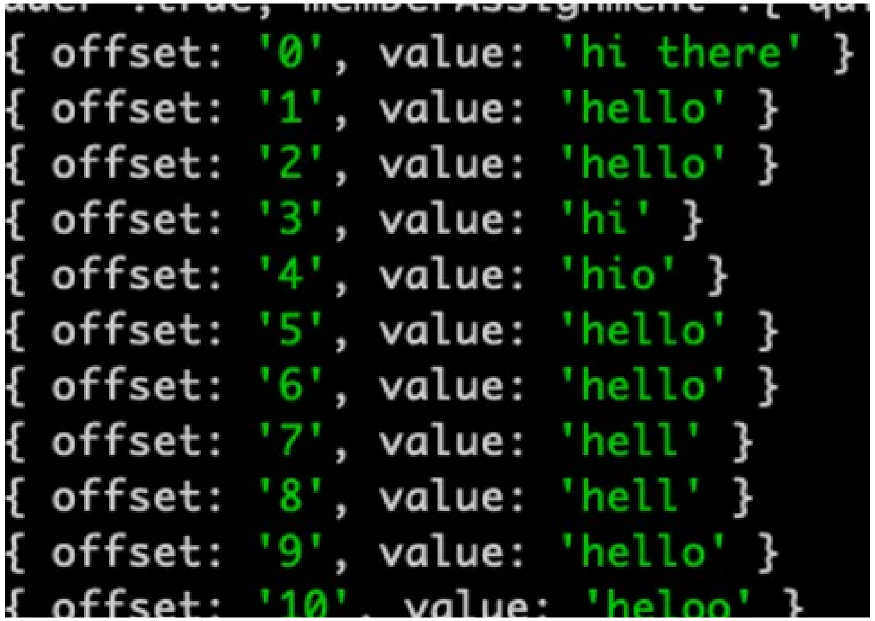
```
import { Kafka } from "kafkajs";

const kafka = new Kafka({
  clientId: "my-app",
  brokers: ["localhost:9092"]
})

const producer = kafka.producer();

const consumer = kafka.consumer({groupId: "my-app3"});


async function main() {
  await producer.connect();
  await producer.send({
    topic: "quickstart-events",
    messages: [{
      value: "hi there"
    }]
  })

  await consumer.connect();
  await consumer.subscribe({
    topic: "quickstart-events", fromBeginning: true
  })

  await consumer.run({
    eachMessage: async ({ topic, partition, message }) => {
      console.log({
        offset: message.offset,
        value: message?.value?.toString(),
      })
    },
  })
}


main();
```

- Update package.json

```
"scripts": {
    "start": "tsc -b && node dist/index.js"
},
```

- Start the process

```
npm run start
```



```
{ offset: '0', value: 'hi there' }
{ offset: '1', value: 'hello' }
{ offset: '2', value: 'hello' }
{ offset: '3', value: 'hi' }
{ offset: '4', value: 'hio' }
{ offset: '5', value: 'hello' }
{ offset: '6', value: 'hello' }
{ offset: '7', value: 'hell' }
{ offset: '8', value: 'hell' }
{ offset: '9', value: 'hello' }
{ offset: '10', value: 'heloo' }
```

Breaking into prodcuer and consumer scripts

Lets break our logic down into two saparate files

- producer.ts

```
import { Kafka } from "kafkajs";

const kafka = new Kafka({
  clientId: "my-app",
  brokers: ["localhost:9092"]
})

const producer = kafka.producer();

async function main() {
  await producer.connect();
  await producer.send({
    topic: "quickstart-events",
    messages: [{
      value: "hi there"
    }]
  });
}


main();
```

- consumer.ts

```
import { Kafka } from "kafkajs";

const kafka = new Kafka({
  clientId: "my-app",
  brokers: ["localhost:9092"]
})

const consumer = kafka.consumer({ groupId: "my-app3" });
```

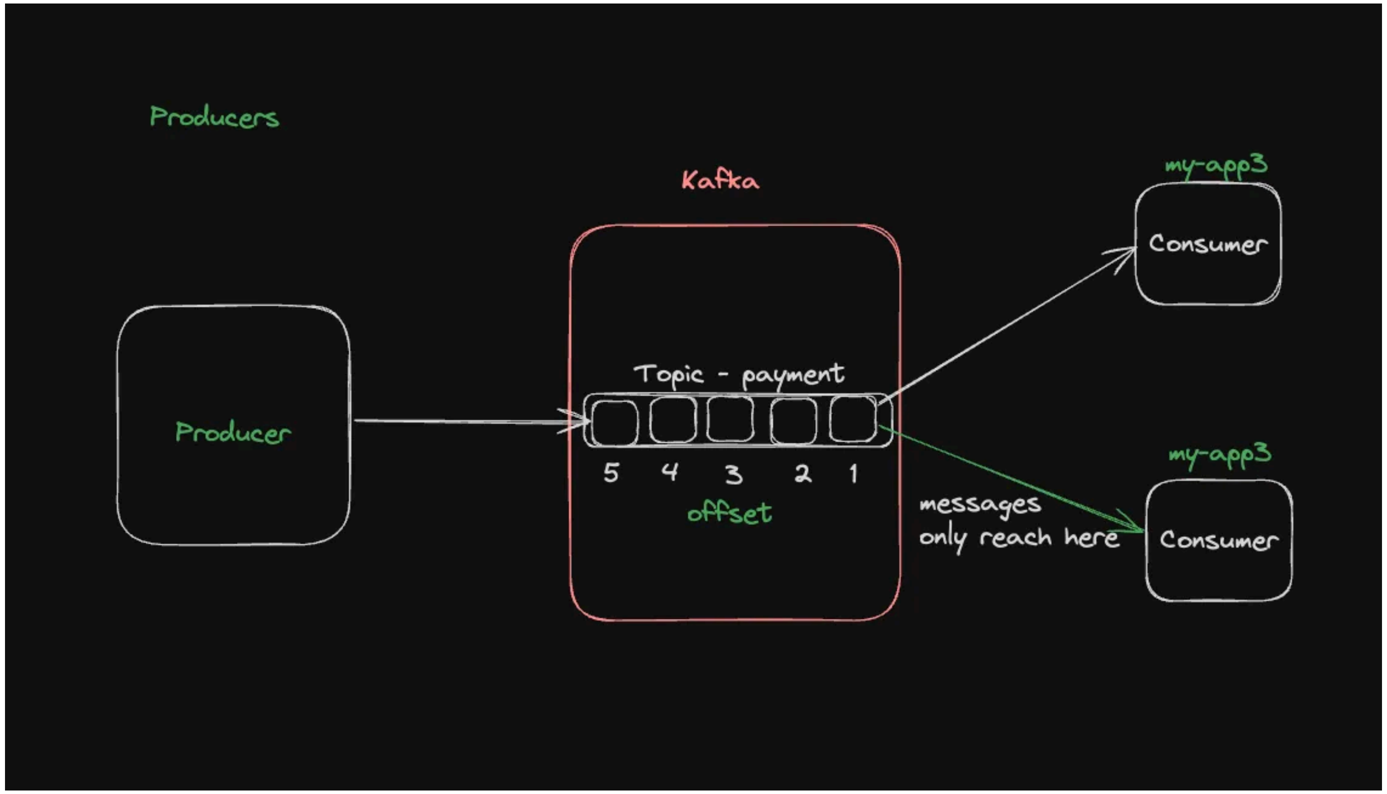
```
async function main() {
  await consumer.connect();
  await consumer.subscribe({
    topic: "quickstart-events", fromBeginning: true
  })

  await consumer.run({
    eachMessage: async ({ topic, partition, message }) => {
      console.log({
        offset: message.offset,
        value: message?.value?.toString(),
      })
    },
  })
}


main();
```

- Update package.json

```
  "scripts": {
    "start": "tsc -b && node dist/index.js",
    "produce": "tsc -b && node dist/producer.js",
    "consume": "tsc -b && node dist/consumer.js"
  },
```

- Try starting multiple consumers, and see if each gets back a message for the messages produced

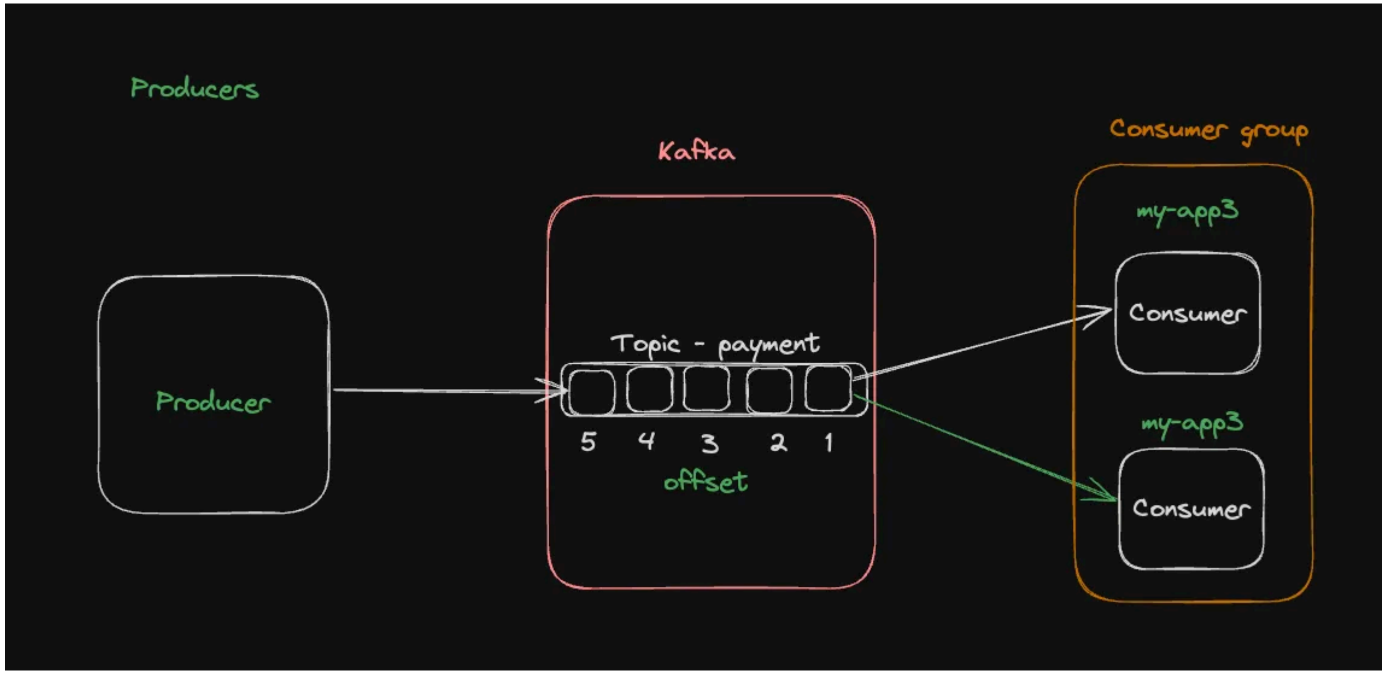
Notice we specified a `consumer group` (my-app3)



Consumer groups and partitions

Consumer group

A consumer group is a group of consumers that coordinate to consume messages from a Kafka topic.



Purpose:

- **Load Balancing:** Distribute the processing load among multiple consumers.

- **Fault Tolerance:** If one consumer fails, Kafka automatically redistributes the partitions that the failed consumer was handling to the remaining consumers in the group.
- **Parallel Processing:** Consumers in a group can process different partitions in parallel, improving throughput and scalability.

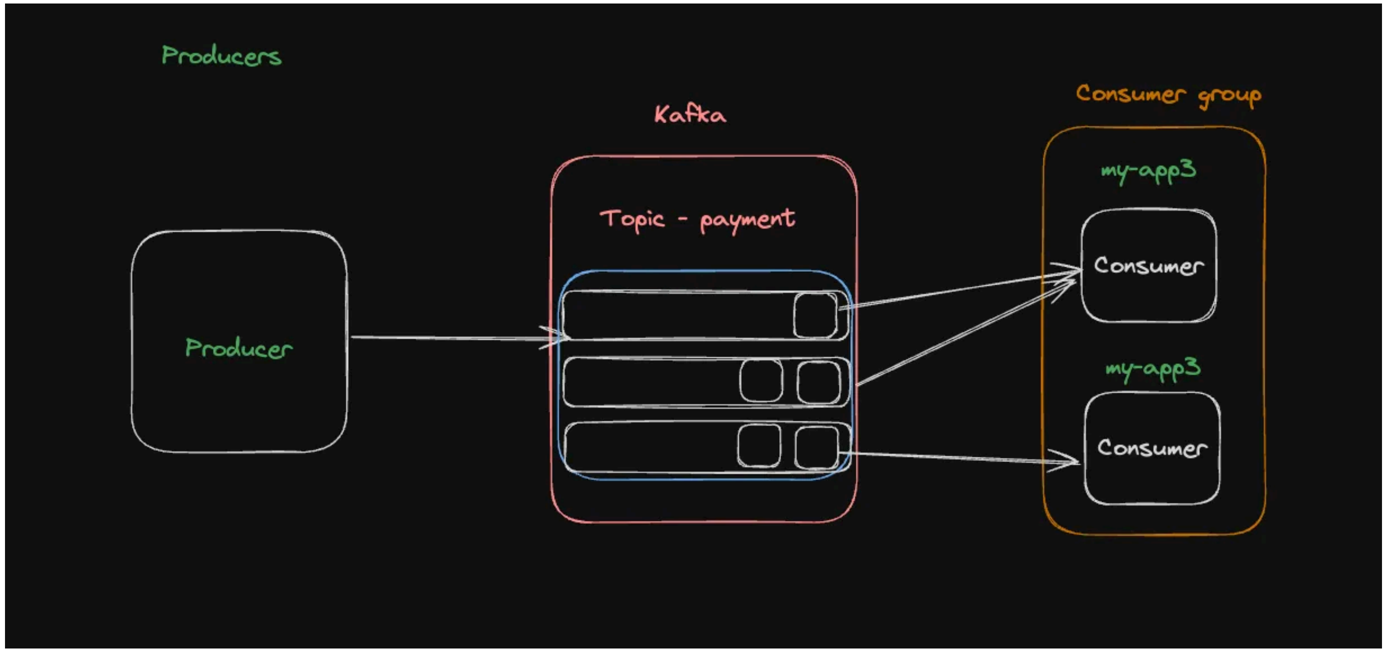
Partitions

Partitions are subdivisions of a Kafka topic. Each partition is an ordered, immutable sequence of messages that is appended to by producers. Partitions enable Kafka to scale horizontally and allow for parallel processing of messages.

How is a partition decided?

When a message is produced to a Kafka topic, it is assigned to a specific partition. This can be done using a round-robin method, a hash of the message key, or a custom partitioning strategy.

Usually you’ll take things like `user id` as the `message key` so all messages from the same user go to the same consumer (so a single user doesnt starve everyone lets say)



Multiple consumer groups

Partitions in kafka

In this slide, we'll talk about what are partitions in Kafka

- Create a new topic with 3 partitions

```
./kafka-topics.sh --create --topic payment-done --partitions 3 --bootstrap-server localhc
```

- Ensure it has 3 partitions

```
./kafka-topics.sh --describe --topic payment-done --bootstrap-server localhost:9092
```

- Update the topic in the node.js script to use `payment-done`

```
async function main() {
 await producer.connect();
 await producer.send({
  topic: "payment-done",
  messages: [{
   value: "hi there",
   key: "user1"
```

```
      }]
    });
  }
  ///
  await consumer.subscribe({
    topic: "payment-done", fromBeginning: true
  })
```

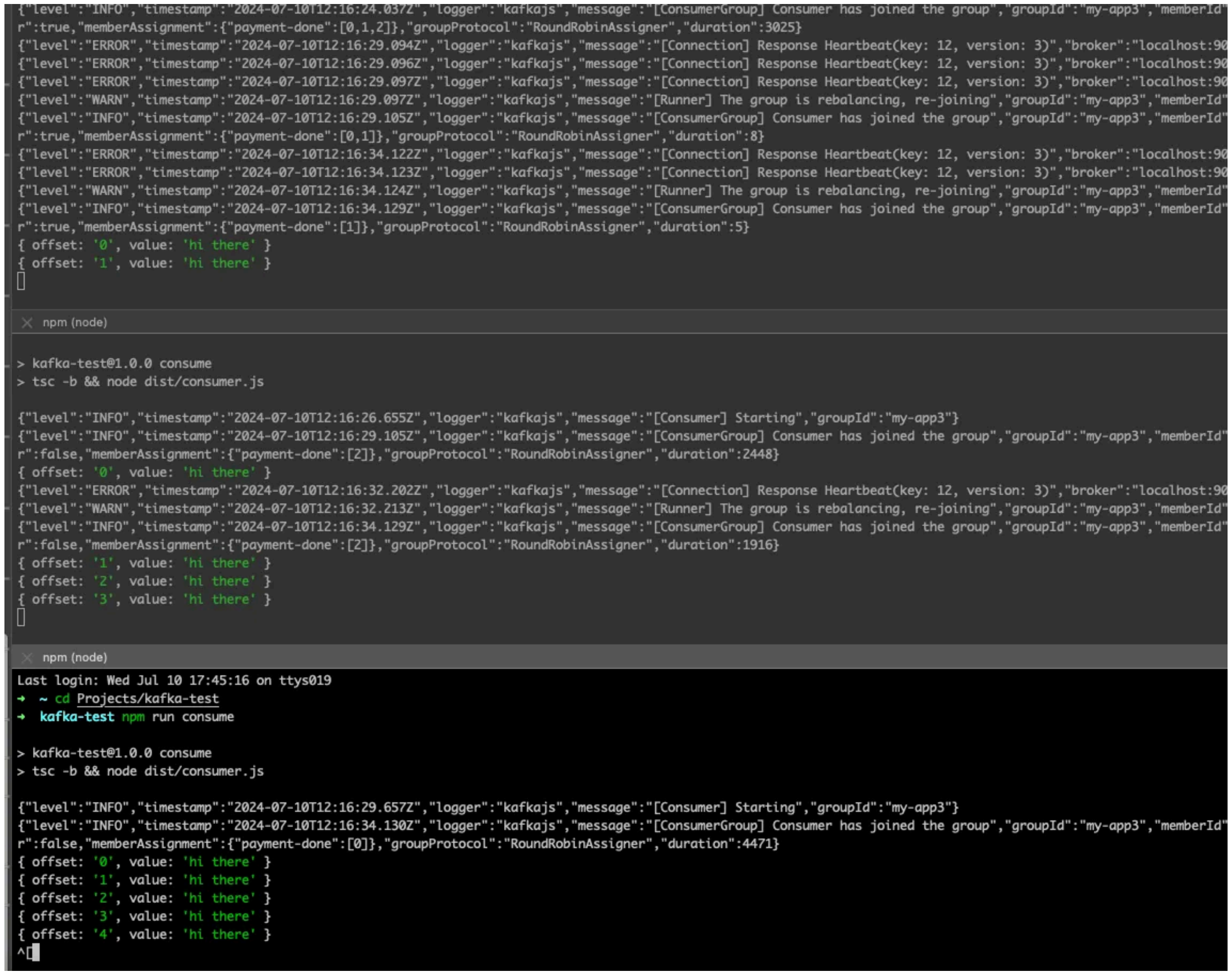
- Consume messages in 3 terminals

```
npm run consume
```

- produce messages

```
npm run produce
```

- Notice the messages get consumed by all 3 consumers

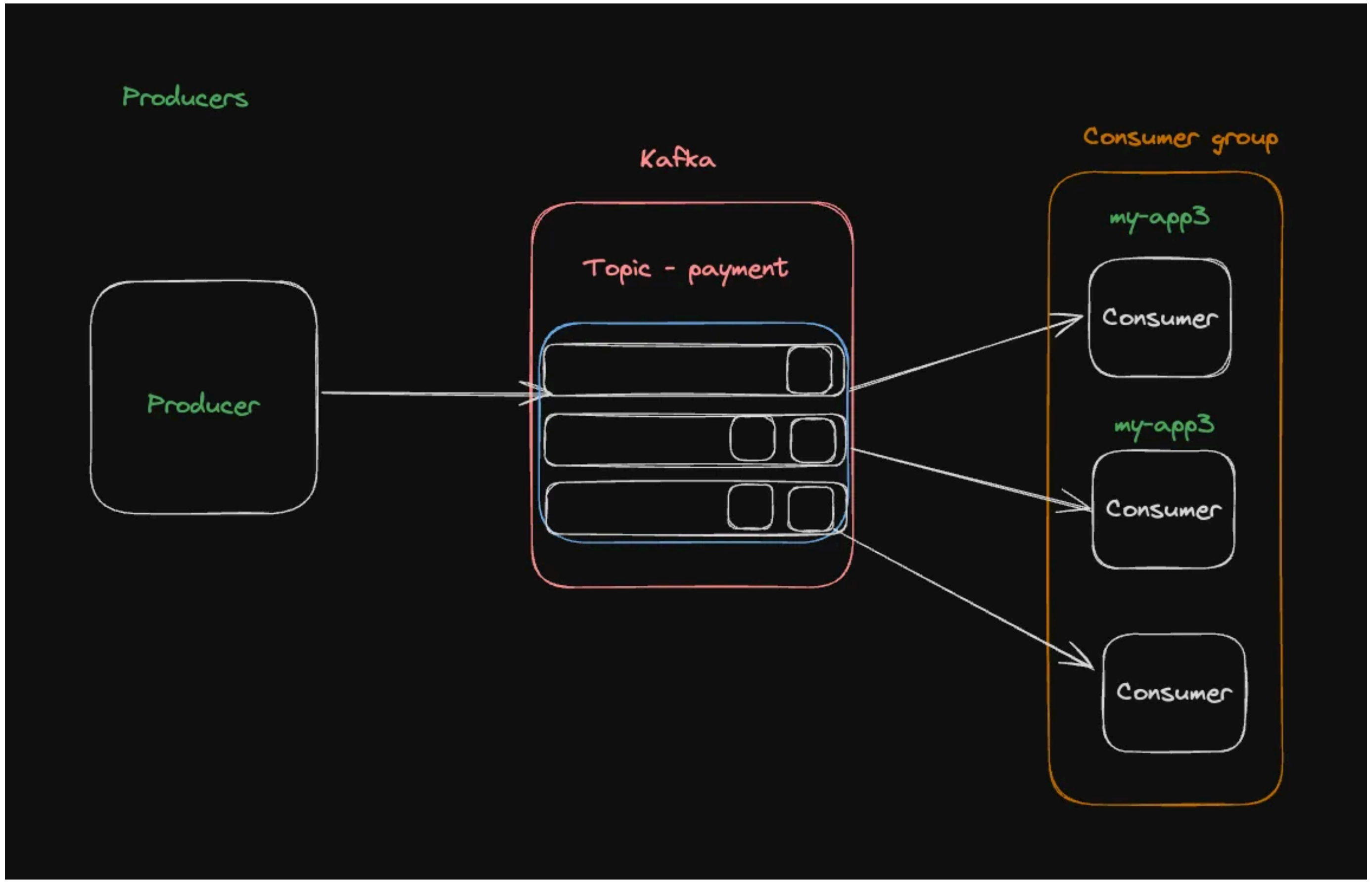


```
{"level":"INFO","timestamp":"2024-07-10T12:16:24.037Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":true,"memberAssignment":{"payment-done":[0,1,2]},"groupProtocol":"RoundRobinAssigner","duration":3025}
{"level":"ERROR","timestamp":"2024-07-10T12:16:29.094Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"ERROR","timestamp":"2024-07-10T12:16:29.096Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"ERROR","timestamp":"2024-07-10T12:16:29.097Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"WARN","timestamp":"2024-07-10T12:16:29.097Z","logger":"kafkajs","message":"[Runner] The group is rebalancing, re-joining","groupId":"my-app3","memberId"
{"level":"INFO","timestamp":"2024-07-10T12:16:29.105Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":true,"memberAssignment":{"payment-done":[0,1]},"groupProtocol":"RoundRobinAssigner","duration":8}
{"level":"ERROR","timestamp":"2024-07-10T12:16:34.122Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"ERROR","timestamp":"2024-07-10T12:16:34.123Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"WARN","timestamp":"2024-07-10T12:16:34.124Z","logger":"kafkajs","message":"[Runner] The group is rebalancing, re-joining","groupId":"my-app3","memberId"
{"level":"INFO","timestamp":"2024-07-10T12:16:34.129Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":true,"memberAssignment":{"payment-done":[1]},"groupProtocol":"RoundRobinAssigner","duration":5}
{ offset: '0', value: 'hi there' }
{ offset: '1', value: 'hi there' }
```

```
> kafka-test@1.0.0 consume
> tsc -b && node dist/consumer.js

{"level":"INFO","timestamp":"2024-07-10T12:16:26.655Z","logger":"kafkajs","message":"[Consumer] Starting","groupId":"my-app3"}
{"level":"INFO","timestamp":"2024-07-10T12:16:29.105Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":false,"memberAssignment":{"payment-done":[2]},"groupProtocol":"RoundRobinAssigner","duration":2448}
{ offset: '0', value: 'hi there' }
{"level":"ERROR","timestamp":"2024-07-10T12:16:32.202Z","logger":"kafkajs","message":"[Connection] Response Heartbeat(key: 12, version: 3)","broker":"localhost:90
{"level":"WARN","timestamp":"2024-07-10T12:16:32.213Z","logger":"kafkajs","message":"[Runner] The group is rebalancing, re-joining","groupId":"my-app3","memberId"
{"level":"INFO","timestamp":"2024-07-10T12:16:34.129Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":false,"memberAssignment":{"payment-done":[2]},"groupProtocol":"RoundRobinAssigner","duration":1916}
{ offset: '1', value: 'hi there' }
{ offset: '2', value: 'hi there' }
{ offset: '3', value: 'hi there' }
```

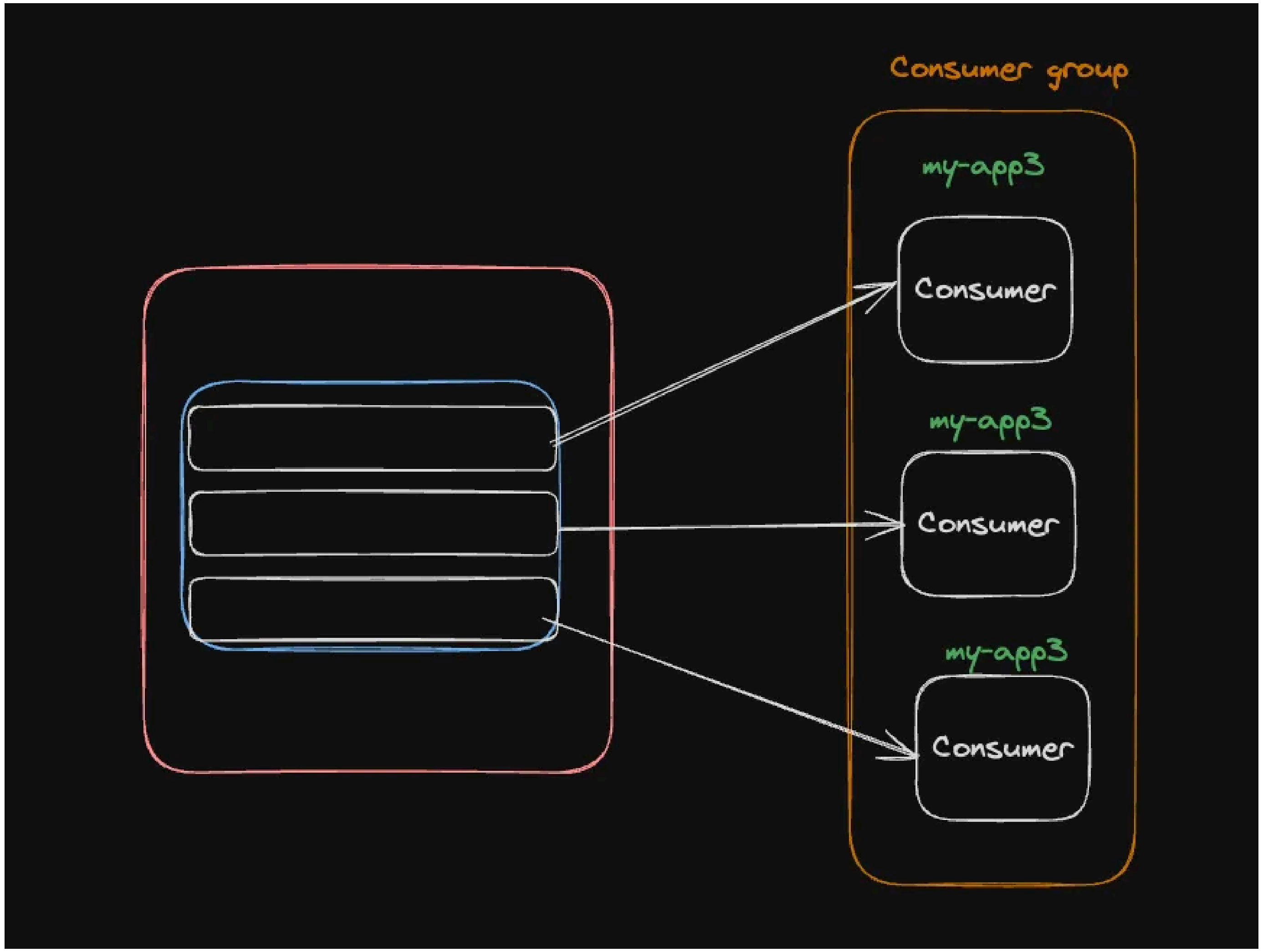
```
Last login: Wed Jul 10 17:45:16 on ttys019
➜  ~ cd Projects/kafka-test
➜  kafka-test npm run consume

> kafka-test@1.0.0 consume
> tsc -b && node dist/consumer.js

{"level":"INFO","timestamp":"2024-07-10T12:16:29.657Z","logger":"kafkajs","message":"[Consumer] Starting","groupId":"my-app3"}
{"level":"INFO","timestamp":"2024-07-10T12:16:34.130Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined the group","groupId":"my-app3","memberId"
r":false,"memberAssignment":{"payment-done":[0]},"groupProtocol":"RoundRobinAssigner","duration":4471}
{ offset: '0', value: 'hi there' }
{ offset: '1', value: 'hi there' }
{ offset: '2', value: 'hi there' }
{ offset: '3', value: 'hi there' }
{ offset: '4', value: 'hi there' }
^
```

Current architecture

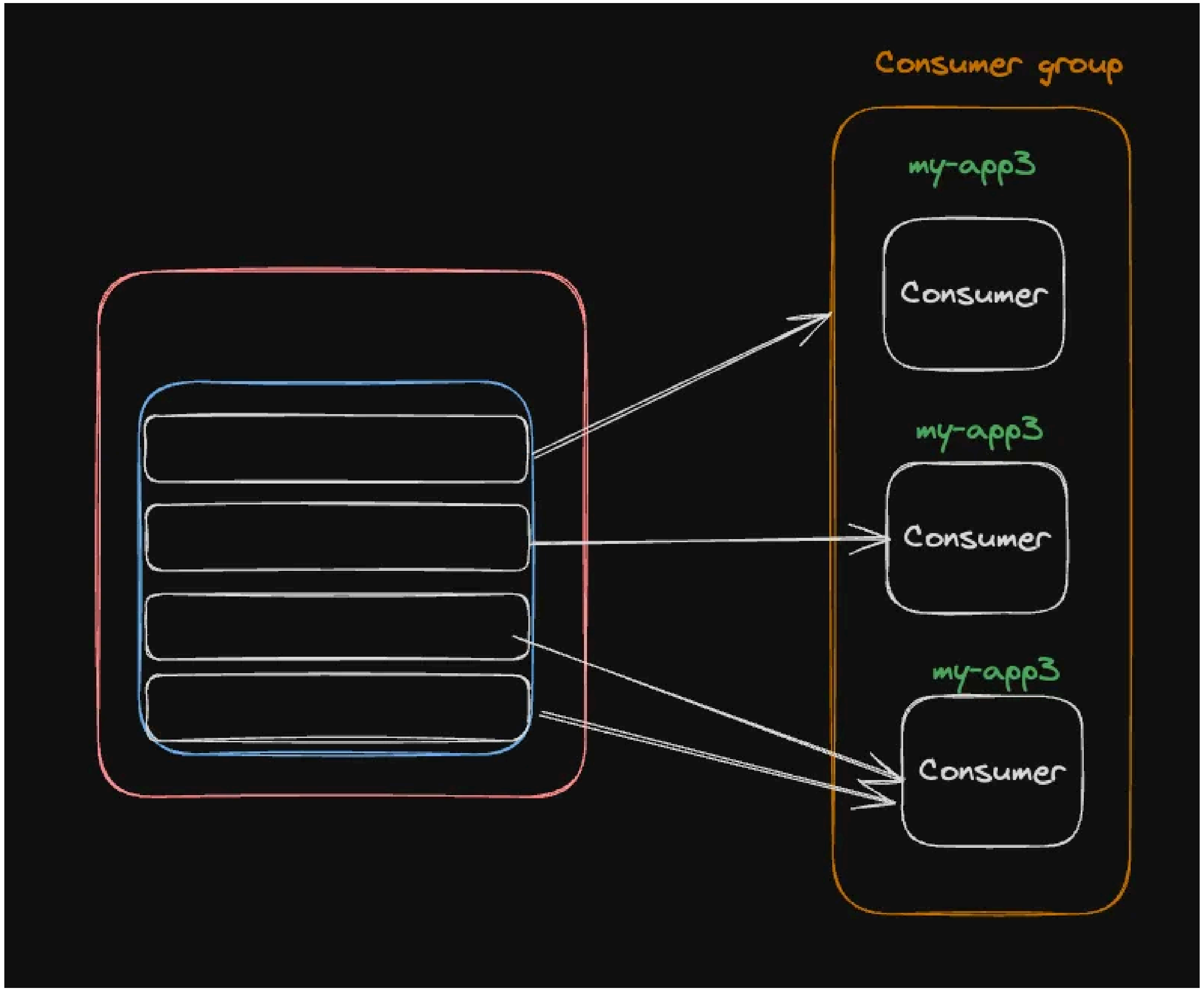


Three cases to discuss

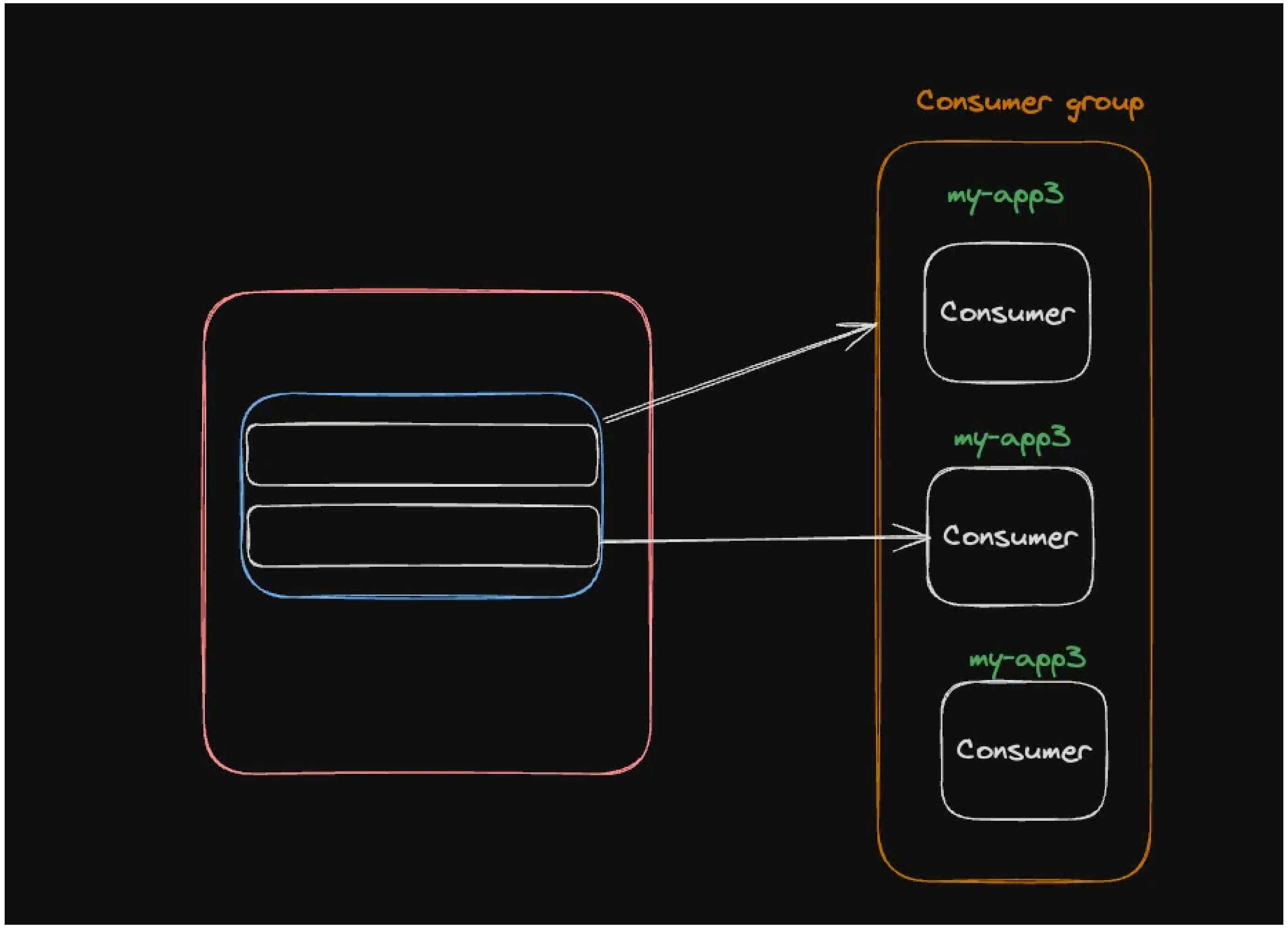
Equal number of partitions and consumers



More partitions



More consumers



Partitioning strategy

When producing messages, you can assign a key that uniquely identifies the event.

Kafka will hash this key and use the hash to determine the partition. This ensures that all messages with the same key (lets say for the same user) are sent to the same partition.

💡 Why would you want messages from the same user to go to the same partition?
Lets say a single user has too many notifications, this way you can make sure they only choke a single partition and not all the partitions

- Create a new `producer-user.ts` file, pass in a `key` when producing the message

```
import { Kafka } from "kafkajs";

const kafka = new Kafka({
  clientId: "my-app",
  brokers: ["localhost:9092"]
})

const producer = kafka.producer();

async function main() {
  await producer.connect();
  await producer.send({
    topic: "payment-done",
    messages: [{
      value: "hi there",
      key: "user1"
    }]
  });
}

main();
```

- Add `produce:user` script

```
"produce:user": "tsc -b && node dist/producer-user.js",
```

- Start 3 consumers and one producer. Notice all messages reach the same consumer

```
npm run produce:user
```