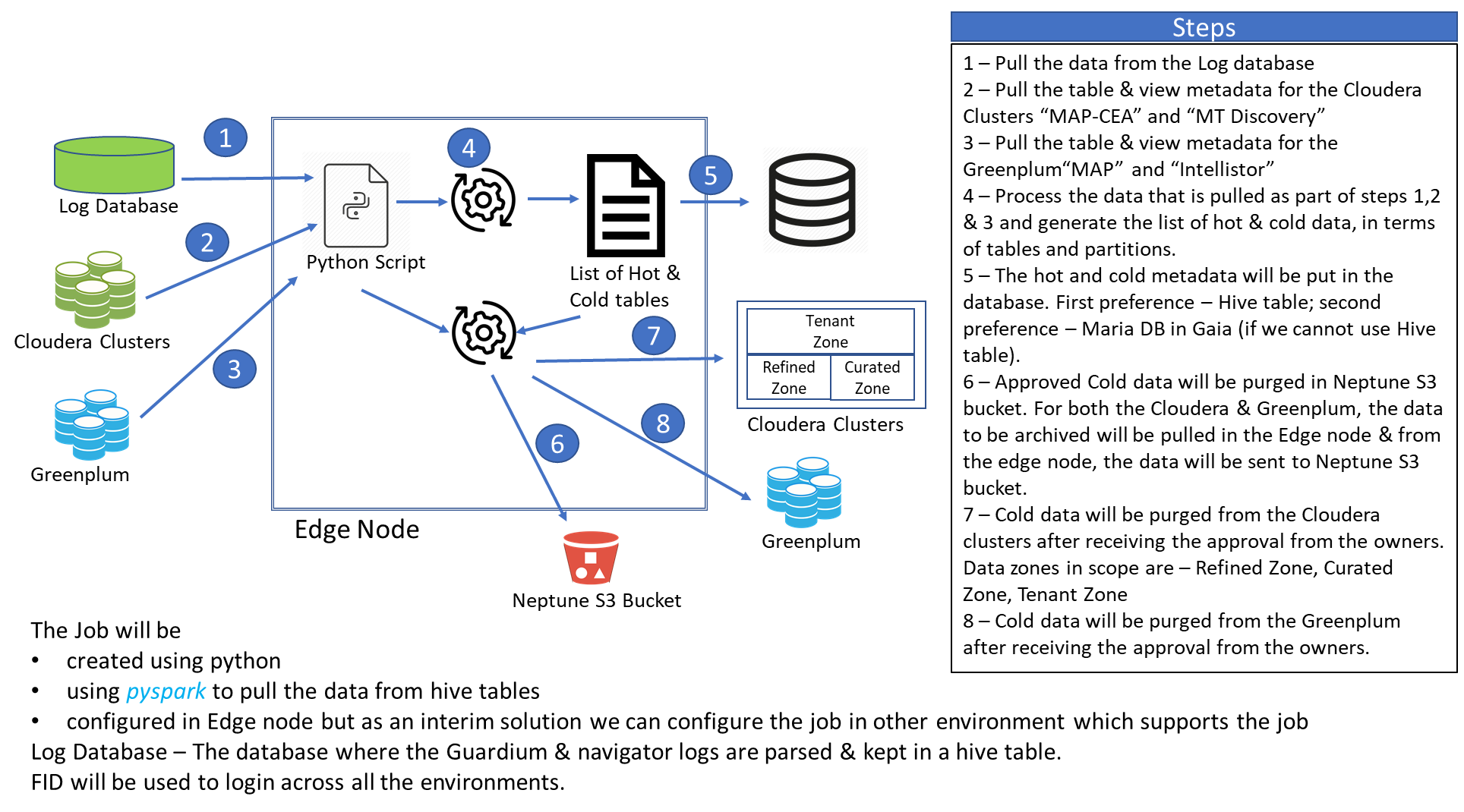
**Hive Metastore/MariaDB/Neptune Connectivity (Using PySpark)**

****

**Prerequisite:**

1. Connection details for
   1. Edge node details where the jobs will be configured
   2. Log database
   3. Cloudera clusters “MAP-CEA” & “MT Discovery”
   4. Greenplum “MAP” & “Intellistor”
   5. Neptune S3 bucket
2. FID that should have read-write permission to all the Cloudera (refined, curated and tenant data zones) and Greenplum clusters
3. Edge node should be able to connect to all the Cloudera and Greenplum clusters
4. Neptune S3 bucket details
5. Following should be present in the edge nodes
   1. Python version 3.x
   2. Pyspark libraries
6. Details of the pyspark libraries that should be present in the edge node

*import pyspark.sql.SparkSession*

*import pyspark.sql.DataFrame*

*import pyspark.sql.Row*

*import pyspark.sql.Column*

*import sys*

*import parquet*

*import avro*

*import json*

*import org.apache.spark.sql.hive.HiveContext;*

*from pyspark.sql import SQLContext*

*from pyspark import SparkConf, SparkContext*

*from pyspark.sql import HiveContext*

*library for Pyspark-Greenplum*

*import psycopg2*

**Required Software’s/Tools:**

* Anaconda 4.3.1 (Jupyter,Spyder)
* Python 3.x
* Java 1.8
* Putty
* Control–M
* FileZilla
* pgAdmin
* Toad for Hadoop
* eSquirreL SQL Client
* Guardium (for logs)
* IntelliJ
* Cloudera-Hive-ODBC Connector

**Code snippets**

* **STEP 1:**

**Pyspark Script to connect Hive Metastore(MySQL):**

*from pyspark import SparkContext, SparkConf*

*from pyspark.sql import SparkSession, HiveContext*

*SparkContext.setSystemProperty("hive.metastore.uris", "thrift://nn1:9083")*

*SparkContext.setSystemProperty("spark.sql.warehouse.dir", "/user/hive/warehouse")*

*sparkSession = (SparkSession*

*.builder*

*.appName('data-usage-analysis')*

*.enableHiveSupport()*

*.getOrCreate()))*

* **STEP 2:**

**Pyspark script to segregate Hot/Cold Data from Hive partition table into Data frames.**

* 1. ***Loading latest 3 months’ data into Data frame from hive (for Hot data analysis)***

*df\_hot\_first = sqlContext.sql('select column1,column2,column3 from schemaName.hive\_table\_Name where Date\_Column <= DATEADD(MONTH, -3, GETDATE())')*

* 1. ***Filter the data accessed more than 10 times and loading into new Data Frame (for Hot data analysis)***

*df\_hot\_second = df\_hot\_first.groupBy("column").count.filter($"count" >= 10)*

***purging hot data – (fid which should have the read and write access to environment) -***

*df\_data\_final = df\_hot\_second.drop\_duplicates(subset ="column1", "column2", "column3", keep = False, inplace = False)*

* 1. ***Loading latest 4 to 6 months’ data into Data frame from hive (for Cold data analysis)***

*df\_cold\_first = sqlContext.sql('select column1,column2,column3 from schemaName.hive\_table\_Name where Date\_Column BETWEEN '2018-04-01' AND '2018-06-30';')*

***purging cold data – (fid which should have the read and write access to environment) -***

*df\_cold\_final = df\_cold\_first.drop\_duplicates(subset ="column1", "column2", "column3", keep = False, inplace = False)*

* **STEP 3:**

**Capturing Hot/Cold Data output into a Parquet/Avro file**

*df\_hot\_second.*write.partitionBy('datelocal').format('parquet').mode(org.apache.spark.sql.SaveMode.Append).save("/user/hive/warehouse/hotdata")

*df\_cold\_final*.write.partitionBy('datelocal').format('parquet').mode(org.apache.spark.sql.SaveMode.Append).save("/user/hive/warehouse/colddata")

* **STEP 4:**

**Load Hot/Cold metadata from Hive table to MariaDB**

**Option 1**

*df\_hot\_second.write.format('jdbc').options(*

*url='jdbc:mysql://mariadbhostname:port/database',*

*driver='com.mysql.jdbc.Driver',*

*dbtable='DestinationTableName',*

*user='FID’,*

*password='XXXXXX’).mode('append').save()*

**Option 2**

Below Sqoop job will be used to load the hive data into Maria DB.

*sqoop export \*

*--connect "jdbc:mariadb://hostname:port/db" \*

*--username FID \*

*--password xxxxxx \*

*--table mariadb\_table \*

*--input-fields-terminated-by ',' \*

*--input-lines-terminated-by '\n' \*

*--export-dir "/user/hive/warehouse/hotdata"*

* **STEP 5:**

**Load archival data from Hadoop to Neptune.**

**Option 1**

1. Existing Bash script used for data ingestion into Neptune.
   1. Bash script bitbucket location.
   2. Unix server: hostname, location of bash script.
2. Oracle database + Table Read access – where Neptune details are stored for existing bash script.
   1. To get the Neptune logon details like Key-Value pair.
   2. Existing bucket access, if it is required to use same bucket to push data etc.
   3. **OR** If we are not supposed to have above Oracle table access, then provide information needed to connect to Neptune.
3. Login access to NEPTUNE for validating.

**Option 2**

* Open up a terminal session of the source Hadoop system.
* Use distcp to move data from Hadoop HDFS to the new S3 bucket

**Reference syntax:**

hadoop distcp -Dfs.s3a.access.key=AKIAHIDEHIDEHIDEHIDE -Dfs.s3a.secret.key=RealLYHidE+ReallYHide+ReallyHide hdfs://{yoursystemname}:{port}/user/hive/warehouse/databaseDirectory/datadirectory/ s3a://{yourbucket}/{somedirectoryStructure}/

**Python script for connecting Greenplum:**

**Option 1**

*import psycopg2*

*try:*

*connection = psycopg2.connect(user = "fid",*

*password = "\*\*\*\*\*\*\*\*",*

*host = "Host\_Name",*

*port = "Port\_Number",*

*database = "GreenPlum")*

*cursor = connection.cursor()*

*# Print Greenplum Connection properties*

*print ( connection.get\_dsn\_parameters(),"\n")*

*# Print Greenplum version*

*cursor.execute("SELECT version();")*

*record = cursor.fetchone()*

*print("You are connected to - ", record,"\n")*

*except (Exception, psycopg2.Error) as error:*

*print ("Error while connecting to Greenplum", error)*

*finally:*

*#closing database connection.*

*if(connection):*

*cursor.close()*

*connection.close()*

*print("Greenplum connection is closed")*

**Option 2**

* Import the data from Greenplum to Hive using Sqoop.

**Sqoop Syntax:**

sqoop import

--connect jdbc://<host\_name>:<port\_no>/<database>

--username FID

--password xxxxxx

--query "select col1 ,col2, col3, col4 from <table\_name>

--split-by col1

--target-dir /hadoop/analytics\_data/