<https://tinyurl.com/citi-kafka-oct19>

House Keeping stuff

a) Hard Start at 9.30

b) First Break - Tea - 11.30 → 15 min again

c) Second Break - lunch → 1.30 ~ 1.45 → 45 min

d) Third Break → 3.45 ~ 4.00 → 15 min

The Shared Location would be:-\\192.168.26.169\d\Kafka\Materials

Topics that would be covered in this module → Apache Storm and Apache Kafka

1. Introduction to Storm
2. Installation and configuration
3. Storm concepts and structure
4. Bolt /Spout/Data Ingestion in Storm
5. Apache Kafka Introduction
6. Produces
7. Consumer
8. Kafka Internals

# Day 1

What is available for you

a) Image in which we have Apache Storm and Apache Kafka already setup

b) Putty →

c) WinScp

When we talk about real time processing

a) Apache Storm

b) Apache Kafka

Why Kafka

a) The earlier systems [ TIBCO ] → Commercial tool v/s open source tool like kafka

b) Kafka stores the message for a longer period of time → default is 7 days

Citi - use technologies like

A) **Hadoop**

Map Reduce

Hive → SQL way of processing data

Pig → Scripting Language

Sqoop → Moving Data from RDBMS to HDFS

HBase → NoSQL database → Columnar

Flume → Source → Sink → destination → Moving data into HDFS

B) **Spark** → 2.3.0

Spark Core → Transformations & Actions

Spark SQL → SQL style of querying + DataFrames

Spark Streaming → Micro Batch [ More than 60 sec ]

→ Spark Machine Learning

→ Spark Graphx

C) **Storm & Kafka** → Real time Processing of data

========================>

If you do not see the Left Hand side [ library tab ], you can see that via View menu - Customize - Library

In the image we already have Kafka and Storm setup and we only have to start it.

We will click on File - Open [From the VMware] and go down to D:\Kafka\Kafka\_Ubuntu14.4\_Version2\Kafka\_Ubuntu14.4\_Version2 → Ubuntu64-bit file which we will have to open.

What we will have to do open the image is loaded in VMware? This will be in the Edit Virtual Machine Setting option.

a) Ideally we can keep the image ½ of what memory we have.

b) Processors → we have to give ½ of what we have in the image. That is why we have given 2 cores, whereas we have i5 in our systems [ which means we have 4 cores ]

c) Network Adapter → Changed it to NAT. If we use Bridged, then we will have the IP of the underlying Office.

The first time when you open the system →

a) I copied it

b) No for IDE:0

===============> Now the system has opened up.

Once we click inside, we will lose our mouse and to get it back we will have to click on Ctrl + Alt.

The credentials for the image

UserName: notroot

Password: hadoop123

Now we will have to start with putty and winscp

a) do ifconfig so that we get our ip and then type the Internet Protocol in the HostName Field and copy the same in the Saved Session and then click on save.

b) Window - Lines of Scrollback → 9999

c) Appearance - Font → Change it a level you want.

d) Colors → click on the checkbox called use System colors → so that we get a white background

e) Click on the Sessions menu → click on the saved session - save

Now we will install winscp in our system →

At the last option it will say do we need to use the IP which we have configured earlier. YES.

Then click on edit so that we can type our username and password and then click on the drop down in the save and click on save password.

To enable the hidden files → Options Menu in winscp - preference tab - panels tab → click on the first checkbox called as Show Hidden Files

Flow of topics that we will do today

a) Theory of Apache Storm → Spout (source of data ) and Bolts (processing components ]. We will talk about the architecture of storm.

b) Single Node Cluster of Storm

→ Zookeeper

→ Master → Nimbus

→ Slave →

→ Java Example of how real time processing happens in storm

c) Multi Node Cluster

**Folder Structure in our /home/notroot**

downloads → This will have all the tar files

lab → Have seperate sub directories for different components

a) data → Landing zone for all user defined data

b) hdfs → location where hadoop components will be there → NameNode and DataNode will be present

c) Programs → this is where all the user defined programs will be present.

d) Software → This is where the extracted copies of the software would be present.

=========================>

When we have data coming in for us, it can come in various ways

a) Batch Data → Hadoop

b) Micro Batch Data → Spark Streaming

c) Real Time Data → Apache Storm, Apache Flink

Distributed? Processing will be distributed and the data will be spread across these systems. Both data and the processing will be distributed.

Storm is Stateless → That means it will not keep any data with itself and might end up persisting the data in a database or HDFS.

At Least once v/s Atmost once

Cloud Based

a) IAAS - PAAS

b) SAAS → AWS, Azure, GCP

If you want to have an example of customers of zookeeper →

<https://cwiki.apache.org/confluence/display/ZOOKEEPER/PoweredBy>

<https://zookeeper.apache.org/doc/current/zookeeperOver.html>

Post Morning Break

<https://storm.apache.org/Powered-By.html>

The best benefit of cloud processing is the ability to do upscaling and downscaling of components and resources.

The way how we will do it in our system → Pseudo Cluster Deployment

Start Process → Zookeeper

Start Process → Nimbus

Start Process → Supervisor

Completed the written test.

Let us start with the Hands on

a) Copy the tar files to downloads directory

b) Open a document → ApacheStorm1.0.1\_Setup.pdf

c) go to lab/software directory → tar -xvf ../../downloads/TarFileName → Do this for Storm and for Zookeeper also

d) .bashrc → keeping the environmental variables

notroot@ubuntu:~/lab/software$ tar -xvf ../../downloads/apache-storm-1.0.1.tar.gz

notroot@ubuntu:~/lab/software$ tar -xvf ../../downloads/zookeeper-3.4.6.tar.gz

export STORM\_HOME=/home/notroot/lab/software/apache-storm-1.0.1

export PATH=$PATH:$STORM\_HOME/bin

export ZK\_HOME=/home/notroot/lab/software/zookeeper-3.4.6

export PATH=$PATH:$ZK\_HOME/bin

Ensure that you save the file and then execute → . .bashrc

notroot@ubuntu:~$ . .bashrc

notroot@ubuntu:~$

notroot@ubuntu:~$ zkServer.sh start

JMX enabled by default

Using config: /home/notroot/lab/software/zookeeper-3.4.6/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

notroot@ubuntu:~$ jps

1713 Jps

1694 QuorumPeerMain

notroot@ubuntu:~$

If you have reached till here, we have our zookeeper started

=====================> **Post Lunch**

Configurations can be done in multiple ways

a) properties

b) xml file

c) YAML → Yet another markup language

Max memory used by nimbus → 1024m

notroot@ubuntu:~$ jps

1406 QuorumPeerMain

1444 **nimbus**

1555 Jps

Max memory used by supervisor → 256m

notroot@ubuntu:~$ jps

1567 **supervisor**

1406 QuorumPeerMain

1444 nimbus

1662 Jps

Max memory used by UI → 768m

notroot@ubuntu:~$ jps

1567 supervisor

1406 QuorumPeerMain

1444 nimbus

1765 Jps

1675 **core**

Now we have everything running→ ensure that we have 3 daemons started →

jps → Java Process status viewer tool

Now let us do a simple hands on → Word Count of the words that we write in the console.

Read from the console →

a) Spout → read one line at a time.

b) Bolt → split based on spaces

c) Bolt → What is its responsibility → Print the status of the word count to the console.

d) Driver program → main program which will execute the flow of the topology. It creates a topology and submits the topology to the cluster.

Eclipse is needed here and extract this from the Kafka Directory.

Start eclipse and in the first step point it to a new folder:- D:\Kafka\workspace

Create a new perspective with java

Create a new project called as Storm1

Create a new package called com.citi

Drag and drop all the 4 .java files inside com.citi

Open each .java file and change the package name

Create a new folder in Kafka called stormjars and copy all the jars via winscp from the lib directory to the storm jars directory.

Keep the stormjars as the add external jars directory. The steps are

Right click on the project

Choose properties

Java Build Path

Click on libraries tab

Add External Jars

Select all the jars and click ok

How are you

I am fine

How about you

How about you

<http://www.corejavaguru.com/bigdata/storm/stream-groupings> → Right now we are using only 1 instance of a class and hence using HashMap but in a real world scenario we should use different types of grouping.

Describing the Fields class in the declareOutputFields method :-

<https://stackoverflow.com/questions/34514168/what-is-the-purpose-of-fields-class-in-apache-storm>

Now we need to stop the existing process and start with our 4 nodes.

So please close your existing code.

a) Kill your nimbus, supervisor and core → kill -9 PID

b) zkServer.sh stop

c) jps to ensure that nothing is running.

Then we need to keep our environment ready for the multi node cluster setup

a) Copy our Kafka\_Ubuntu14.4\_Version2.rar file [ This will be our image file ] into 4 separate folder which we will call it as Storm1, Storm2, Storm3 and Storm4 folder

b) extract each of the rar file in to the folders.

This will be the basic step that we will have to do for the multi node.

======================> **Post Evening Tea Break**

1) Power off the Putty and winscp

2) Power off the Original image [ In which we worked with a Pseudo cluster of storm ] → VM Menu - Power - Power Off

3) Now we will load the 4 images which we had and after loading the first image, we will change the name to Storm1 and so on and so forth.

4) In each of the image → Change the network adapter to NAT

5) Power on all the 4 systems → Take their IPs and change it in the Storm\_MultiNode.txt

6) Then we will change the /etc/hosts and write down all the 4 IPs there.

Now we have to do this only on Storm1 System

1) extract the tar file

2) change the .bashrc

3) change the zoo\_sample.cfg to zoo.cfg

notroot@ubuntu:~/lab/software/zookeeper-3.4.6/conf$ mv zoo\_sample.cfg zoo.cfg

4) Modify the contents of zoo.cfg

Note that when we are changing the storm.yaml file the zookeeper name should be like this :-

storm.zookeeper.servers:

- "zk1"

Pre-Req for tomorrow:- https://www.youtube.com/watch?v=I32hmY4diFY

# Day 2

Stuff done on Day 1

1) What is Apache Storm and its architecture

2) Pseudo Cluster

3) Simple example to prove how real time it works

4) Multi Node Cluster

Plan for today

1) What is Apache Kafka

2) Exercise 1 → Single Broker

3) Exercise 2 → Multi Broker

4) Exercise 3 → Custom Producer and Consumer →

a) How to run this with Maven

b) How to work with a jar file

5) Partitioner

6) Custom Serializer →

7) Confluent →

a) Simple Example

b) Avro Serialization

We will not be covering

→ KSQL

→ KStream

We will keep our system ready

a) We will power on the Default image and Nothing to do with the 4 storm instances we created as that was for Spark Multi Node Cluster

b) Start with Winscp

c) Start with putty

Zookeeper was a key component in kafka till version 0.8 and from 0.9 they have limited the responsibility of zookeeper and doing everything that the zookeeper does with the Kafka Server itself.

While we are scaling up, the zookeeper became a bottle neck and hence we are reducing our dependence on that.

Step 1 to 9 → Single Node Cluster →

The kafka documentation default link: <https://kafka.apache.org/documentation/>

Step 9 to 13 → Second part in Example 1

Now we will work with Example 2 → Where we are creating a multi Node Broker setup for our use.

notroot@ubuntu:~$ jps

5799 Jps

5077 Kafka

4755 QuorumPeerMain

5440 Kafka

notroot@ubuntu:~$ cd lab/software/kafka\_2.11-1.1.0/

notroot@ubuntu:~/lab/software/kafka\_2.11-1.1.0$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 1 --topic my-replicated-topic

Created topic "my-replicated-topic".

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

notroot@ubuntu:~/lab/software/kafka\_2.11-1.1.0$

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

Now the message is there in both the logs

View of how earlier it was 1,2 and then it came to 1 when No 2 broker was killed and when i brought it up again, it came to 2,1

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

notroot@ubuntu:~/lab/software/kafka\_2.11-1.1.0$ kill -9 5077

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: 2 Replicas: 1,2 Isr: 2

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: 2 Replicas: 1,2 Isr: 2,1

===============> Post Lunch Break

Working with Exercise 3 → Having a Console Producer and Console Consumer.

2 deployment techniques

1. copying the source code to lab/programs and then do mvn clean and run it with mvn
2. Creating a jar file in windows → copying the jar file to lab/programs and then running it.

We will start with eclipse now.

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

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

a) broker-id => 0

b) logs.dir ⇒ kafka-log1

c) listener port no → 9092

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

notroot@ubuntu:~/lab/software/kafka\_2.11-1.1.0$ bin/kafka-topics.sh -create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic topic1

Created topic "topic1".

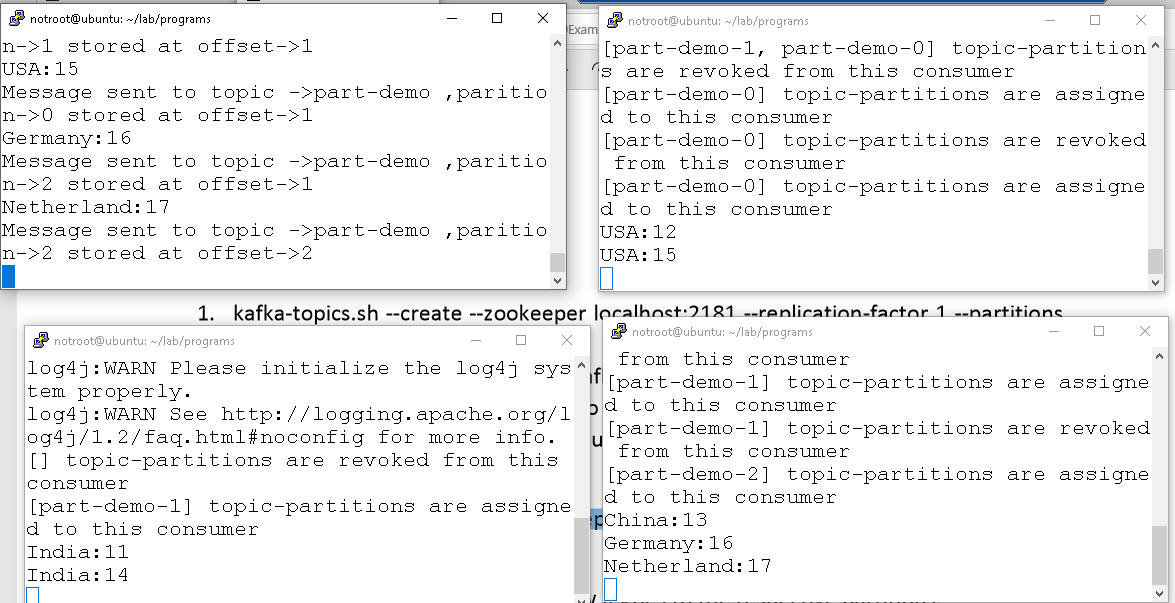
While copying the pom.xml file, ensure that we only copy from <dependencies> till the <build> steps.

notroot@ubuntu:~/lab/programs/Kafka1$ java -cp target/Kafka1-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.citi.SimpleProducer topic1

notroot@ubuntu:~/lab/programs/Kafka1$ java -cp target/Kafka1-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.citi.SimpleConsumer topic1 group1

======================> End of this exercise.

Exercise 4 → Playing with Partitioners



Exercise 5: Custom Serializer →

Compile and run the 5 codes.

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

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

Exercise 6: Confluent → How Schema Registry helps us.

bin/zookeeper-server-start etc/kafka/zookeeper.properties

bin/kafka-server-start etc/kafka/server.properties

bin/schema-registry-start etc/schema-registry/schema-registry.properties

So now we will copy the AvroProducer and AvroConsumer code in our programs directory

We will have to create a java file for ClickRecord and we currently have a avsc file and we will do it in this fashion. Note we will have to do this both on the producer and consumer.

java -jar avro-tools-1.8.1.jar compile schema ClickRecord.avsc .

notroot@ubuntu:~/lab/programs/AvroProducer$ sbt compile

notroot@ubuntu:~/lab/programs/AvroConsumer$ sbt compile

Then we will have to do sbt run on the producer and then do sbt run on the consumer also.

Note: For running this first do the consumer and then do the producer and we should see the output in the consumer as :-

Session id=10001 Channel=HomePage Referrer=null

Now we will copy AvroProducer1 and compile the schema for it.

In the second example,we will see the output like this :-

Session id=10001 Channel=HomePage Referrer=None

Post Exam Link: <https://tinyurl.com/citi-kafka-oct19-1>

Feedback URL: <https://tinyurl.com/citi-kafka-oct19-2>

Contact Details for Venkat:

9821422745

venkatraji71@gmail.com