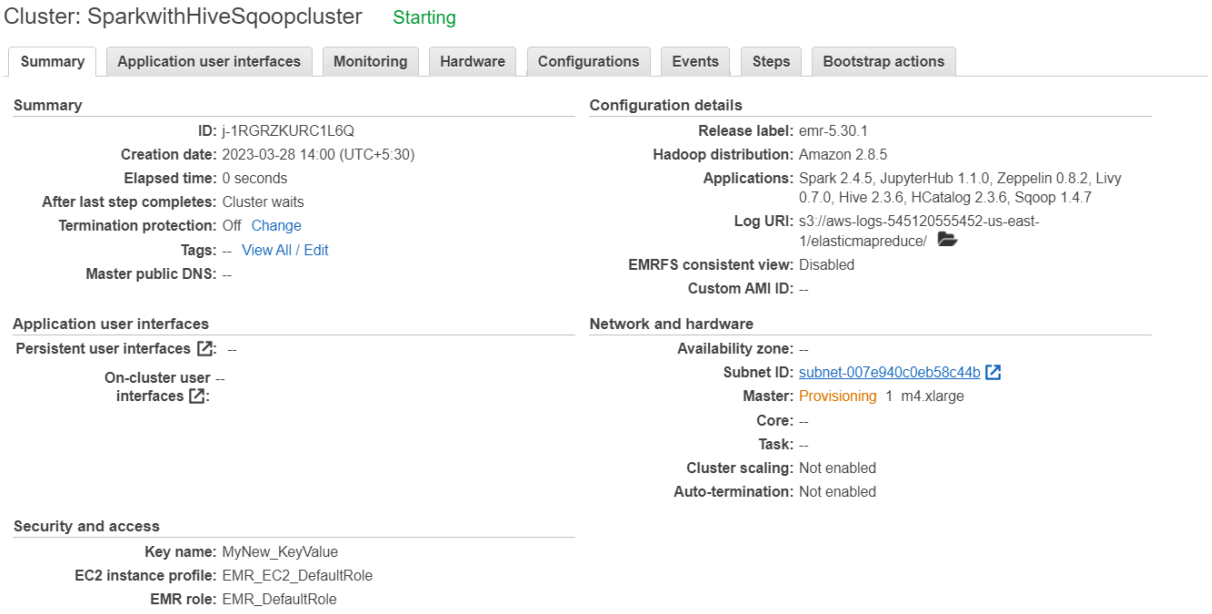
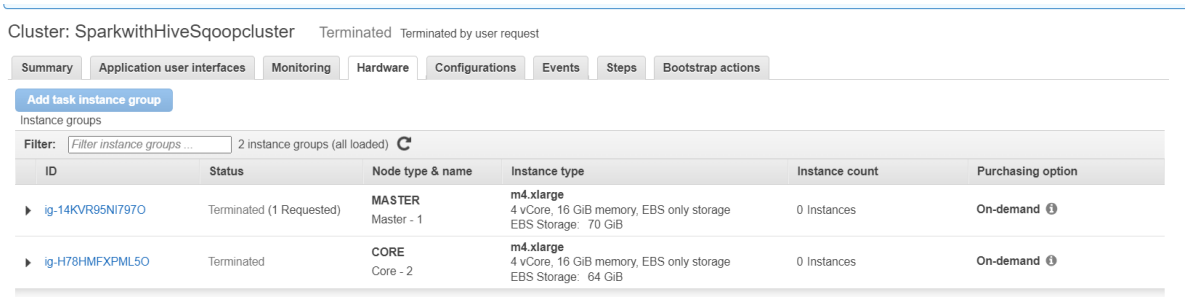
Capstone: Instant Health Alert System – Final Submission

EMR instances

- **EMR cluster** *(with Spark, Hive, Sqoop)*



EMR Hardware Configuration *(with 1 master node and 2 core nodes)*



- **HBase Cluster**

Cluster: Hbasecluster Waiting Cluster ready to run steps.

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Summary

ID: j-1W722OU65RTVS
Creation date: 2023-03-28 13:57 (UTC+5:30)
Elapsed time: 6 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-52-23-186-126.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.30.1
Hadoop distribution: Amazon 2.8.5
Applications: HBase 1.4.13
Log URI: s3://aws-logs-545120555452-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

Persistent user interfaces: YARN timeline server
On-cluster user interfaces: Not Enabled Enable an SSH Connection

Network and hardware

Availability zone: us-east-1b
Subnet ID: subnet-007e940c0eb58c44b
Master: Running 1 m4.xlarge
Core: --
Task: --
Cluster scaling: Not enabled
Auto-termination: Not enabled

Security and access

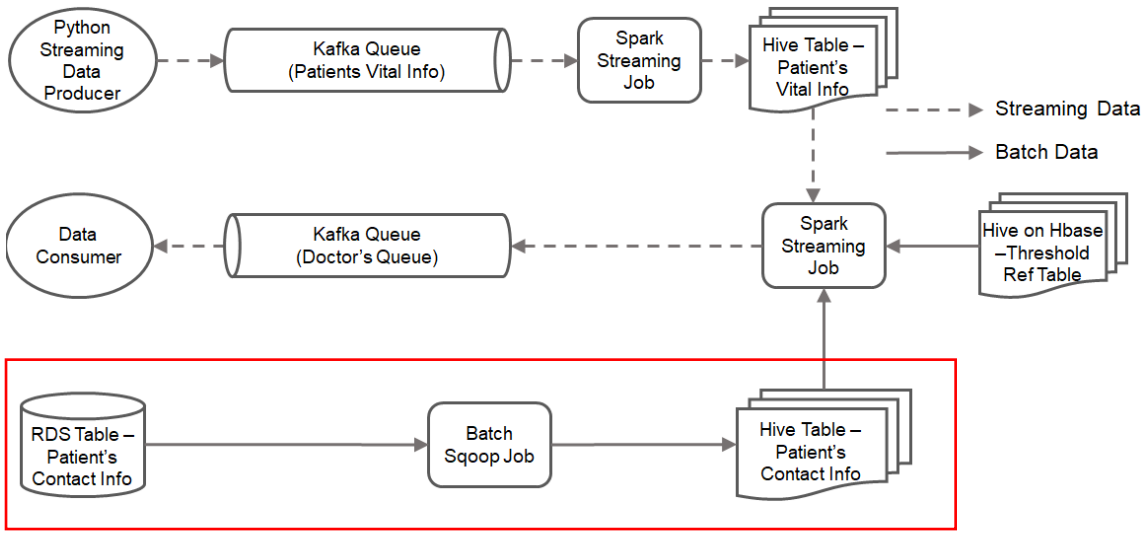
Key name: MyNew_KeyValue
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole

EC2 instance

Covered in PART 3

PART 1:

We will be importing the patient contact info from the RDS using Sqoop and loading the data further into Hive table



1. Import Patient Contact Info records to HDFS using Sqoop
2. Creating an external Hive table for storing Patient's Contact Info

SQOOP SETUP

Following steps are followed to setup Sqoop on EMR Cluster

1. To install the MySQL connector jar file.

```
wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
```

```
[hadoop@ip-172-31-83-130 ~]$ wget https://de-mysql-connector.s3.amazonaws.com/my
sql-connector-java-8.0.25.tar.gz
--2023-03-25 07:08:33--  https://de-mysql-connector.s3.amazonaws.com/mysql-conne
ctor-java-8.0.25.tar.gz
Resolving de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.c
om)... 54.231.130.1, 3.5.16.186, 3.5.20.112, ...
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazona
ws.com)|54.231.130.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4079310 (3.9M) [application/x-gzip]
Saving to: 'mysql-connector-java-8.0.25.tar.gz'

100%[======================================>] 4,079,310   20.6MB/s   in 0.2s

2023-03-25 07:08:34 (20.6 MB/s) - 'mysql-connector-java-8.0.25.tar.gz' saved [40
79310/4079310]
```

2. Extract the MySQL connector tar file

```
tar -xvf mysql-connector-java-8.0.25.tar.gz
```

```
[hadoop@ip-172-31-83-130 ~]$ tar -xvf mysql-connector-java-8.0.25.tar.gz
mysql-connector-java-8.0.25/
mysql-connector-java-8.0.25/src/
mysql-connector-java-8.0.25/src/build/
mysql-connector-java-8.0.25/src/build/java/
mysql-connector-java-8.0.25/src/build/java/documentation/
mysql-connector-java-8.0.25/src/build/java/instrumentation/
mysql-connector-java-8.0.25/src/build/misc/
mysql-connector-java-8.0.25/src/build/misc/debian.in/
mysql-connector-java-8.0.25/src/build/misc/debian.in/source/
mysql-connector-java-8.0.25/src/demo/
mysql-connector-java-8.0.25/src/demo/java/
mysql-connector-java-8.0.25/src/demo/java/demo/
mysql-connector-java-8.0.25/src/demo/java/demo/x/
mysql-connector-java-8.0.25/src/demo/java/demo/x/devapi/
mysql-connector-java-8.0.25/src/generated/
mysql-connector-java-8.0.25/src/generated/java/
mysql-connector-java-8.0.25/src/generated/java/com/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/x/
mysql-connector-java-8.0.25/src/generated/java/com/mysql/cj/x/protobuf/
mysql-connector-java-8.0.25/src/legacy/
mysql-connector-java-8.0.25/src/legacy/java/
mysql-connector-java-8.0.25/src/legacy/java/com/
mysql-connector-java-8.0.25/src/legacy/java/com/mysql/
mysql-connector-java-8.0.25/src/legacy/java/com/mysql/jdbc/
```

3. Go to the MySQL Connector directory created in the previous step and copy it to the Sqoop library to complete the installation.

```
cd mysql-connector-java-8.0.25/
sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/
```

```
[hadoop@ip-172-31-83-130 ~]$ cd mysql-connector-java-8.0.25/
[hadoop@ip-172-31-83-130 mysql-connector-java-8.0.25]$ sudo cp mysql-connector-j
ava-8.0.25.jar /usr/lib/sqoop/lib/
```

4. Set up MySQL on your EMR cluster (Inside this folder mysql-connector-java-8.0.25)

```
mysql_secure_installation
```

Enter current password for root (enter for none): ENTER

Set root password [Y/n] Y
New password: 123
Re-enter password: 123
Remove anonymous users [Y/n] Y
Disallow root login remotely [Y/n] n
Remove test database and access to it [Y/n] Y
Reload privilege tables now [Y/n] Y

```
[hadoop@ip-172-31-83-130 mysql-connector-java-8.0.25]$ mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
 ... skipping.

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
```

```
By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

6. With this, MySQL setup is done. Now, we can access the MySQL shell. Enter the following command, type 123 when the password prompt comes up, and finally, press Enter.

```
mysql -u root -p
```

```
[hadoop@ip-172-31-83-130 mysql-connector-java-8.0.25]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 66
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' identified by '123'
```

7. Inside MariaDB (MariaDB >)
Following queries need to be run for granting all privileges to the root user.

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' identified by '123' WITH GRANT OPTION;
flush privileges;
exit;
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' identified by '123'
WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> exit;
Bye
```

8. Restart the MySQL service to finish setting up MySQL. (Inside this folder mysql-connector-java-8.0.25)

```
sudo service mariadb restart
```

```
[hadoop@ip-172-31-83-130 mysql-connector-java-8.0.25]$ sudo service mariadb rest
art
Redirecting to /bin/systemctl restart mariadb.service
```

9. Change the directory (come outside mysql-connector-java-8.0.25 folder)

```
cd ..
```

Sqoop Commands

1. Import data to HDFS

```
sqoop import --connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/testdatabase --table patients_information --username student --password
STUDENT123 --target-dir /user/livy/patient_contact_info -m 1
```

```
[root@ip-172-31-83-130 ~]# sqoop import --connect jdbc:mysql://upgraddetest.cyai
elc9bmnf.us-east-1.rds.amazonaws.com/testdatabase --table patients_information -
-username student --password STUDENT123 --target-dir /user/livy/patient_contact_
info -m 1
```

```
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=161328
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=3361
                Total vcore-milliseconds taken by all map tasks=3361
                Total megabyte-milliseconds taken by all map tasks=5162496
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=67
                CPU time spent (ms)=1890
                Physical memory (bytes) snapshot=261730304
                Virtual memory (bytes) snapshot=3281002496
                Total committed heap usage (bytes)=247463936
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=230
23/03/25 07:11:39 INFO mapreduce.ImportJobBase: Transferred 230 bytes in 20.9571
 seconds (10.9748 bytes/sec)
23/03/25 07:11:39 INFO mapreduce.ImportJobBase: Retrieved 5 records.
```

2. View the list of files in HDFS target directory

```
hadoop fs -ls /user/livy/patient_contact_info
```

```
[root@ip-172-31-83-130 ~]# hadoop fs -ls /user/livy/patient_contact_info
Found 2 items
-rw-r--r--   1 root livy          0 2023-03-25 07:11 /user/livy/patient_contact_
info/_SUCCESS
-rw-r--r--   1 root livy        230 2023-03-25 07:11 /user/livy/patient_contact_
info/part-m-00000
```

3. View the imported contents in HDFS file

```
hadoop fs -cat /user/livy/patient_contact_info/part-m-00000
```

```
[root@ip-172-31-83-130 ~]# hadoop fs -cat /user/livy/patient_contact_info/part-m
-00000
1,Alex S,XDC test Address,8982739282,1,23,null
2,Sammy A,New Building Address,2382739282,2,45,null
3,Karan C,Aws Address,8923739282,3,56,null
4,Dara M,India Address,2182739282,4,67,null
5,Pam,ABC test Address,4982739282,5,72,null
```

Hive table creation *(for Patients_Contact_Info)*

- Open Hive shell.

```
^C[hadoop@ip-172-31-82-68 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: true
```

- Create a database ***patient_health_care***

```
create database if not exists patient_health_care;
```

```
hive> create database if not exists patient_health_care;
OK
Time taken: 0.855 seconds
```

- Use database patient_health_care

```
use patient_health_care;
```

```
hive> use patient_health_care;
OK
Time taken: 0.046 seconds
```

- Create external table named ***Patients_Contact_Info***

```
CREATE EXTERNAL TABLE IF NOT EXISTS Patients_Contact_Info (
   patientid int,
   patientname string,
   patientaddress string,
   phone_number string,
   admitted_ward int,
   age int,
   other_details string
)
row format delimited
fields terminated by ','
lines terminated by '\n'
location '/user/livy/patient_contact_info';
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS Patients_Contact_Info (
    >     patientid int,
    >     patientname string,
    >     patientaddress string,
    >     phone_number string,
    >     admitted_ward int,
    >     age int,
    >     other_details string
    > )
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > location '/user/livy/patient_contact_info';
OK
Time taken: 0.318 seconds
```

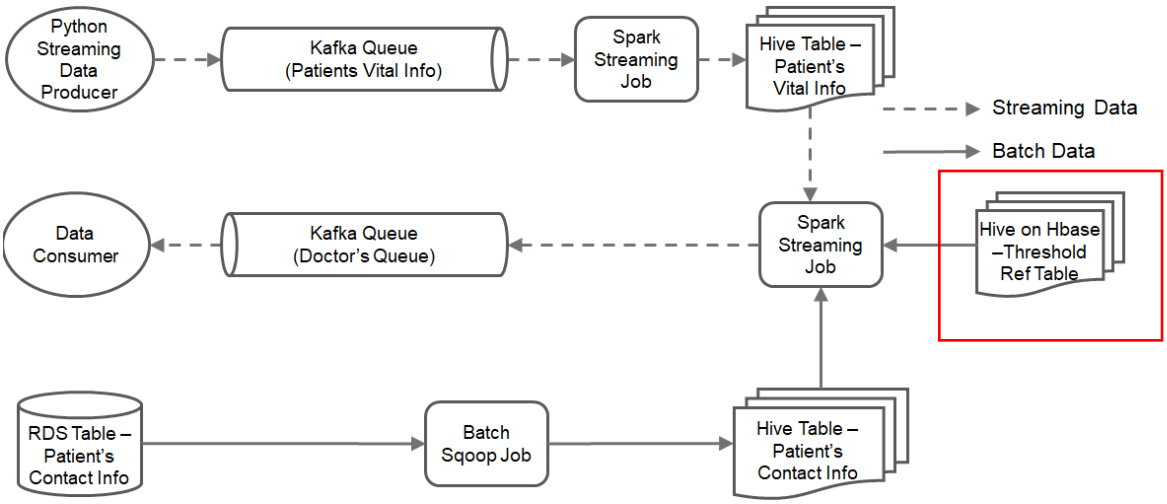
- View the records in ***Patients_Contact_Info*** table

```
select * from Patients_Contact_Info;
```

```
hive> select * from Patients_Contact_Info;
OK
patients_contact_info.patientid patients_contact_info.patientname       patients
_contact_info.patientaddress    patients_contact_info.phone_number      patients
_contact_info.admitted_ward     patients_contact_info.age       patients_contact
_info.other_details
1       Alex S  XDC test Address        8982739282      1       23      null
2       Sammy A New Building Address    2382739282      2       45      null
3       Karan C Aws Address     8923739282      3       56      null
4       Dara M  India Address   2182739282      4       67      null
5       Pam     ABC test Address        4982739282      5       72      null
Time taken: 1.541 seconds, Fetched: 5 row(s)
```

PART 2:

Create an HBase table to store threshold reference information and create a hive external table on top of this HBase table



1. Create a HBase table named **threshold_ref** with 3 column families: attribute, limit, alert
2. Insert 12 records in this HBase table
3. Set up Hive-HBase integration (since HBase and Hive are on separate clusters)
4. Create a Hive external table named **Threshold_Reference_Table** on top of HBase table

Navigate to HBase shell using below commands:

```
sudo -i
hbase shell
```

```
[root@ip-172-31-92-60 ~]# hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.13, rUnknown, Fri Apr 17 15:18:24 UTC 2020
```

Create threshold_ref table in HBase

```
create 'threshold_ref','attribute','limit','alert';
```

```
hbase(main):001:0> create 'threshold_ref','attribute','limit','alert'
0 row(s) in 1.6340 seconds
```

Insert 12 records into HBase table

```
put 'threshold_ref', '1', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '1', 'limit:low_age_limit', '0';
put 'threshold_ref', '1', 'limit:high_age_limit', '40';
put 'threshold_ref', '1', 'limit:low_value', '0';
put 'threshold_ref', '1', 'limit:high_value', '69';
put 'threshold_ref', '1', 'alert:alert_flag', '1';
put 'threshold_ref', '1', 'alert:alert_message', 'Low Heart Rate than Normal';

put 'threshold_ref', '2', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '2', 'limit:low_age_limit', '0';
put 'threshold_ref', '2', 'limit:high_age_limit', '40';
put 'threshold_ref', '2', 'limit:low_value', '70';
put 'threshold_ref', '2', 'limit:high_value', '78';
put 'threshold_ref', '2', 'alert:alert_flag', '0';
put 'threshold_ref', '2', 'alert:alert_message', 'Normal';

put 'threshold_ref', '3', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '3', 'limit:low_age_limit', '0';
put 'threshold_ref', '3', 'limit:high_age_limit', '40';
put 'threshold_ref', '3', 'limit:low_value', '79';
put 'threshold_ref', '3', 'limit:high_value', '9999';
put 'threshold_ref', '3', 'alert:alert_flag', '1';
put 'threshold_ref', '3', 'alert:alert_message', 'Higher Heart Rate than Normal';

put 'threshold_ref', '4', 'attribute:attribute', 'bp';
put 'threshold_ref', '4', 'limit:low_age_limit', '0';
put 'threshold_ref', '4', 'limit:high_age_limit', '40';
put 'threshold_ref', '4', 'limit:low_value', '0';
put 'threshold_ref', '4', 'limit:high_value', '160';
put 'threshold_ref', '4', 'alert:alert_flag', '1';
put 'threshold_ref', '4', 'alert:alert_message', 'Low BP than Normal';

put 'threshold_ref', '5', 'attribute:attribute', 'bp';
put 'threshold_ref', '5', 'limit:low_age_limit', '0';
put 'threshold_ref', '5', 'limit:high_age_limit', '40';
put 'threshold_ref', '5', 'limit:low_value', '161';
put 'threshold_ref', '5', 'limit:high_value', '220';
put 'threshold_ref', '5', 'alert:alert_flag', '0';
put 'threshold_ref', '5', 'alert:alert_message', 'Normal';

put 'threshold_ref', '6', 'attribute:attribute', 'bp';
put 'threshold_ref', '6', 'limit:low_age_limit', '0';
put 'threshold_ref', '6', 'limit:high_age_limit', '40';
put 'threshold_ref', '6', 'limit:low_value', '221';
put 'threshold_ref', '6', 'limit:high_value', '9999';
put 'threshold_ref', '6', 'alert:alert_flag', '1';
put 'threshold_ref', '6', 'alert:alert_message', 'Higer BP than Normal';

put 'threshold_ref', '7', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '7', 'limit:low_age_limit', '41';
put 'threshold_ref', '7', 'limit:high_age_limit', '100';
```

```
put 'threshold_ref', '7', 'limit:low_value', '0';
put 'threshold_ref', '7', 'limit:high_value', '65';
put 'threshold_ref', '7', 'alert:alert_flag', '1';
put 'threshold_ref', '7', 'alert:alert_message', 'Low Heart Rate than Normal';

put 'threshold_ref', '8', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '8', 'limit:low_age_limit', '41';
put 'threshold_ref', '8', 'limit:high_age_limit', '100';
put 'threshold_ref', '8', 'limit:low_value', '66';
put 'threshold_ref', '8', 'limit:high_value', '73';
put 'threshold_ref', '8', 'alert:alert_flag', '0';
put 'threshold_ref', '8', 'alert:alert_message', 'Normal';

put 'threshold_ref', '9', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '9', 'limit:low_age_limit', '41';
put 'threshold_ref', '9', 'limit:high_age_limit', '100';
put 'threshold_ref', '9', 'limit:low_value', '74';
put 'threshold_ref', '9', 'limit:high_value', '9999';
put 'threshold_ref', '9', 'alert:alert_flag', '1';
put 'threshold_ref', '9', 'alert:alert_message', 'Higher Heart Rate than Normal';

put 'threshold_ref', '10', 'attribute:attribute', 'bp';
put 'threshold_ref', '10', 'limit:low_age_limit', '41';
put 'threshold_ref', '10', 'limit:high_age_limit', '100';
put 'threshold_ref', '10', 'limit:low_value', '0';
put 'threshold_ref', '10', 'limit:high_value', '150';
put 'threshold_ref', '10', 'alert:alert_flag', '1';
put 'threshold_ref', '10', 'alert:alert_message', 'Low BP than Normal';

put 'threshold_ref', '11', 'attribute:attribute', 'bp';
put 'threshold_ref', '11', 'limit:low_age_limit', '41';
put 'threshold_ref', '11', 'limit:high_age_limit', '100';
put 'threshold_ref', '11', 'limit:low_value', '151';
put 'threshold_ref', '11', 'limit:high_value', '180';
put 'threshold_ref', '11', 'alert:alert_flag', '0';
put 'threshold_ref', '11', 'alert:alert_message', 'Normal';

put 'threshold_ref', '12', 'attribute:attribute', 'bp';
put 'threshold_ref', '12', 'limit:low_age_limit', '41';
put 'threshold_ref', '12', 'limit:high_age_limit', '100';
put 'threshold_ref', '12', 'limit:low_value', '181';
put 'threshold_ref', '12', 'limit:high_value', '9999';
put 'threshold_ref', '12', 'alert:alert_flag', '1';
put 'threshold_ref', '12', 'alert:alert_message', 'Higher BP than Normal';
```

Screenshot for insertion of records

```
hbase(main):003:0* put 'threshold_ref', '1', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '2', 'limit:high_value', '78';
put 'threshold_ref', '2', 'alert:alert_flag', '0';
put 'threshold_ref', '2', 'alert:alert_message', 'Normal';

put 'threshold_ref', '3', 'attribute:attribute', 'heartBeat';
put 'threshold_ref', '3', 'limit:low_age_limit', '0';
put 'threshold_ref', '3', 'limit:high_age_limit', '40';
put 'threshold_ref', '3', 'limit:low_value', '79';
put 'threshold_ref', '3', 'limit:high_value', '9999';
put 'threshold_ref', '3', 'alert:alert_flag', '1';
put 'threshold_ref', '3', 'alert:alert_message', 'Higher Heart Rate than Normal'
;

put 'threshold_ref', '4', 'attribute:attribute', 'bp';
put 'threshold_ref', '4', 'limit:low_age_limit', '0';
put 'threshold_ref', '4', 'limit:high_age_limit', '40';
put 'threshold_ref', '4', 'limit:low_value', '0';
put 'threshold_ref', '4', 'limit:high_value', '160';
put 'threshold_ref', '4', 'alert:alert_flag', '1';
put 'threshold_ref', '4', 'alert:alert_message', 'Low BP than Normal';

put 'threshold_ref', '5', 'attribute:attribute', 'bp';
put 'threshold_ref', '5', 'limit:low_age_limit', '0';
put 'threshold_ref', '5', 'limit:high_age_limit', '40';
put 'threshold_ref', '5', 'limit:low_value', '161';
put 'threshold_ref', '5', 'limit:high_value', '220';
put 'threshold_ref', '5', 'alert:alert_flag', '0';
```

View records in HBase table

```
scan 'threshold_ref';
```

```
hbase(main):098:0* scan 'threshold_ref'
```

Screenshots of records

```
ROW                   COLUMN+CELL
 1                    column=alert:alert_flag, timestamp=1679727530966, value=1
 1                    column=alert:alert_message, timestamp=1679727530971, value
                      =Low Heart Rate than Normal
 1                    column=attribute:attribute, timestamp=1679727530926, value
                      =heartBeat
 1                    column=limit:high_age_limit, timestamp=1679727530951, valu
                      e=40
 1                    column=limit:high_value, timestamp=1679727530962, value=69
 1                    column=limit:low_age_limit, timestamp=1679727530947, value
                      =0
 1                    column=limit:low_value, timestamp=1679727530958, value=0
 10                   column=alert:alert_flag, timestamp=1679727531306, value=1
 10                   column=alert:alert_message, timestamp=1679727531310, value
                      =Low BP than Normal
 10                   column=attribute:attribute, timestamp=1679727531291, value
                      =bp
 10                   column=limit:high_age_limit, timestamp=1679727531297, valu
                      e=100
 10                   column=limit:high_value, timestamp=1679727531303, value=15
                      0
 10                   column=limit:low_age_limit, timestamp=1679727531294, value
                      =41
 10                   column=limit:low_value, timestamp=1679727531300, value=0
 11                   column=alert:alert_flag, timestamp=1679727531326, value=0
 11                   column=alert:alert_message, timestamp=1679727531329, value
                      =Normal
 11                   column=attribute:attribute, timestamp=1679727531312, value
                      =bp
 11                   column=limit:high_age_limit, timestamp=1679727531318, valu
                      e=100
 11                   column=limit:high_value, timestamp=1679727531323, value=18
                      0
 11                   column=limit:low_age_limit, timestamp=1679727531315, value
                      =41
 11                   column=limit:low_value, timestamp=1679727531321, value=151
```

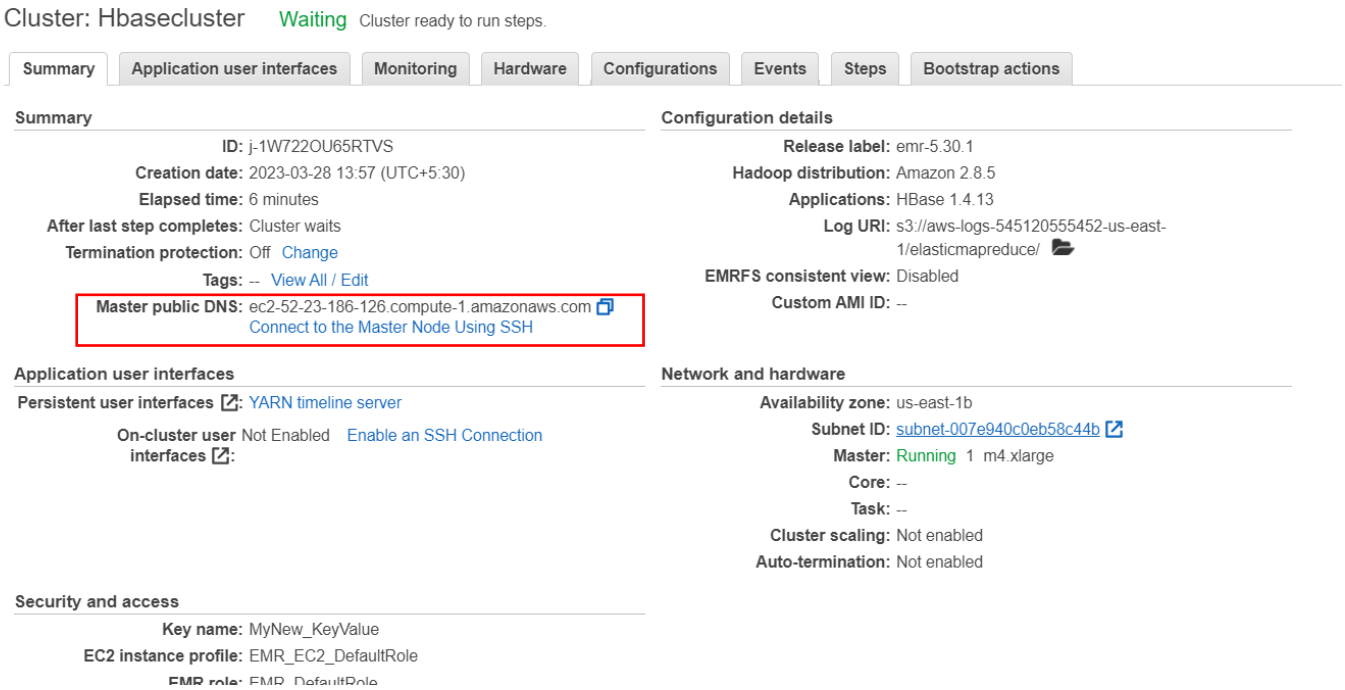
```
 8                    column=alert:alert_message, timestamp=1679992681690, value
                      =Normal
 8                    column=attribute:attribute, timestamp=1679992681657, value
                      =heartBeat
 8                    column=limit:high_age_limit, timestamp=1679992681668, valu
                      e=100
 8                    column=limit:high_value, timestamp=1679992681679, value=73
 8                    column=limit:low_age_limit, timestamp=1679992681663, value
                      =41
 8                    column=limit:low_value, timestamp=1679992681674, value=66
 9                    column=alert:alert_flag, timestamp=1679992681728, value=1
 9                    column=alert:alert_message, timestamp=1679992681734, value
                      =Higher Heart Rate than Normal
 9                    column=attribute:attribute, timestamp=1679992681696, value
                      =heartBeat
 9                    column=limit:high_age_limit, timestamp=1679992681708, valu
                      e=100
 9                    column=limit:high_value, timestamp=1679992681722, value=99
                      99
 9                    column=limit:low_age_limit, timestamp=1679992681702, value
                      =41
 9                    column=limit:low_value, timestamp=1679992681715, value=74
12 row(s) in 0.3570 seconds
```

Threshold_Reference_Table in Hive

```
CREATE EXTERNAL TABLE Threshold_Reference_Table (
  key int,
  Attribute string,
  low_age_limit int,
  high_age_limit int,
  Low_Range_Value int,
  High_Range_Value int,
  Alert_Flag int,
  Alert_Message string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
  'hbase.columns.mapping' = ':key, attribute:attribute, limit:low_age_limit, limit:high_age_limit, limit:low_value,
limit:high_value, alert:alert_flag, alert:alert_message',
  'hbase.table.name' = 'threshold_ref'
)
TBLPROPERTIES ('hbase.mapred.output.outputtable' = 'threshold_ref');
```

Set up for the Hive and HBase integration

HBase Cluster



For the Hive-HBase integration on different clusters, few inbound rules were added to the security group for HBase master node and Hive master node.

Screenshot of HBase cluster's master node (Security Group – master rules)

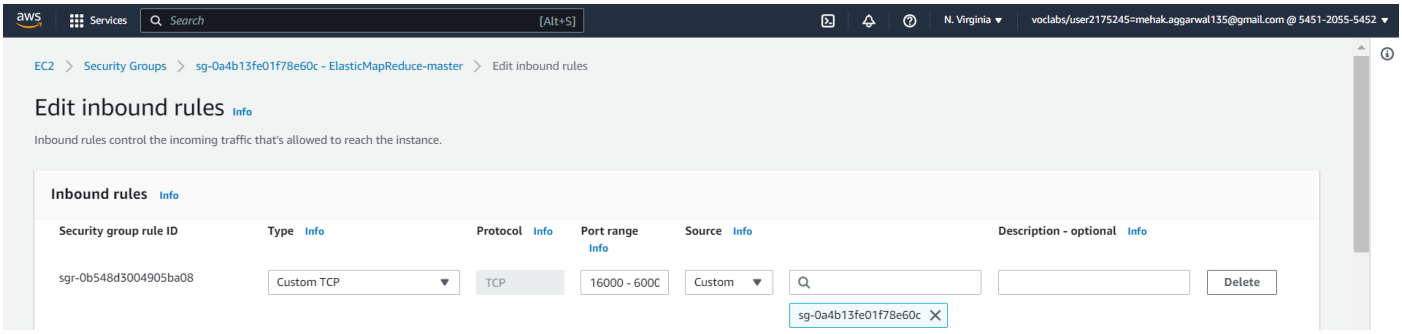
Following rule is added:

Type: "Custom TCP Rule"

Protocol: "TCP"

Port Range: "16000 - 60000"

Source: "Custom" and enter the Master Security Group for Hive cluster's master node.



Screenshot of Hive cluster's master node (Security Group – master rules)

Following rule is added:

Type: "Custom TCP Rule"

Protocol: "TCP"

Port: "10000"

Source: "Custom" and enter the Master Security Group for HBase cluster's master node.

In the Hive Shell

- Connect the HBase client on your Hive cluster to the HBase cluster that contains your data.

set hbase.zookeeper.quorum= <public DNS name of the master node of the HBase cluster>;

```
set hbase.zookeeper.quorum=ec2-52-23-186-126.compute-1.amazonaws.com;
```

```
^C[hadoop@ip-172-31-82-68 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: true
hive> set hbase.zookeeper.quorum=ec2-52-23-186-126.compute-1.amazonaws.com;
```

- Use database patient_health_care

```
use patient_health_care;
```

```
hive> use patient_health_care;
OK
Time taken: 0.046 seconds
```

- Create external table named ***Threshold_Reference_Table***

```
CREATE EXTERNAL TABLE Threshold_Reference_Table (
  key int,
  Attribute string,
  low_age_limit int,
  high_age_limit int,
  Low_Range_Value int,
  High_Range_Value int,
  Alert_Flag int,
  Alert_Message string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
  'hbase.columns.mapping' = ':key, attribute:attribute, limit:low_age_limit, limit:high_age_limit,
limit:low_value, limit:high_value, alert:alert_flag, alert:alert_message',
  'hbase.table.name' = 'threshold_ref'
)
TBLPROPERTIES ('hbase.mapred.output.outputtable' = 'threshold_ref');
```

```
hive> CREATE EXTERNAL TABLE Threshold_Reference_Table (
    >    key int,
    >    Attribute string,
    >    low_age_limit int,
    >    high_age_limit int,
    >    Low_Range_Value int,
    >    High_Range_Value int,
    >    Alert_Flag int,
    >    Alert_Message string
    > )
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >    'hbase.columns.mapping' = ':key, attribute:attribute, limit:low_age_lim
it, limit:high_age_limit, limit:low_value, limit:high_value, alert:alert_flag, a
lert:alert_message',
    >    'hbase.table.name' = 'threshold_ref'
    > )
    > TBLPROPERTIES ('hbase.mapred.output.outputtable' = 'threshold_ref');
OK
Time taken: 2.31 seconds
```

- View the contents of Threshold_Reference_Table

```
set hive.cli.print.header = true;

SELECT * FROM Threshold_Reference_Table order by key;
```

```
hive> set hive.cli.print.header = true;
hive> select * from Threshold_Reference_Table order by key;
Query ID = hadoop_20230328084446_4e04390b-9e15-4799-a8f7-ef03baf510dd
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1679992808558
_0001)

Map 1: -/-      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
```

Screenshot of Threshold_Reference_Table records:

```
OK
threshold_reference_table.key   threshold_reference_table.attribute     threshol
d_reference_table.low_age_limit threshold_reference_table.high_age_limit       t
hreshold_reference_table.low_range_value        threshold_reference_table.high_r
ange_value      threshold_reference_table.alert_flag    threshold_reference_tabl
e.alert_message
1       heartBeat       0       40      0       69      1       Low Heart Rate t
han Normal
2       heartBeat       0       40      70      78      0       Normal
3       heartBeat       0       40      79      9999    1       Higher Heart Rat
e than Normal
4       bp      0       40      0       160     1       Low BP than Normal
5       bp      0       40      161     220     0       Normal
6       bp      0       40      221     9999    1       Higer BP than Normal
7       heartBeat       41      100     0       65      1       Low Heart Rate t
han Normal
8       heartBeat       41      100     66      73      0       Normal
9       heartBeat       41      100     74      9999    1       Higher Heart Rat
e than Normal
10      bp      41      100     0       150     1       Low BP than Normal
11      bp      41      100     151     180     0       Normal
12      bp      41      100     181     9999    1       Higher BP than Normal
Time taken: 15.749 seconds, Fetched: 12 row(s)
```

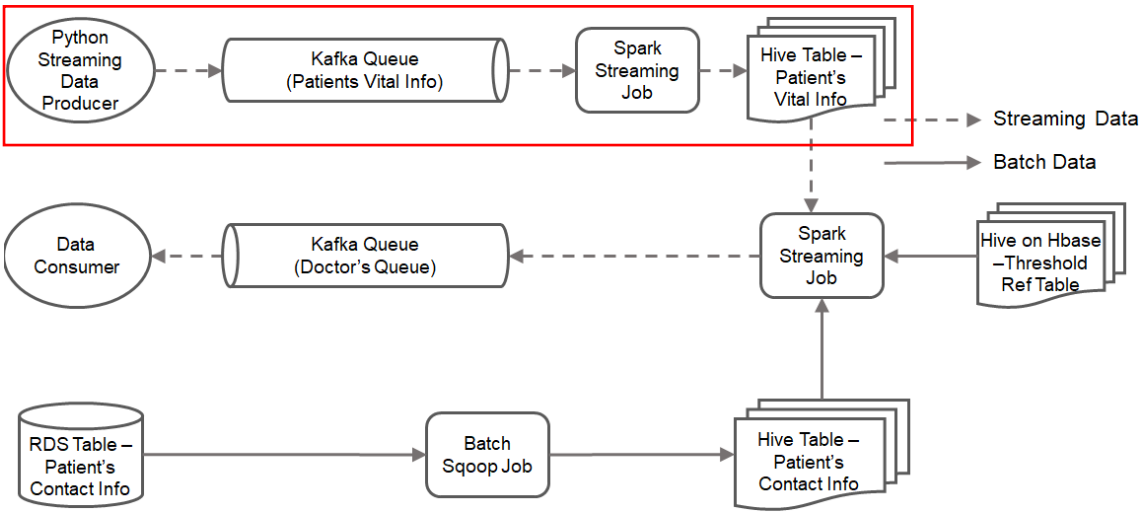
- Create copy of threshold table ***Threshold_Reference*** in Hive and insert the records from Threshold_Reference_Table to Threshold_Reference

This table can be accessed by spark streaming application 2 for Part 4 mentioned below)

```
CREATE EXTERNAL TABLE Threshold_Reference (
  key int,
  Attribute string,
  low_age_limit int,
  high_age_limit int,
  Low_Range_Value int,
  High_Range_Value int,
  Alert_Flag int,
  Alert_Message string
);

INSERT INTO table Threshold_Reference SELECT * FROM Threshold_Reference_Table;
```

PART 3:

Taking streaming data (patients' vital information) and storing it in a table



This involves following tasks:

1. Stimulate streaming data by building a Kafka Producer application in Python to read data from RDS per second. This Kafka producer will push the patient vitals to the Kafka topic 'patient_vital_topic'
2. Write a spark streaming job to read data from Kakfa topic and add the timestamp column and store in HDFS location in parquet format
3. Create an external Hive table 'Patients_Vital_Info' that reads streaming data from HDFS location

There are 1800 records, therefore, the producer will be able to read all the records in 30 minutes.

Kafka EC2 Setup

1. Launch EC2 Instance
2. In the Application and OS Images section, you will need to select the Image to be used for the EC2 instance.
Click on "Browse more AMIs".

3. In the search box that appears at the top, copy and paste the following AMI id and press enter: ami-06c41d8b5a6ddd3c2
In the “Community AMIs” tab, you will find the AMI with the following AMI Name: Kafka_Anaconda-New-2022
Click on the “Select” button to choose the image.
4. Select the General Purpose m4.large type EC2 instance, as shown in the image below.
5. Select the Key Pair to login to the instance via SSH
6. In the “Network Settings” section, go with the default security group. Make sure that the option “Allow SSH traffic from Anywhere” is ticked. This will ensure that you’re able to SSH into the instance from your SSH client
7. In the “Configure storage” settings, you need to enter the volume size as 30 GiB and volume type as standard (magnetic)
8.Once the settings have been updated, click on the “Launch Instance” button to create the instance.
9. Security -> Security groups -> edit inbound rules
Click on Add Rule button and configure the security group as shown below. Enter the following values as shown in the image below
Type: Custom TCP
Port Range: 8888
Source: Anywhere-IPv4
10. Similarly, you need to add the following port numbers:
2181, 9092, 9000, 8080
11. Next click on the Save rules.
12. Go to Elastic IP -> Associate IP with currently running insatnce
13. Login to EC2 machine ,
14. cd /home/ec2-user/downloads/kafka_2.12-2.3.0
15. cd config/
vi server.properties
there is a line 36
#advertised.listeners=PLAINTEXT://your.host.name:9092
Then go to the above line and uncomment it by removing the #. Next in place of your.host.name enter the IPv4 Public IP of your EC2 instance. In this case, that line would read as follows:
advertised.listeners=PLAINTEXT://52.21.15.133:9092
:wq!

- Start Zookeeper server

```
cd downloads/kafka_2.12-2.3.0
bin/zookeeper-server-start.sh config/zookeeper.properties
```

- Start Kafka server

```
cd downloads/kafka_2.12-2.3.0
bin/kafka-server-start.sh config/server.properties
```

- Delete the topic if it already exists and create again

```
cd downloads/kafka_2.12-2.3.0

bin/kafka-topics.sh --bootstrap-server ec2-44-196-94-216.compute-1.amazonaws.com:9092 --delete --topic patient_vital_topic
```

```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic patient_vital_topic
```

- Transfer the Python producer file and Install required packages

```
WINSCP transfer file to downloads/kafka_2.12-2.3.0 folder

pip install mysql-connector-python
pip install mysql-connector-repackaged
```

- Run the producer script

```
python kafka_produce_patient_vitals.py
```

Spark Streaming Job 1 (*kafka_spark_patient_vitals.py*)

It reads data from Kakfa topic and add the timestamp column and store in HDFS location in parquet format.

```
export SPARK_KAFKA_VERSION=0.10
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 kafka_spark_patient_vitals.py
```

Note: In Spark streaming code, we have added a new column using lit('2022-03-16'), and while writing the data we have partitioned the data by this newly added constant column, thus all of the parquet files get written inside this folder /user/livy/output/date=2022-03-16.

This was done because _spark_metadata directory is by default created on the HDFS output directory. In our case, it will get created on path /user/livy/output, and due to partition by functionality, another folder is created named '2022-03-16' inside which the parquet files are present.

In the Hive Shell

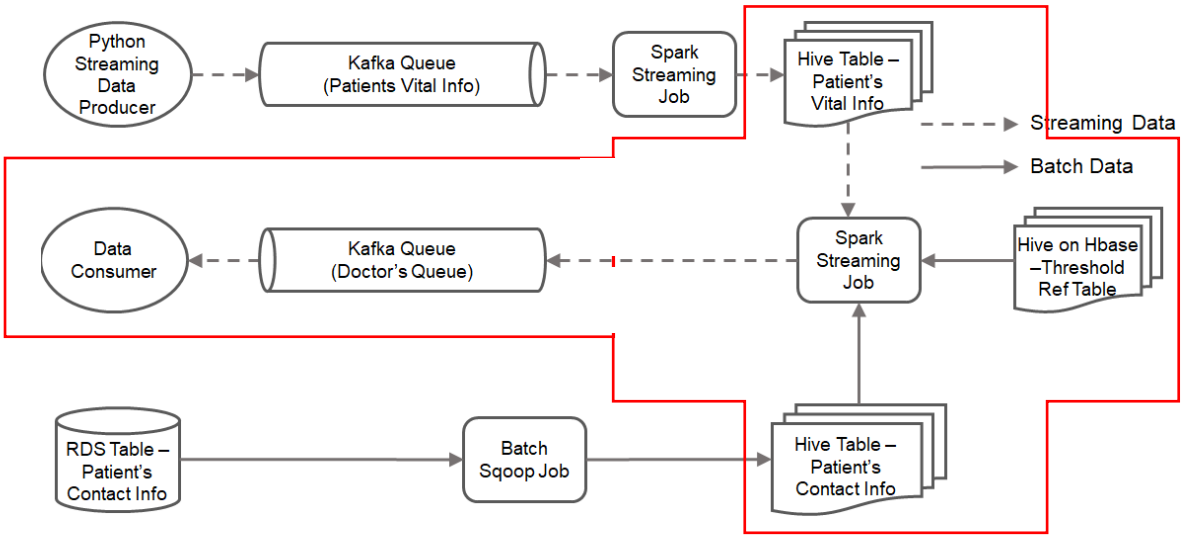
- Create external table 'Patients_Vital_Info'

```
hive

use patient_health_care;
set hive.cli.print.header=true;

CREATE EXTERNAL TABLE IF NOT EXISTS Patients_Vital_Info (
   CustomerID int,
   BP int,
   HeartBeat int,
   Message_time timestamp
)
STORED AS PARQUET
LOCATION '/user/livy/output/date=2022-03-16';
```

```
[root@ip-172-31-82-23 ~]# hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: true
hive> create database if not exists patient_health_care;
OK
Time taken: 1.098 seconds
hive> Show databases;
OK
default
patient_health_care
Time taken: 0.323 seconds, Fetched: 2 row(s)
hive>
    > use patient_health_care;
OK
Time taken: 0.111 seconds
hive>
    >
    > CREATE EXTERNAL TABLE IF NOT EXISTS Patients_Vital_Info (
    >     CustomerID int,
    >     BP int,
    >     HeartBeat int,
    >     Message_time timestamp
    > )
    > STORED AS PARQUET
    > LOCATION '/user/livy/output/date=2022-03-16';
OK
Time taken: 0.333 seconds
```

PART 4:

Comparing the vital information with threshold information and analysing and sending notifications if the data is out of the threshold limits



This involves following tasks:

1. Create a Kafka topic named ***Alerts_Message*** to store irregular patient vitals.
2. Write spark streaming job to read data from 3 hive tables and analyse the patient vitals. If they are irregular, push them to a Kafka topic.
3. Create a Kafka consumer to read messages pushed to the above topic
4. Send email notifications for the messages read by consumer using SNS.

Create a Kafka topic (*Alerts_Message*)

- Delete the topic if it already exists

```
bin/kafka-topics.sh --bootstrap-server ec2-44-196-94-216.compute-1.amazonaws.com:9092 --delete --topic Alerts_Message
```

- Create the topic again

```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Alerts_Message
```

- View the list of topics

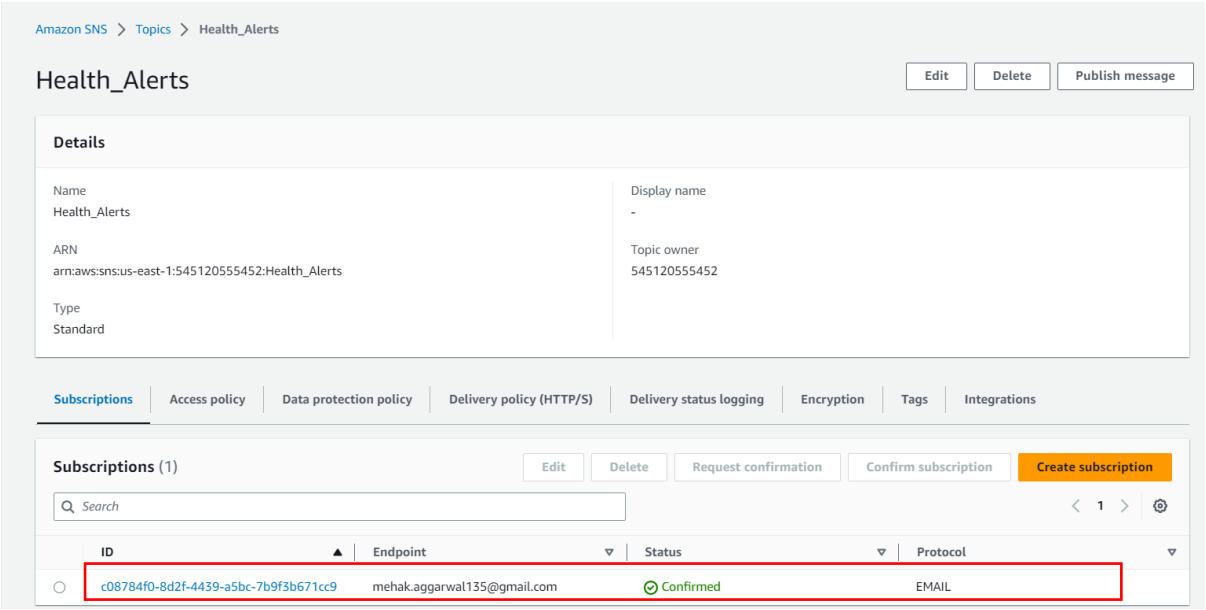
```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

Spark Streaming Job to push irregular patient vitals to 'Alerts_Message' Kafka topic

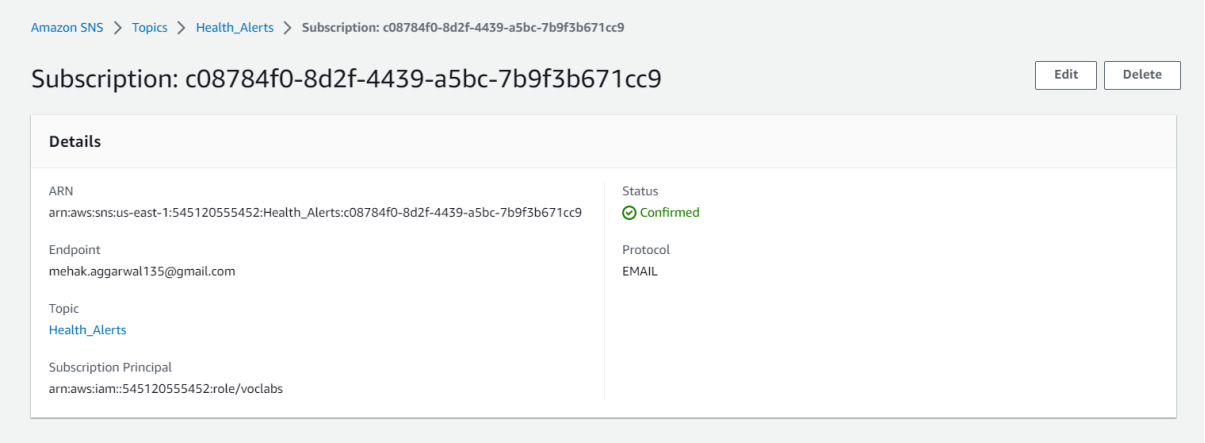
```
export SPARK_KAFKA_VERSION=0.10
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5
kafka_spark_generate_alerts.py
```

Configure SNS

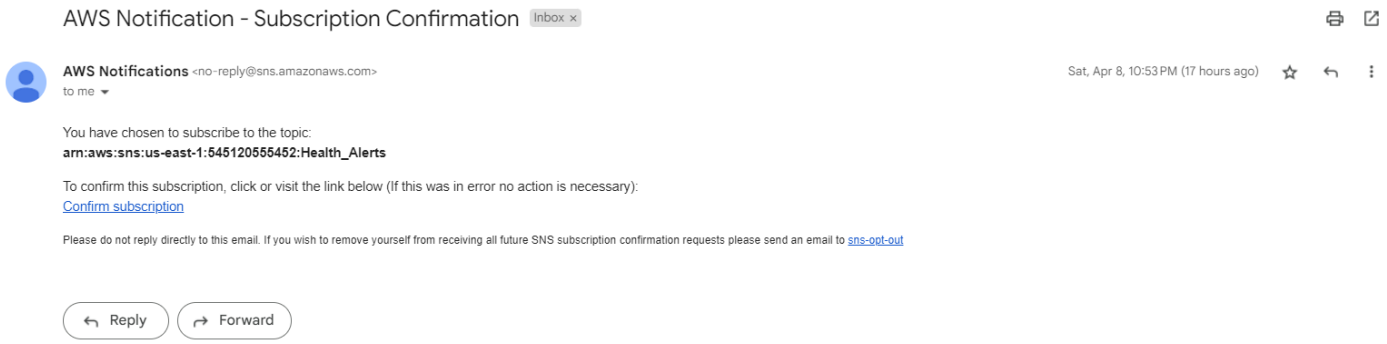
Create SNS topic (Health_Alerts)



Subscribe to SNS topic



Subscription Confirmation Email

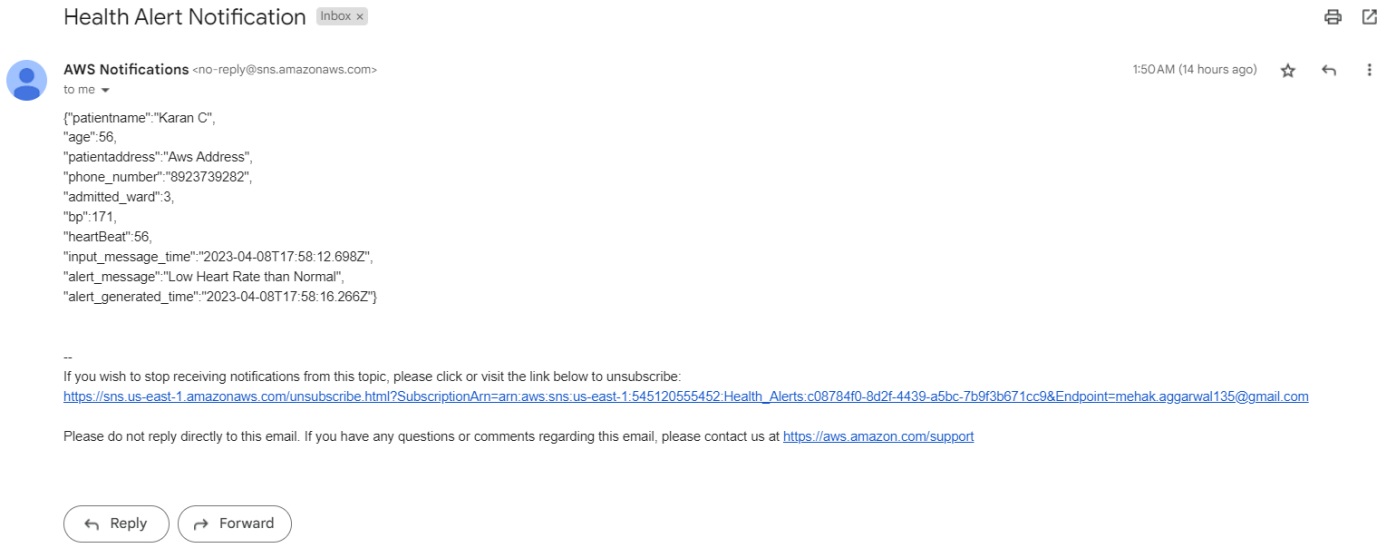


Kafka consumer

```
pip install kafka-python
pip install boto3
python kafka_consume_alerts.py
```

Final Output Screenshot:

Patient health Alert email - using SNS



In short

- Create HBase table, install sqoop and run import
- Then create all external hove tables
- Set up SNS configurations
- Run Kafka zookeeper, server, create topic, producer & consumer in separate SSH terminal windows
- Run both spark streaming jobs