A Guide to Zookeeper, Kafka and Docker


WHO

Bello

Muhammad

ETCBASE

Contents

Zookeeper:

A high-performance coordination service for distributed applications.

Usage:

- Centralized service for distributed systems to a hierarchical key-value store,
- Distributed configuration service,
- Synchronization service, and
- Naming registry for large distributed systems

How to Install Zookeeper:

Download link:
https://zookeeper.apache.org/releases.html#download

$> tar -xzf zookeeper-version.tar

$> cd zookeeper-version

The server can then be run.

Zookeeper Standalone Minimal Configurations:

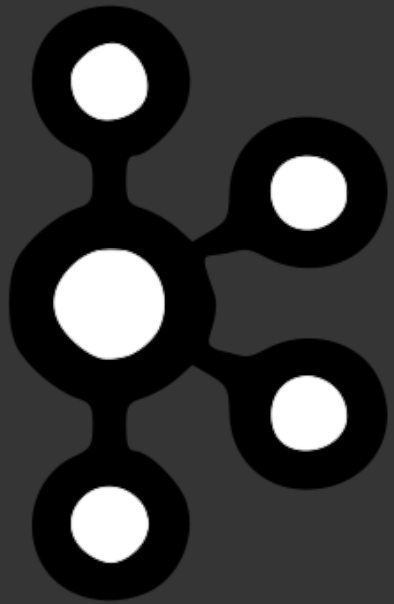
Editing **zoo.cfg** in **config/zoo.cfg**

- ➔ tickTime=2000
- ➔ dataDir=/file/path/fileName
- ➔ clientPort=2181

Apache Kafka:

Apache Kafka is a distributed streaming platform.

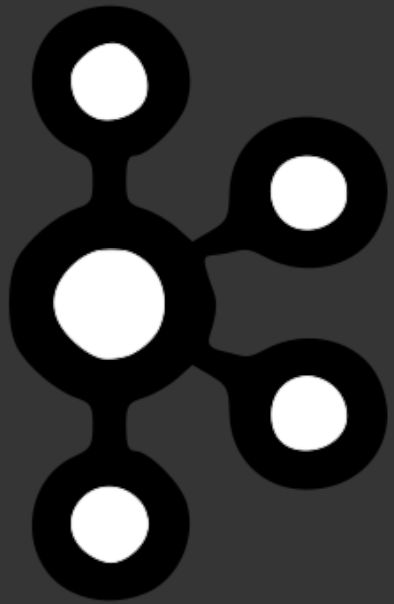
- ➢ Publish and subscribe to streams of records.
- ➢ Store streams of records in a fault-tolerant durable way.
- ➢ Process streams of records as they occur.



Cont.

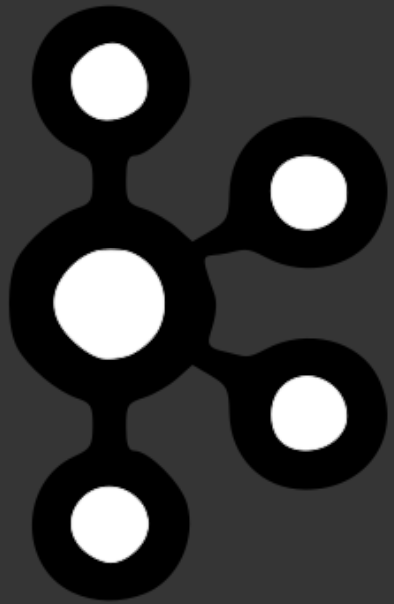
Kafka is generally used for two broad classes of applications:

- Building real-time streaming data pipelines and
- Building real-time streaming applications



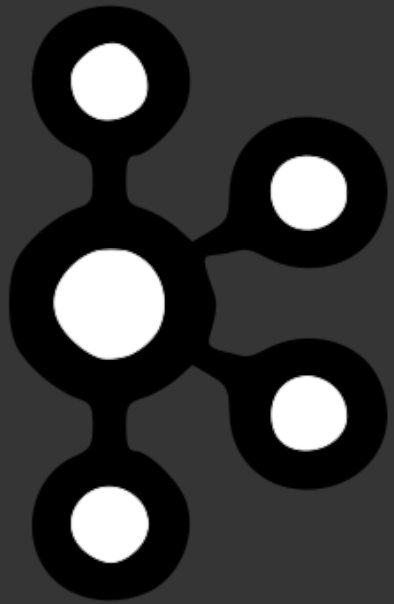
The Concept:

- Kafka is run as a cluster
- *Records* are in categories called *topics*.
- A record has a key, a value, and a timestamp.

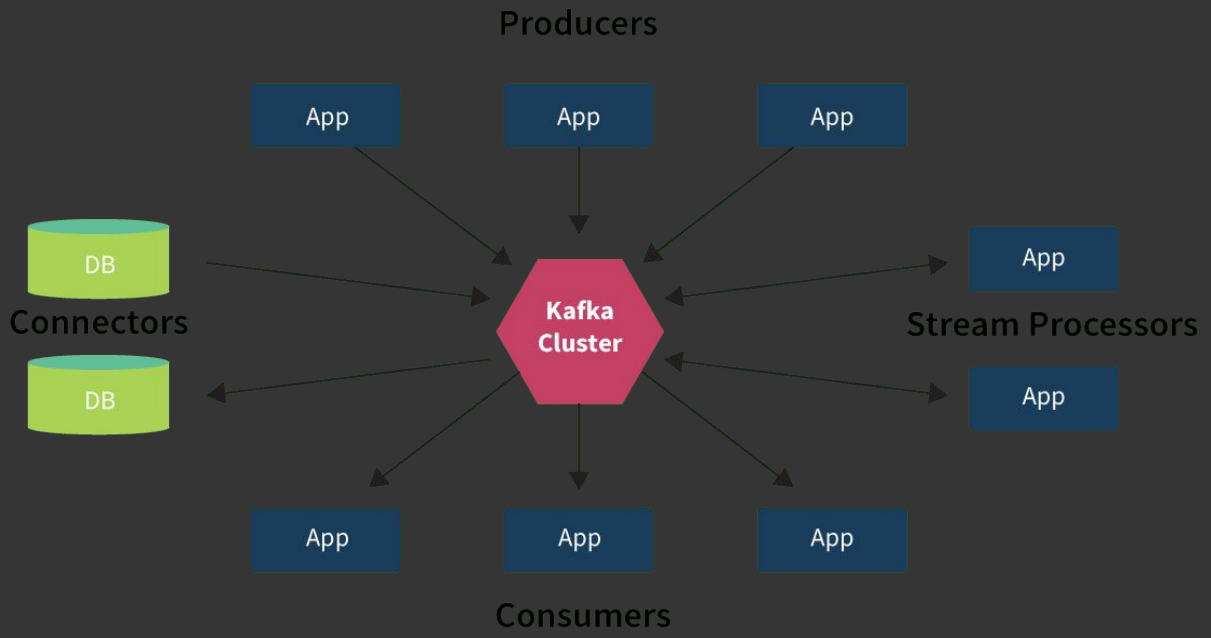


Kafka Core APIs:

- Kafka is run as a cluster
- *Records* are in categories called *topics*.
- A record has a key, a value, and a timestamp.

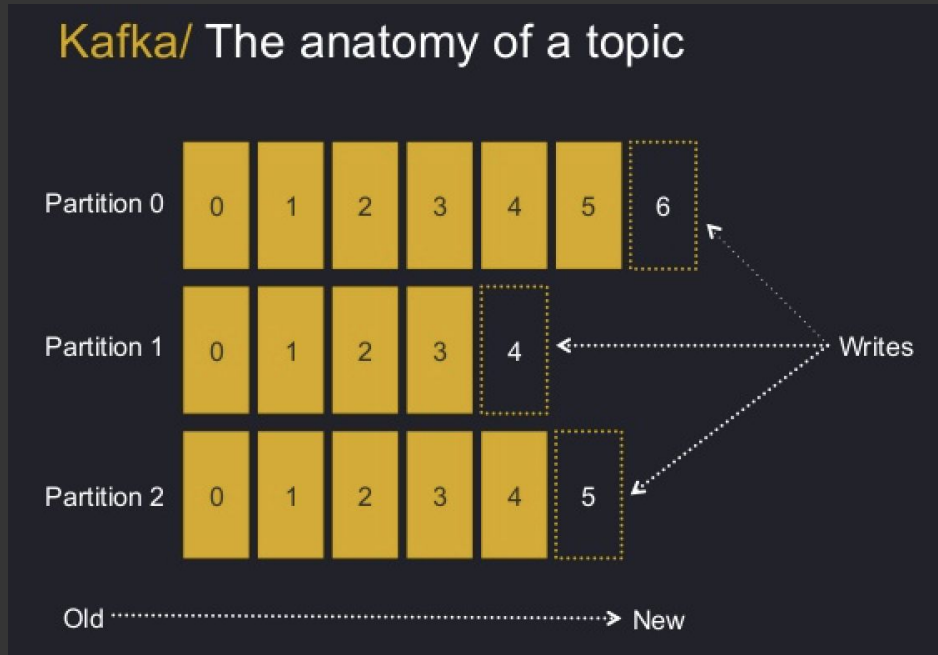


Kafka Core APIs:



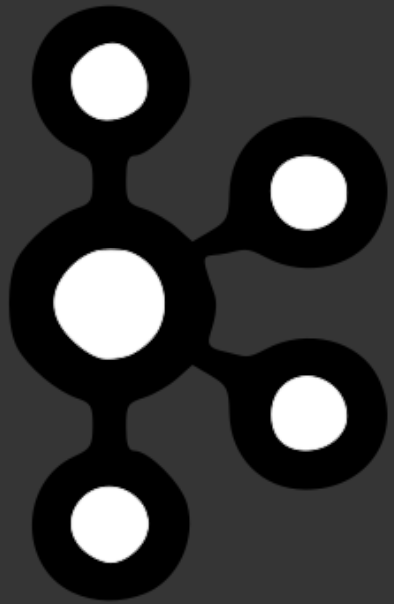
Topics and Logs:

A topic is a category or feed name to which records are published. For each topic, the Kafka cluster maintains a partitioned log that looks like this:



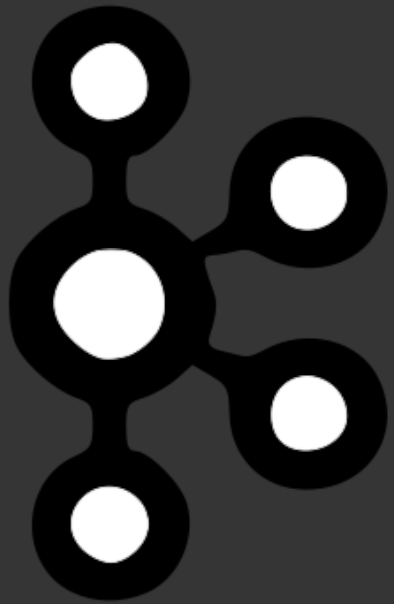
Producers:

Producers publish data to the topics of their choice. The producer is responsible for choosing which record to assign to which partition within the topic.



Consumer:

Consumers label themselves with a *consumer group* name, and each record published to a topic is delivered to one consumer instance within each subscribing consumer group.



Kafka/ How it's used...

The might:

Installing and Starting Kafka

Cont.

Download link:
https://www.apache.org/dyn/closer.cgi?path=/kafka/2.1.0/kafka_2.11-2.1.0.tgz

$> tar -xzf kafka_version.tar

$> cd kafka_version

Starting Kafka:

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

Creating a topic:

Create:

$> ./bin/kafka-topic.sh --create --zookeeper hostName:2181 --replication-factor 1 --partition 1

List Topics:

$> ./bin/kafka-topic.sh --list --zookeeper hostName:2181

Messaging With Console Clients:

Starting console producer client:

$> ./bin/kafka-console-producer.sh --broker-list hostName:9092 --topic topicName

Starting console consumer client:

$> ./bin/kafka-console-consumer.sh --bootstrap-server hostName:9092 --topic topicName --from-beginning

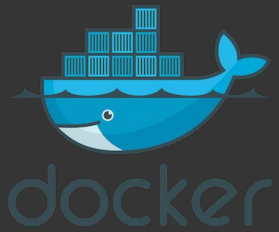

OOO MANYAK LAF

Cont.

Docker is a computer program that performs operating-system-level virtualization.

Docker is a tool designed to make it easier to create, deploy and run applications by using containers.

- Easier to create, deploy and run applications by using containers.
- Docker allows applications to use the same linux kernel.



Docker Container:

$> docker version

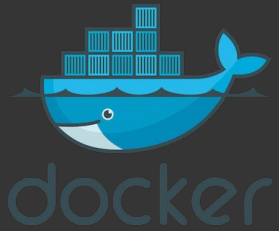
$> docker info

$> docker container run -it -p 8080:8080 tomcat

$> docker container run -d -p 8080:8080 tomcat

$> docker container run -d -p 8080:8080 --name containerName tomcat

$> docker run -d -p 8080:8080 imageName



Container Management:

$> docker container ls (ls -a)

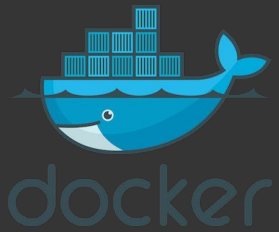
$> docker ps (ps -a)

$> docker container stop containerName/Id

$> docker container rm containerId

$> docker container rm -f containerId

$> docker rm $(docker ps -aq) -f



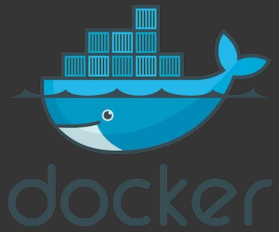
Docker Images:

Docker uses local image or download from docker hub.

$> docker pull imageName(e.g. tomcat)

$> docker images

$> docker image rm imageId

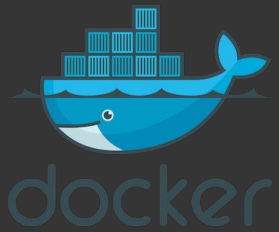


Bashing Container:

After starting a container, we can bash into the container to work with the file systems.

$> docker container exec -it containerName bash

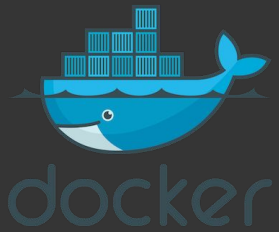
$> exit



Dockerfile:

To create docker image we need to create Dockerfile in the directory where we want to create the image.

```
FROM openjdk:9
ADD target/theJarFileName.jar /where/to/put/the/jar/in/docker/container
EXPOSE 8080
ENTRYPOINT [“java”, “-jar”, “theJarFileName.jar”]
```

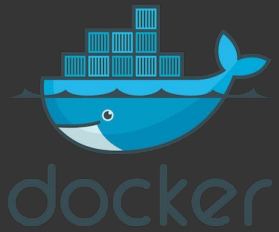


Creating Image:

$> docker image build -t dockerUsername/imageName docker/file/directory (or . means current directory)

$> docker build -f Dockerfile -t imageName .

$> docker push dockerUsername/imageName

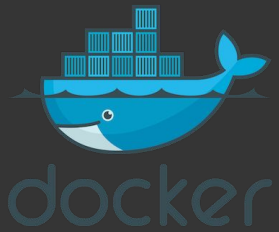


Docker Compose:

Compose is a tool for defining and running multi-container Docker applications.

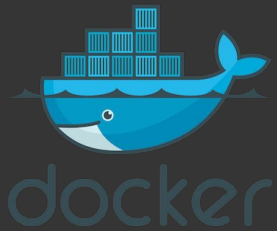
Three Steps to compose:

1. Dockerfile
2. docker-compose.yml
3. docker-compose up (up -d)



– docker-compose.yml:

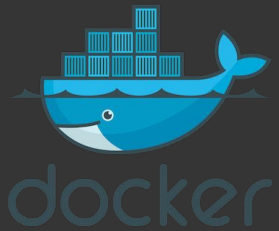
```
version: '3'

services:
 app:
   container_name: my_app_container
   restart: always
   build: .
   ports:
     - '80:80'
   links:
     - mongo
 mongo:
   container_name: mongo
   image: mongo
   ports:
     - '27017:27917'
```



Running Docker Compose:

$> docker-compose up (up -d)

$> docker-compose down



Thank You Very Much.


Q&A