Shared URL → https://tinyurl.com/jp-kafka-Oct25-18

The Shared Location → \\192.168.1.14\Kafka Training

House Keeping

Start → 10.00
First Break → 11.45 -- 15 min
Lunch Break → 1.30 ~ 1.45 -- 45 min
Second Break → 3.45 ~ 4.00 -- 15 min
Wrap up by 6

Welcome to the Kafka Training Module → Java

Instructor Name: Venkat Krishnan [Hadoop, Spark, Kafka]

Hadoop → Batch Style Processing
- Map Reduce
- Hive
- Pig
- Sqoop
- HBase

Spark → Micro Batch Style of processing
- Spark Core
- Spark SQL
- Spark Streaming
- Spark Machine Learning
- Graphx

Kafka → Real Time Processing Engine

The Version of Kafka to download → https://kafka.apache.org/downloads

If we have kafka implementation

P1 P2 P3

Kafka

C1 C2 C3 C4 [Stored 7 days]

What are the Hardware requirements for Kafka →

1) VMWare Workstation
2) Image → Ubuntu Image → Shared with you
3) Other things are given in the software folder

=====> Start our Image and see how things work

In the Image → we have already setup Kafka. Kafka comes in 2 flavors and for people who need the additional capabilities [Avro - SchemaRegistry] , they work on Confluent Kafka.

→ Vanilla Kafka → 2.0
→ Confluent → https://www.confluent.io/download/

Now we will power on the image and see how Kafka is already setup.

1) Start with VMWare workstation
2) Load the image in to VMWare → File Open → Navigate to the folder where the image location is and select a file called as Ubuntu 64-bit.vmx
3) Change the memory and processors [Give it 50% of what you have in your system]
4) To get library window → View menu - Customize - Library Tab
5) Click on Power on the Virtual Machine.
6) UserName: notroot and Password: hadoop123
7) Start with putty →We have to get the IP Address of the image. If it does not give you the IP, we have to power off the image and we have to change the network adapter to NAT. Then power on the image and when we type ifconfig. We should see the IP Now.
8) NAT → Connection to completely new IP Address, If we had said bridged, it would connected to the underlying network IP.
9) There are 4 settings which we have to do in putty → First give the IP and copy that to the Saved Session TextBox and click on Save.
    a) Window - Lines of Scrollback - 9999
    b) Window - Appearance - Change - font to 18
    c) Window - Colors - Click on Use System Colors
    d) Save these settings → Click Session - Click on the IP in the Default Settings and click on Save
10) Install Winscp. → Give the UserName and Password and click on Save. The hidden files will not be shown by default. We will have to configure the same. Options - Preferences - Panels → Click on the checkbox called show hidden files.

==========> Post Morning Tea

Getting Started with Apache Kafka → Ryan Plant

Why Distributed Systems?
→ Fault Tolerant → Data A. → DR [Active - Passive]
Who is responsible for storing the data in 2 places→ manually doing it.

So the plan is to remove the manually way of replication → Automatic.
In Kafka → When a real time data comes in, it is automatically replicated. We have something called as replication-factor.
Distributed System → popular in a environment called hadoop → 2006

Hadoop → Is core Aware → Not

Spark → Oct 14 → Core Aware

Continuous Flow of Data → As and when data comes in, it has to be stored and then processed.

Jay Kreps, Neha Narkhade, Juan Rao → created kafka in Linkedin and now they are a part of confluent which supports kafka in a managed way.

1000 events in 1 min → In Real Time as and when the events comes in i will process it and not store anyways.

Kafka → Real Time Streaming Applications→

1) Language Independence
2) Topic → This is a unit which stores message in Kafka
    a) replication factor →
    b) partition → The data will be stored in multiple partitions.
    Topic A → RF-2 and P-2
    One
    Two

Cluster A:	B1	B2	B3	B4
	Original-1	Replicated	Original-2	Original-3
	One	One		
		Two	Two	

https://kafka.apache.org/powered-by

Downloads → All the basic tar files

Lab

data → source files to process
hdfs →
programs → user defined programs
software → extracted copies of the tar file

==========> extract the tar files

tar -xvf locationofthebasictar file
.bashrc → setup file → set Location of the software + Set the path

==========> Post Afternoon Lunch

Exercise 1 → Hands on → Single Kafka Server → Default Producer and Consumer.

zookeeper
kafka server
topic
    Producer
    Consumer

https://zookeeper.apache.org/doc/r3.1.2/zookeeperOver.html

Power off the System → VM - Power - Power Off
Close putty and Winscp
        Our old System is dead now
Copy the Image from the Shared Location [Software folder]
    Extract the image inside the Software folder
In the VM do the following

File - Open - Navigate to the software folder /image location → select a file called Ubuntu 64-bit → Power on the System
Putty

In a document called 9Kafka_Server_Defaultconfig.pdf we have all the default settings of the Kafka Server.

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

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic first

bin/kafka-topics.sh --list --zookeeper localhost:2181

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic first

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic first

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic first --from-beginning

==========> Post Evening Tea

Ensure that you kill the consumers and the server first before killing the producer and the zookeeper server

Second Example: 2 Broker node.

notroot@ubuntu:~/lab/software/kafka_2.11-1.1.0$ bin/kafka-topics.sh --zookeeper localhost:2181 --describe --topic my-replicated-topic

Topic:my-replicated-topic PartitionCount:1 ReplicationFactor:2 Configs:
Topic: my-replicated-topic Partition: 0 Leader: 1 Replicas: 1,2 Isr: 1,2

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

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 1 --topic my-replicated-topic

bin/kafka-topics.sh --zookeeper localhost:2181 --describe --topic my-replicated-topic

bin/kafka-console-producer.sh --broker-list localhost:9093 --topic my-replicated-topic

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic my-replicated-topic

bin/kafka-server-start.sh config/server2.properties

==========> Custom Producer and Custom Consumer

Restart Consumer

Start Producer where you have server1.properties files.

bin/kafka-topics.sh -create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic topic1

java -cp target/Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleProducer topic1

java -cp target/Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleConsumer topic1 group1

Day 2

Stuff covered on Day 1

1. What is Kafka
2. Setting up the environment for Kafka - Ubuntu 14.4 and the Image
3. Components in Kafka → Topic, partitions, Replication Factor
4. First Example of setting up a Single Node Cluster → start with zookeeper, Kafka server, Setting up a topic, starting with Producer and then start with Consumer. → To read from beginning - from-beginning as an attribute. Storage of the messages → log
5. Second Example → Setting up 2 brokers → server.properties file [broker id, location of the log, listener]. We will create 2 separate server.properties files.
6. Third Example → How to create the Maven project and the complete structure will get move to programs dir. mvn clean and mvn install.

==========> Plan for today

1) Custom Partitioner
2) Custom Serializer → How to do without Avro
3) Monitoring in Kafka
4) How to manage the Offset and to and read a particular Offset
5) Working with Kafka → Confluent API

Examples can be done in 2 ways

1) Copy the Kafka folder to the programs dir and then running via mvn clean and mvn install
2) Compiling in eclipse and copying only the jar file and executing the tar file

Try the second approach of running Kafka Codes.

==========>

Perhaps you are running on a JRE rather than a JDK?
Ensure that you have JDK in the installed JREs and in the lap it was already configured.

Window - Preferences - Java - Installed JREs - Add - Standard VM and then point to the JDK folder.

==========> Then move the jar file - with-dependencies to the programs dir and then execute the same.

java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleProducer topic1

java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleConsumer topic1 group1

Note we can create a consumer group and only 1 consumer out of the group will receive the messages.

Why a consumer group? Scale it as multiple producers are writing to the topic. Why should i create a consumer group? Scalability

3 Producers writing to a topic.

1 Consumer → Read from a topic which is put by all the 3 producers.
2 Consumer → Revamp and it will have to start a new restart → It will call the

ConsumerRebalanceListener(). And the partitions will reassigned, and 1 consumer will get 1 producer and the second consumer will get 2nd and 3rd producer.

3 Consumer → Again there will be a restart and each of the consumers will get 1 producer.
4th Consumer → Again there will be a restart and the ConsumerRebalanceListerner will be called? One of the consumers will be idle.

How many consumers can be a part of the consumer group if we have 3 producers putting messages?

Partitioner is a class which is responsible for deciding which entity is going to receive which message.

Retail Site → India, USA, Else

How to run the Forth Example

1) Your zookeeper and kafka servers are up and running
2) Ensure that your topic is created

   bin/kafka-topics.sh --list --zookeeper localhost:2181
   bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic part-demo1

3) Now create the Producer →
   java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleProducer part-demo1
4) Now create a single Consumer → Which will have all the 3 partitions.
   java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SimpleConsumer part-demo1 group1

   Now you will see that all the 3 partitions will be received by this consumer.

5) Now we will create the second consumer

   At the beginning, you will see that it will call the ConsumerRebalanceListener and the methods will be printed out.

Example 5: Custom Serialization → We are creating our code for Serialization.

Serialization -> saving the state of the object in to some type of persistence.

There are various ways of serialization:

textFile - Binary in Kafka
Avro - Standard way of serialization when there is a change in the code,

When we write our own serialization

1) POJO class - Supplier.java
2) Producer Class → SupplierProducer.class
3) Custom Serialization Logic
4) Deserialization Logic
5) Consumer Class → Custom Deserialization Class

   java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SupplierProducer

   java -cp Kafka3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.jpmc.SupplierConsumer

   Note that we start with the SupplierConsumer first and it will waiting. We will then start with the SupplierProducer and it will Producer the message and it will be shown on the SupplierConsumer Window.

==========>