Kafka, Spark, and Cassandra pipeline in BlueData

Register applications to catalog

1. Login to BlueData controller and sudo to user who installed BlueData
2. Create a new directory to keep catalog bin files and cd to that directory
3. Download Cassandra bin file from following location to current directory https://s3.amazonaws.com/bluedata-vmimages/bluedata-catalog-entdoc-centos-cass_2_1_10-release-2.1.bin
4. Add execute permission to bin Cassandra bin file and run it
5. Download Kafka bin file from following location to current directory https://s3.amazonaws.com/bluedata-vmimages/bluedata-catalog-entdoc-centos-kafka_8_2_2-release-1.0.bin
6. Add execute permission to Kafka bin file and run it
7. Generally it takes upto 24 hours to refresh the catalog from remote feed. For immediate use, download the follwing script, make it executable and run it https://s3.amazonaws.com/bluedata-catalog/appstore_refresh.sh
8. At this point, you are ready to install Kafka and Cassandra from Site Catalog page.

Test Cassandra:

1. Login to Cassandra cluster using “ssh -i “Tenant KeyPair” bluedata@<Cassandra_Node>
2. Run “export CQLSH_HOST=<IP of first cassandra node>”. Example: export CQLSH_HOST=10.39.249.18
3. scl enable python27 bash
4. cqlsh --cqlversion 3.4.2

===

Create instances of clusters

1. Login to BlueData with your credentials
2. Create new clusters of Kafka, Cassandra and Spark. Select a flavor with minimum of 6GB RAM. Accept all other defaults at this time.
3. Verify that your instances are up and running. Note down the ipaddress of the first node in the list of nodes for each cluster.
4. Download the ‘Tenant Keypair’ from Cluster management screen
5. chmod 400 “Tenant Keypair”

Run example 1

Description

The sample program is designed to do the following

1. Generate "consumer complaints" topics generated periodically from a file downloaded from data.gov, into a Kafka queue called "consumer_complaints"
2. Spark cluster will read these complaints and process them on an on-going basis
3. Processed complaints are then stored in Cassandra for iterative and further use.

Generate Kafka events

1. Log into Kafka cluster command line using "ssh -i "Tenant Keypair" bluedata@kafka-node-ip
2. Create a new "src" directory. CD to src
3. Setup the environment for running applications
    - Install Maven to build the sample code using standard Maven documentation. Sample installation from http://preilly.me/2013/05/10/how-to-install-maven-on-centos/
        - $ wget http://mirror.cc.columbia.edu/pub/software/apache/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-bin.tar.gz
        - $ sudo tar xzf apache-maven-3.0.5-bin.tar.gz -C /usr/local
        - $ cd /usr/local
        - $ sudo ln -s apache-maven-3.0.5 maven
        - $ sudo vi /etc/profile.d/maven.sh

          ```
          export M2_HOME=/usr/local/maven

          export PATH=${M2_HOME}/bin:${PATH}
          ```

    - Exit and re-login. cd to src.
    - Run "**mvn -version**" and validate maven home and java home variables

4. If you don't have git installed, run "sudo yum install git"
5. git clone https://github.com/nandav/Realtime-Pipeline.git
6. cd to "Realtime-Pipeline"
7. tar -xvzf Consumer.tgz -C ./file-producer/
8. cd to "file-producer"
9. Edit "src/main/resources/config.properties". **Update the hostname "bluedata-175.bdlocal" with your kafka hostname**
10. Run the command "mvn clean install". If all goes well you should see success when this command finishes running.

11. Create a kafka topic called “**consumer_complains**” by running the following command. Be sure to change the <hostname> with the name of your kafka machine.

/usr/local/kafka_2.10-0.8.2.2/bin/kafka-topics.sh --create --zookeeper <hostname>:2181 --replication-factor 1 --partition 1 --topic consumer_complaints

12. Run the command “**java -cp target/kafkamessages-0.0.1-SNAPSHOT.jar com.bluedata.messages.TestProducer 10**”
13. You should see producer creating messages
14. <ctrl> c (kill process)now to stop the messages at this point.

Consume Kafka events and add data to a Cassandra table using Spark streaming

1. Login to Spark cluster master using “ssh -i “Tenant KeyPair” bluedata@spark-master-ip
2. Create a directory src. “mkdir src”. Then “cd src”
3. You need **sbt** to build the source code. Install sbt using standard sbt documentation. A sample installation is given here. Following this should also work

```
wget http://dl.bintray.com/sbt/rpm/sbt-0.13.5.rpm

sudo yum localinstall sbt-0.13.5.rpm
```

4. Install git to clone “Spark consumer” for consumer complaints. <sudo yum install git>
5. git clone https://github.com/nandav/Realtime-Pipeline.git
6. cd into “Realtime-Pipeline”
7. cd into “spark_consumer”
8. Edit “consumer.conf” file. Modify “spark.master”, “spark.kafka.broker”, and “spark.cassandra.connection.host” to point to the right hosts. spark master should be the hostname of spark master. Kafka broker to point to first node of kafka cluster. Similarly, update Cassandra connection host to the ip address of Cassandra first node.
9. Run “sbt assembly”. Runs for sometime and downloads all the dependencies for Kafka and Cassandra from Spark.
10. Run “sudo bash” to login as root to the container. There are some log files that need “root” permission.
11. If you like less verbose logs from Spark, edit the following file and update INFO to ERROR as shown below.
    a. vi /usr/lib/spark/spark-1.4.0-bin-hadoop2.4/conf/log4j.properties (If you have a different version of Spark, be sure to replace 1.4 with your version of Spark in “**build.sbt**”). Cassandra connector is 1.5.0 of Spark

   b. Update line “log4j.rootLogger=” from INFO to ERROR
   c. Modified line looks like “**log4j.rootLogger=ERROR, file, stdout ,stderr**”
12. Run the following command. Be sure to update <SparkMaster> with actual Spark Master hostname. **spark-submit --properties-file consumer.conf --class KafkaSparkCassandra --master spark://<SparkMaster>:7077 target/scala-2.10/kafka-streaming-cassandra-assembly-1.0.jar**

Check updates on Cassandra cluster

3. Login to Cassandra cluster using “ssh -i “Tenant KeyPair” bluedata@<Cassandra_Node>
4. Run “export CQLSH_HOST=<IP of first cassandra node>”.
   Example: export CQLSH_HOST=10.39.249.18
5. Run “cqlsh”
6. Type command “describe keyspaces;”. You should see ‘bluedata’ as one of the key spaces.
7. Type “use bluedata;”
8. Type “describe tables;”. You should see a table called ‘consumer_complaints’
9. Type “select count(*) from consumer_complaints;”. Counts should keep increasing.
10. Type “Select * from consumer_complaints;”.

Summary:

At the end of this workflow, you should see that Kafka is reading a file, creating messages into Kafka queue. Spark streaming is processing it and storing it in Cassandra. This is a powerful workflow that can be altered to suit your needs.

Running sample 2

Description

The sample program is designed to do the following:

1. Collect tweets into a Kafka queue called “twitter-topic”
2. Spark cluster will read these tweets and process popular hashtags in last 60 seconds, on a continuous basis

Generate Kafka events

1. Log into Kafka cluster command line using “ssh -i “Tenant Keypair” bluedata@kafka-node-ip
2. Create a new “src” directory. CD to src
3. Setup the environment for running applications

a. Install Maven to build the sample code using standard Maven documentation. Sample installation from http://preilly.me/2013/05/10/how-to-install-maven-on-centos/
    1. $ wget http://mirror.cc.columbia.edu/pub/software/apache/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-bin.tar.gz
    2. $ sudo tar xzf apache-maven-3.0.5-bin.tar.gz -C /usr/local
    3. $ cd /usr/local
    4. $ sudo ln -s apache-maven-3.0.5 maven
    5. $ sudo vi /etc/profile.d/maven.sh

```
export M2_HOME=/usr/local/maven

export PATH=${M2_HOME}/bin:${PATH}
```

b. Exit and log back into the shell
c. Run mvn -version and validate maven home and java home variables

4. If you don’t have git installed, run “sudo yum install git”
5. git clone **https://github.com/nandav/twitter-producer.git**
6. cd to “twitter-producer”
7. Edit “config.properties”. **Update the hostname with your kafka hostname and change all twitter access credentials to your credentials. If you do not have twitter app credentials, please login to apps.twitter.com->Create New App following directions. You should be able to obtain all 4 credentials.**
8. Run the command “mvn clean install”. If all goes well you should see success when this command finishes running.
9. Create a kafka topic called “**twitter-topic**” by running the following command. Please make sure to change <hostname> to appropriate name, It is the first node of Kafka cluster where zookeeper is installed. Mostly your current logged in machine. **“/usr/local/kafka_2.10-0.8.2.2/bin/kafka-topics.sh --create --zookeeper <hostname>:2181 --replication-factor 1 --partition 1 --topic twitter-topic**”
10. Start twitter producer using the following command “**java -cp target/kafkamessages-0.0.1-SNAPSHOT.jar com.bluedata.messages.TwitterKafkaProducer”**
11. Open a new shell window and start a kafka console consumer. Be sure to change name of kafka-first-name before running the command. **/usr/local/kafka_2.10-0.8.2.2/bin/kafka-console-consumer.sh --zookeeper <kafka-first-node>:2181 --topic twitter-topic**

Consume Kafka events using Spark streaming

1. Login to Spark cluster master using “ssh -i “Tenant KeyPair” bluedata@spark-master-ip
2. Create a directory src. “mkdir src”. Then “cd src”
3. You need **sbt** to build the source code. Install sbt using standard sbt documentation. A sample installation is given here. Following this should also work

```
wget http://dl.bintray.com/sbt/rpm/sbt-0.13.5.rpm

sudo yum localinstall sbt-0.13.5.rpm
```

6. Install git to clone “Spark consumer” for Kafka tweets. <sudo yum install git>
7. git clone https://github.com/nandav/twitter-spark-consumer.git
8. cd into “twitter-spark-consumer/
9. Edit “consumer.conf” file. Modify “spark.master”, “spark.kafka.broker”, to point to the right hosts. spark master should be the hostname of spark master. Kafka broker to point to first node of kafka cluster.
10. Run “sbt assembly”. Runs for sometime and downloads all the dependencies for Kafka and Spark Streaming. (If you have a different version of Spark, be sure to replace 1.4 with your version of Spark in “**build.sbt**”)
11. Run “sudo bash” to login as root to the container. There are some files that need “root” permission.
12. If you like less verbose logs from Spark, edit the following file and update INFO to ERROR as shown below.
    a. vi /usr/lib/spark/spark-1.4.0-bin-hadoop2.4/conf/log4j.properties (Or modify the spark directory to reflect your cluster)
    b. Update line “log4j.rootLogger=” from INFO to ERROR
    c. Modified line looks like “**log4j.rootLogger=ERROR, file, stdout ,stderr**”
13. Run the following command “**spark-submit --properties-file consumer.conf --class KafkaSparkStreaming --master spark://<Spark-master>:7077 target/scala-2.10/kafka-streaming-assembly-1.0.jar**”
14. You should see popular hashtags in last 60 seconds in rolling fashion.
15. End producer and consumer using <ctrl> c - kill the process