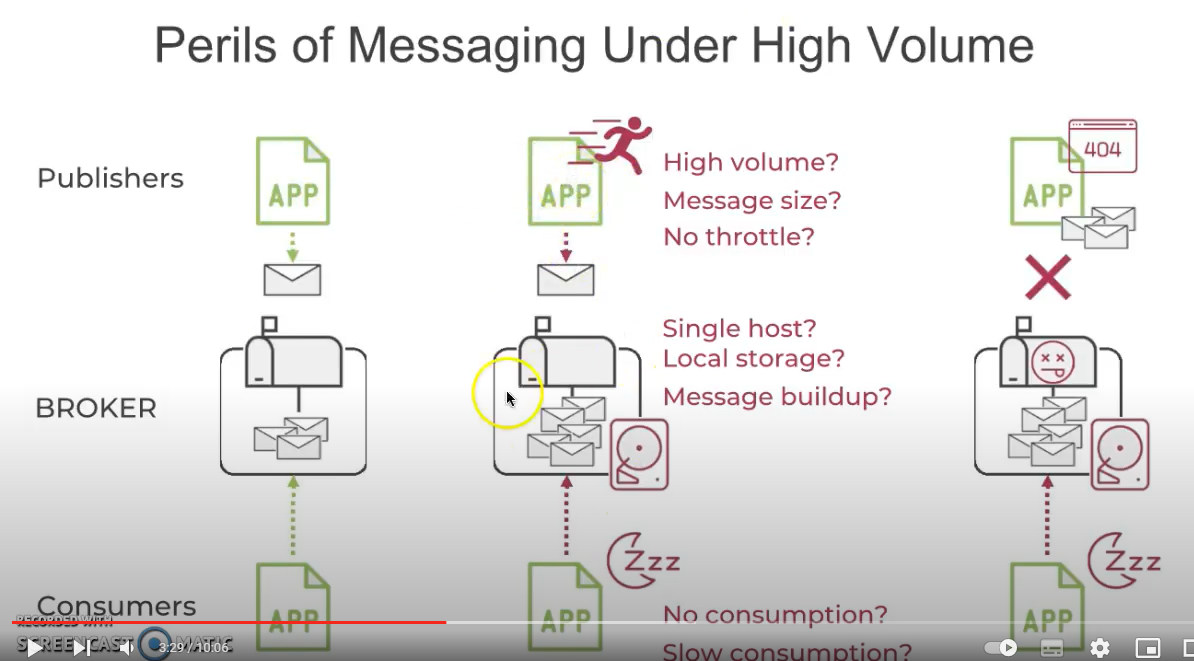


Kafka là gì:

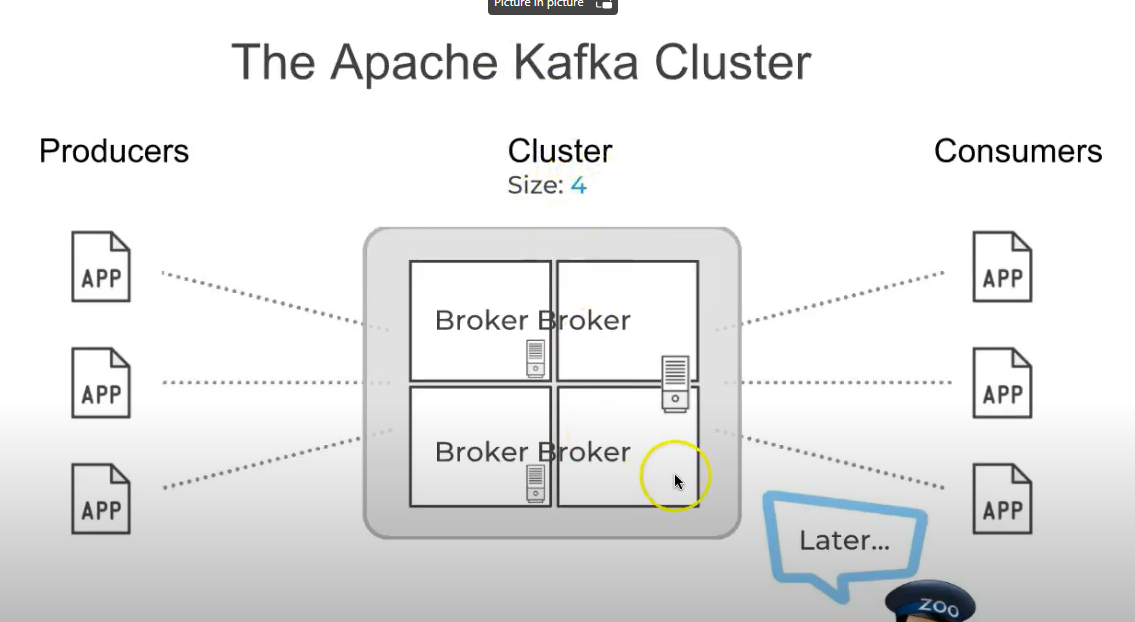


Publishers: người gửi

Brooker: hòm thư

Consumers: người nhận

* TH1: Nếu hệ thống lớn, hòm thư quá tải dẫn tới ng nhận ko thể nhận được thư hoặc hòm thư ko thể tiếp nhận quá trình gửi thư nữa
* TH2: Nếu trong quá trình hòm thư xử lý lỗi, n cần nhận lại thư nhưng thư đó đã bị mất, ko nhận lại dk



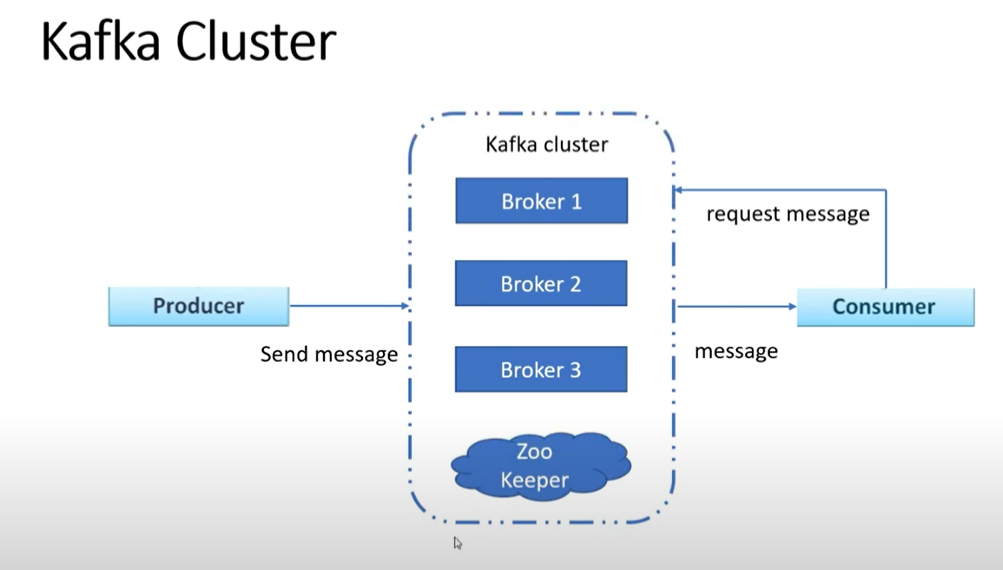
Cụm cluster là apache-kafka cluster

Broker

Apache Zookeeper là nên tảng apache kafka chạy trên đó

# Kafka Components

1. Producer
2. Consumer
3. Broker
4. Cluster
5. Topic



Kafka cluster như 1 văn phòng môi giới để chuyển dữ liệu từ ng gửi đến ng nhận

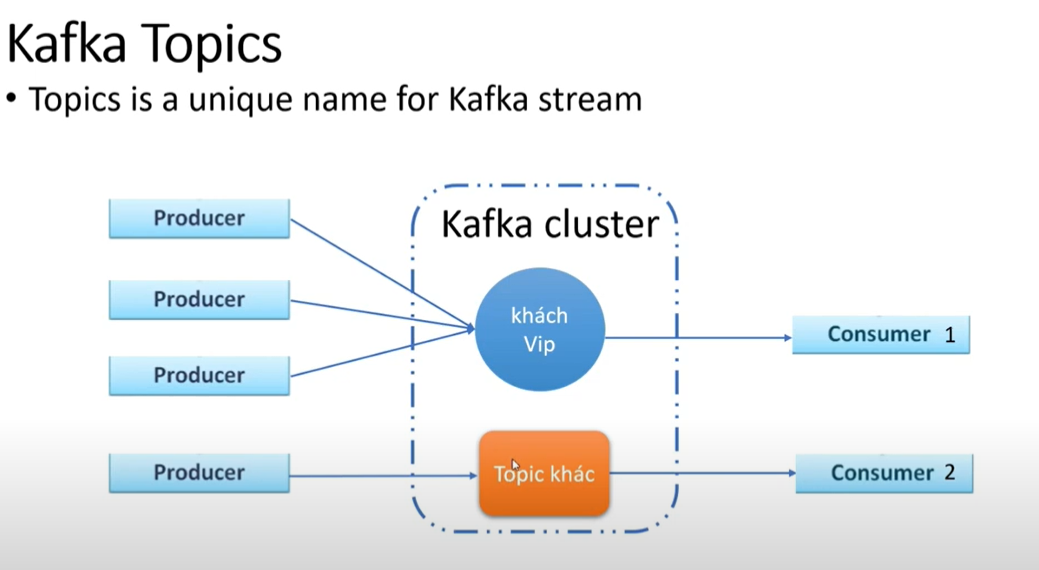
Kafka cluster : văn phòng môi giới

Zoo Keeper: Sếp

Trong 3 broker chỉ có 1 broker hoạt động (gửi nhận từ producer - consumer)

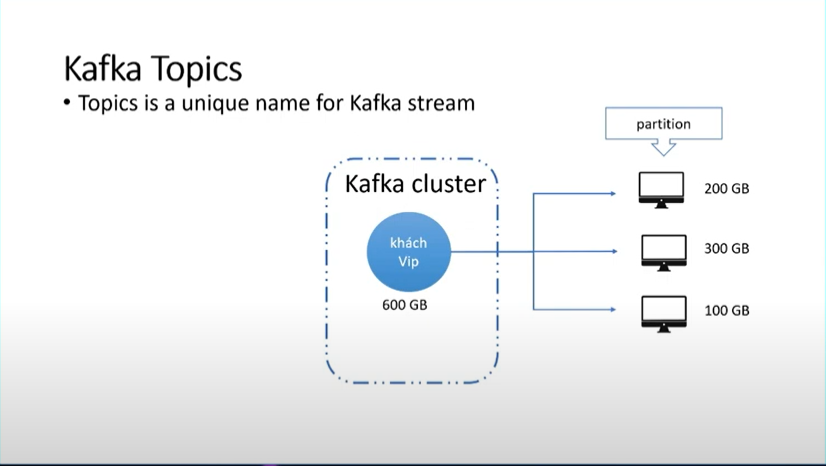
Broker 2 3, sẽ có bản copy để khi broker 1 bị hư thì broker 2, 3 sẽ thay thế

* Vấn đề đặt ra là broker 2 hay broker 3 sẽ thay thế
* Vấn đề này sẽ do thằng Zoo Keeper quyết định

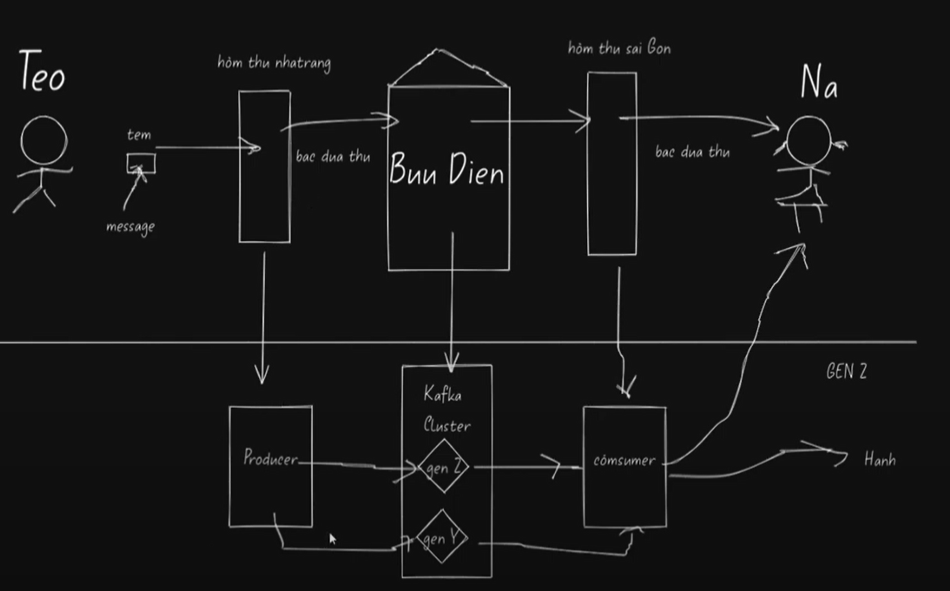


Chia ra các topic để kafka xử lý (nó có thể xét tính năng ưu tiên cho các topic)

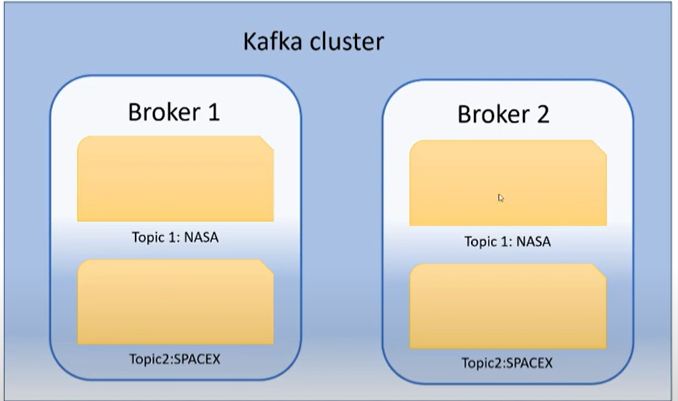
* Producer gửi thông tin cho kafka cluster
* Consumer sẽ nhận thông tin từ kafka cluster
* Trong kafka cluster sẽ có nhiều topic: khách víp, genZ, gen Y



* Partition: nghĩa là khi 1 máy quá nhiều thông tin mà n ko lưu trữ được hết trên 1 máy thì n sẽ chia làm nhiều máy khác nhau gọi là 1 partition



Gen Z, Y là các topic



1 kafka cluster có nhiều broker

1 broker có nhiều topic

Trong topic có nhiều partition

Producer: có nhiều producer

Consumer: có consumer group

# Install kafka

B1: tạo kafka Zoo Keeper vs Broker

Start the ZooKeeper service

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

Start the Kafka broker service

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

* tạo topic

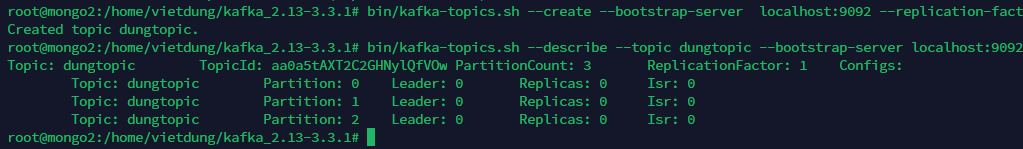
bin/kafka-topics.sh --create --zookeeper localhost:9092 --replication-factor 1 --partitions 3 --topic dungtopic

* lệnh lỗi, đổi thành

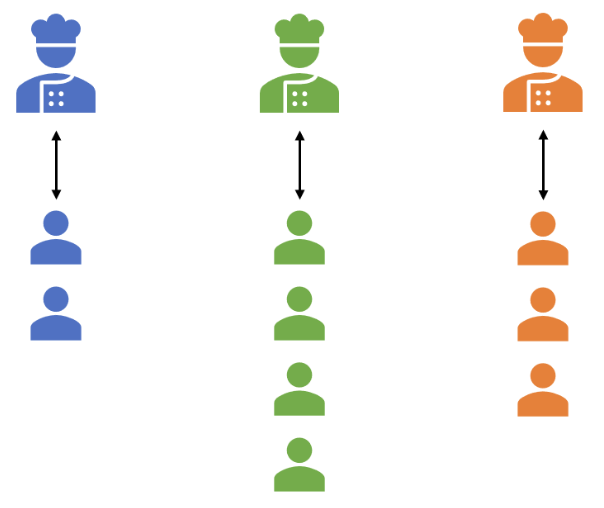
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3 --topic dungtopic

* kiểm tra topic phải có 3 partition ko

bin/kafka-topics.sh --describe --topic dungtopic --bootstrap-server localhost:9092

 oke

# Giải thích ý nghĩa của kafka



Ví dụ khi khách hàng order món ăn (client), và nhân viên là (server)

Thì mỗi client khi order phải đợi nhau

Giải pháp là gì: tăng nhân viên => ko hay, ko ổn

**Vấn đề đang là mỗi người phải xếp hàng đợi nhau, server tèo thì lại phải đợi, chờ server phục hồi**

* sinh ra giải pháp order, gọi món. Khách hàng ko phải chờ đợi việc order, vẫn đi chơi. Còn lại để bọn nhân viên xử lý
* giải pháp là **tách biệt request và response**.

Message-drivenprogramming bắt nguồn từ tư tưởng trên. Server và client không giao tiếp trực tiếp với nhau nữa.(tránh việc client –server: nhân viên và khách hàng đứng tán nhau làm nghẽn server hoặc server chết). Tất cả các request sẽ được gửi dưới dạng message cho bên thứ 3. Bên thứ 3 có nhiệm vụ điều hướng các message đến địa chỉ cụ thể với 2 mục tiêu: