Today's Menu

- **What is Kafka?**
- **Kafka basic concepts**
- **Installing & Running Kafka**
- **Configurations concerns**
- **Launching a cluster**
- **Poking around with the command line**
- **Adding metrics**
- **Adding topics**
- **Top considerations for topic configuration**
- **Simple producing and consuming**
- **Running simple load test**
- **Breaking things and reviving**
- **Creating a dashboard**
- **Troubleshooting**

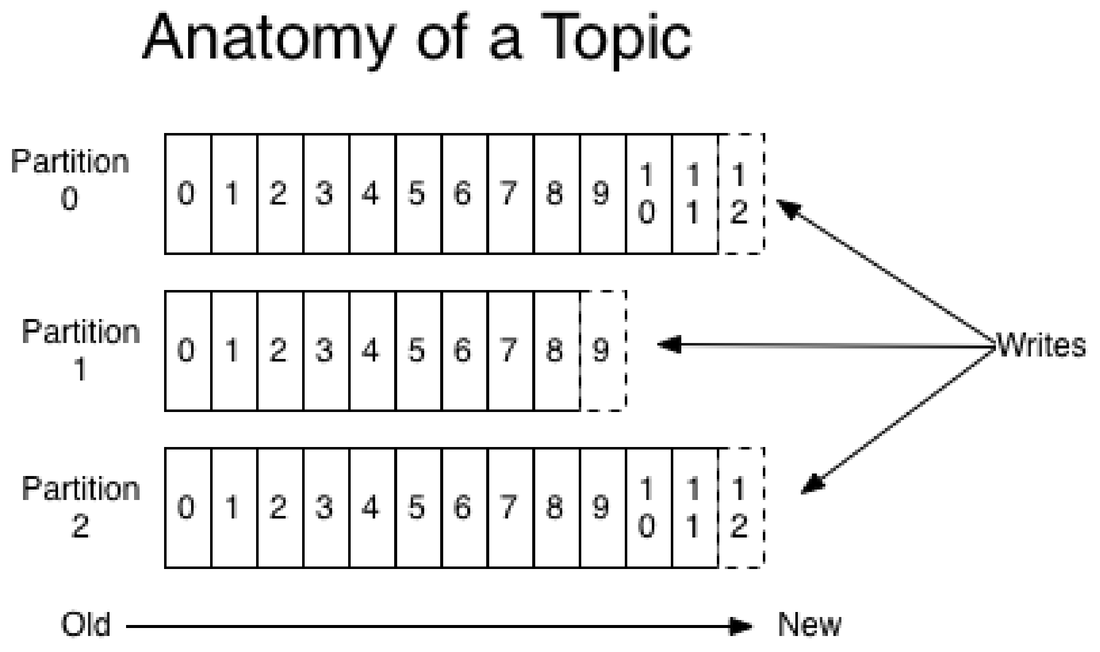
Quick Kafka

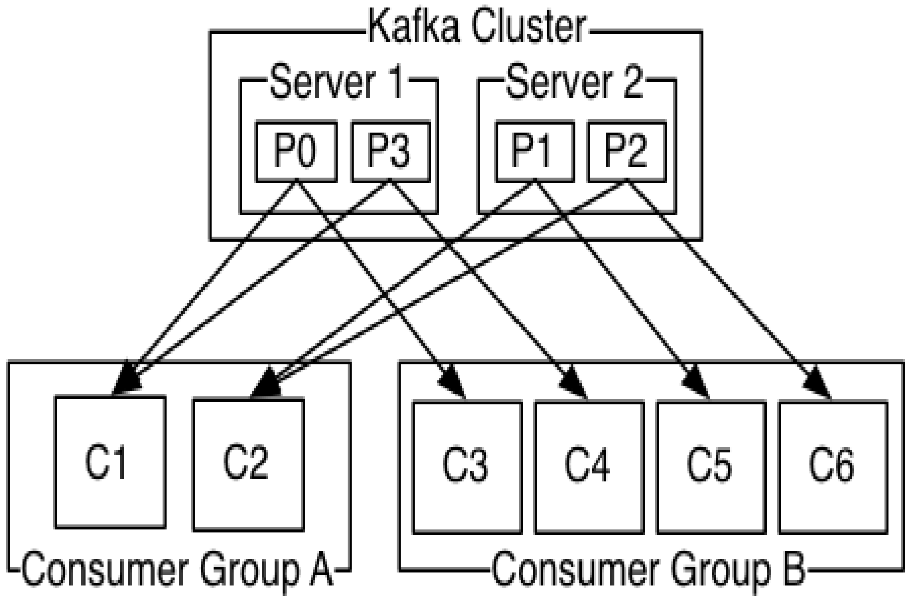
Kafka Overview

"An open source, distributed, partitioned and replicated commit-log based publish - subscribe messaging system"

Kafka Overview

- **Topic:** Category in which messages are published
- **Broker:** Kafka server process (usually one per node)
- **Partitions:** Topics are partitioned, each partition is represented by the ordered immutable sequence of messages. Each message in the partition is assigned a unique ID called offset





Installing and running

Not much of an installation, just download and open a tgz file

Apache:

https://kafka.apache.org/downloads

OR

Confluent:

https://www.confluent.io/download/

Running:

Starting zookeeper: (see example in demo)

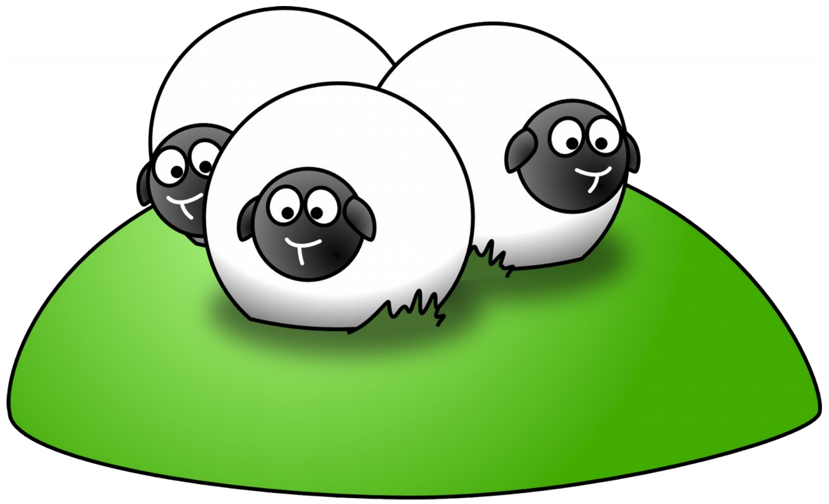
Starting kafka:

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

Configuration concerns

Let's deep dive into the configuration file...

Launching a cluster



Poking around with the command line

Adding Metrics

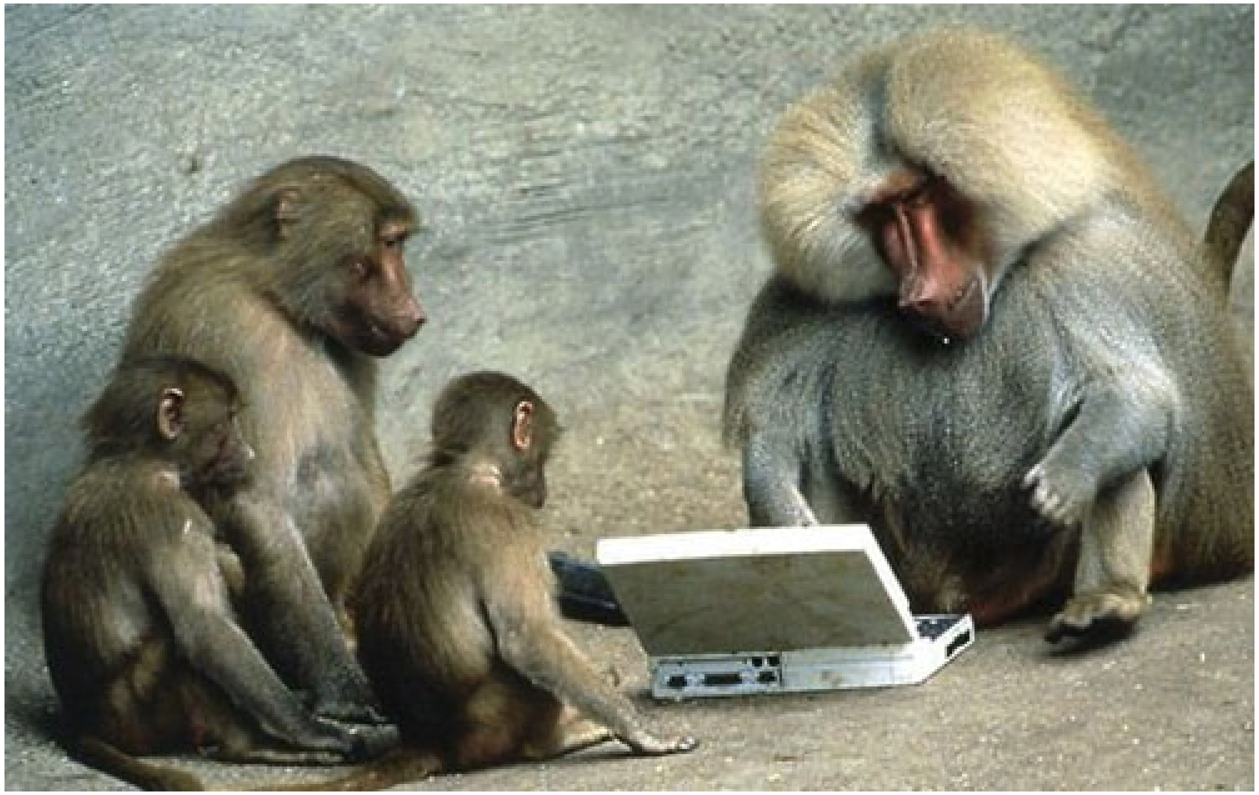
Top consideration for topic configuration

- Minimize replication factor as possible to avoid extra load on the Leaders
- Balance partition number to support parallelism
- Make sure that leaders count is well balanced between brokers
- Retention (time based) should be long enough to recover from failures
- Keep spare disk space to increase retention if needed

Live Demo

- Producing and consuming
- Running simple load test
- Breaking and reviving
- Troubleshooting

Creating a dashboard



THANK YOU

Lessons learned - I

- Minimize replication factor as possible to avoid extra load on the Leaders
- Make sure that leaders count is well balanced between brokers
- Balance partition number to support parallelism
- Split cluster logically considering traffic and business importance
- Retention (time based) should be long enough to recover from failures
- Keep spare disk space to increase retention if needed

Lessons learned - II

- Make sure you are running with adequate FD value (we use 64K)
- In AWS, consider spreading cluster between AZ
- Support cluster dynamic changes by clients
- Create automation for reassign
- Save cluster-reassignment.json of each topic for future needs
- Have enough IOPS for Zookeepers as well
- Check that your client version is compatible with message format

Lessons learned - III

- Careful with adapting new instances
- Automate your cluster migration
- Backup your messages, we use secor
- Monitor: ISR, leader election, Iowait, network bandwidth, messages count, df
- Control brokers recovery
- Take public stress tests with grain of salt